Optimization and Applications of Extreme Learning Machine Method

Doctoral Dissertation
(Rozprawa doktorska)

M.Sc. Eng. Karol Struniawski
(mgr inż. Karol Struniawski)

Polish-Japanese Academy of Information Technology
Faculty of Computer Science
(Polsko-Japońska Akademia Technik Komputerowych
Wydział Informatyki)

Supervisor:
D.Sc. Ryszard Kozera
(Promotor: dr hab. Ryszard Kozera)

Warsaw, 2025
(Warszawa, 2025)

(This page contains a formal declaration of authorship and originality, in accordance with institutional and Polish copyright regulations.)

Oświadczenie

Ja, niżej podpisany Karol Struniawski, autor rozprawy doktorskiej pt. „Optimization and Applications of Extreme Learning Machine Method", oświadczam, iż wyżej wskazaną rozprawę napisałem samodzielnie i żaden jej fragment lub całość nie był pisany przez osobę trzecią. Jednocześnie oświadczam, iż:

- praca nie była wcześniej podstawą nadania stopnia doktora innej osoby,
- załączona wersja elektroniczna jest tożsama z wydrukiem rozprawy,
- wszystkie elementy pracy, które zostały wykorzystane do jej realizacji, a nie będące mojego autorstwa, zostały odpowiednio oznaczone oraz zostało podane źródło ich pochodzenia,
- przedstawiona przeze mnie wyżej wskazana praca nie narusza przepisów ustawy z dnia 4 lutego 1994 r. o prawach autorskich i prawach pokrewnych (tj. z dnia 17 maja 2006 r. Dz.U. Nr 90, poz. 631 z późn. zm.).

Mam również świadomość, iż złożenie nieprawdziwego oświadczenia skutkować będzie niedopuszczeniem do dalszych czynności postępowania w sprawie nadania stopnia doktora lub cofnięciem decyzji o nadaniu mi stopnia doktora oraz wszczęciem postępowania dyscyplinarnego.

Abstract

This dissertation focuses on the development, optimization, and applications of the Extreme Learning Machine (ELM) method across various fields, with particular emphasis on: (a) *the creation of programming tools and the search for energy-efficient deployment platforms for ELM*, (b) *algorithmic improvements for ELM*, and (c) *practical applications of ELM together with other Machine Learning methods in microorganisms' identification.* This order reflects the logical structure of the research, as the development of the TfELM platform in (a) served as the computational foundation for all subsequent experiments and analyses presented in (b-d).

(a) One of the key contributions of this work is the development of *TfELM*, a flexible software programming platform developed from scratch in Python, fully integrated with *TensorFlow*, *CUDA*, and *scikit-learn* libraries. This open-source tool, available through the Python Package Index (PyPI) and GitHub, supports 18 ELM variants and is fully compatible with *scikit-learn* conventions, enabling automatic computation of metrics during cross-validation, application of Explainable Artificial Intelligence (XAI) techniques, and model saving and loading. Performance evaluations demonstrated that *TfELM* outperforms existing implementations in computational efficiency, due to its compatibility with the latest versions of *TensorFlow* and *CUDA*, as well as the adoption of techniques such as Nyström approximation. The dissertation also addresses the impact of hardware architecture on the performance of machine learning algorithms, including ELM. The energy efficiency of Apple Silicon (M1, M2, M3 and M4) processors was analyzed and compared to traditional GPU configurations, such as the NVIDIA RTX 3090. The results confirmed that M2 processors can offer an energy-efficient alternative for selected ELM tasks while still maintaining high computational performance.

(b) A significant part of the dissertation is dedicated to the algorithmic optimization of ELM. Research on metaheuristic algorithms (MAs) demonstrated that techniques such as the Salp Swarm Algorithm (SSA) and Manta Ray Foraging Optimization (MRFO) can significantly enhance the accuracy of ELM models. Further studies focused on optimizing the selection of activation function parameters for ELM using MAs, resulting in noticeable improvements in classifier performance. A comprehensive analysis of 36 activation functions, including Mish and Sexp-previously unused in ELM-was conducted, confirming the critical importance of activation function selection for classification outcomes. Moreover, the dissertation provides a detailed investigation into the influence of randomness in the weight initialization process in ELM. It was demonstrated that the choice of random number generator and probability distribution function substantially impacts on the stability and on accuracy of the used models. Experimental results indicated that the Pareto distribution in most cases outperforms the uniformly distributed initialization commonly used in ELM.

(c) Another key element of this dissertation is the practical application of ELM in the analysis of biological images. A series of studies focused on the identification of soil microorganisms using machine learning and image processing techniques. One of the issues studied in this dissertation compares the effectiveness, in terms of the model precision for ELM, Support Vector Machines, and Random Forests in the classification of soil bacteria, demonstrating the superiority of ELM in single-cell classification tasks using the *TfELM* library and algorithmic enhancements based on insights from ELM optimization process. Subsequent research exploits also the use of Convolutional Neural

Networks (CNNs) to identify fungi and Chromista, confirming the accuracy of single-instance analysis even in datasets with substantial morphological diversity. The most extensive study addresses the automatic identification of soil microorganisms cultivated on agar media. With the aid of developed *TfELM* package, the ELM and CNN precision and execution time is subsequently analyzed. This is achieved upon comparing calculated features (color, texture, and shape descriptors) with traits extracted via CNNs. Notably, manually computed features usually proved more effective than those extracted from deep convolutional networks.

In summary, this dissertation attempts to make a substantial contribution to the advancement of ELM by developing modern programming software tools (*TfELM* library), enhancing algorithmic techniques through SSA and MRFO, and demonstrating practical applications in biological image analysis. The research hypothesis of this dissertation is that the ELM can be significantly improved by enhancing its algorithmic techniques-specifically through the incorporation of the aforementioned extensions. These enhancements not only boost ELM's performance but also broaden its applicability to real-world, data-driven problems. The results offer a solid foundation for future research into other applications of ELM and its variants across various machine learning tasks.

Streszczenie

Niniejsza rozprawa koncentruje się na opracowaniu rozszerzeń i usprawnień, optymalizacji oraz zastosowaniach metody Extreme Learning Machine (ELM) w różnych dziedzinach, ze szczególnym uwzględnieniem a) *tworzenia narzędzi programistycznych oraz poszukiwaniu efektywnych energetycznie platform uruchomieniowych dla ELM*, b) *opracowaniu i realizacji usprawnień algorytmicznych dla ELM* oraz c) *praktycznych zastosowaniach ELM i innych metod uczenia maszynowego w zadaniach identyfikacji mikroorganizmów ryzosferowych*. Taka kolejność odzwierciedla logiczną kolejność badań, ponieważ opracowanie platformy TfELM w części (a) stanowiło podstawę obliczeniową dla wszystkich dalszych eksperymentów i analiz przedstawionych w częściach (b-d).

a) Podstawę pracy stanowi *TfELM* – stworzona przez autora tej rozprawy od podstaw elastyczna platforma programistyczna napisana w języku Python, w pełni zintegrowana z bibliotekami *TensorFlow*, *CUDA* oraz *scikit-learn*. To otwarte oprogramowanie, dostępne w menadżerze pakietów dla języka Python - PyPI oraz GitHub wspiera 18 wariantów ELM i jest zgodne z konwencjami biblioteki *scikit-learn*, co umożliwia m.in. automatyczne obliczanie metryk w walidacji krzyżowej, zastosowanie Wyjaśnialnej Sztucznej Iteligencji (eng. Explainable AI) oraz zapisywanie i wczytywanie rozważanych modeli. Przeprowadzone testy wydajnościowe wykazały, że *TfELM* przewyższa istniejące implementacje pod względem efektywności obliczeń (krótszy czas nauczania i testowania modelu) dzięki zgodności z najnowszymi wersjami bibliotek *TensorFlow* i *CUDA* oraz zastosowaniu metod takich jak aproksymacja Nyströma. W rozprawie podjęto także temat wpływu architektury sprzętowej na wydajność algorytmów uczenia maszynowego, w tym ELM. Przeanalizowano możliwości energooszczędnych układów Apple Silicon (M1-M4), porównując ich wydajność z tradycyjnymi konfiguracjami GPU, takimi jak NVIDIA RTX 3090. Wyniki badań potwierdziły, że procesory M2 mogą stanowić efektywną energetycznie alternatywę dla wybranych zadań modeli typu ELM, przy zachowaniu wysokiej wydajności obliczeniowej.

b) Znaczną część pracy poświęcono optymalizacji algorytmicznej ELM. W badaniach nad Metaheurystycznymi Algorytmami (MA) wykazano, że techniki takie jak Salp Swarm Algorithm (SSA) czy Manta Ray Foraging Optimization (MRFO) pozwalają na zwiększenie dokładności modeli ELM. Kolejne badania skupiły się na optymalizacji doboru wartości parametrów dla różnych funkcji aktywacji do ELM za pomocą algorytmów MA, co przyniosło zauważalny wzrost skuteczności klasyfikatorów. Przeprowadzono również kompleksową analizę 36 funkcji aktywacji, w tym też dotąd niewykorzystywanych w ELM funkcji Mish i Sexp, potwierdzając kluczowe znaczenie doboru funkcji aktywacji dla wyników klasyfikacji. W rozprawie szczegółowo przeanalizowano także wpływ losowości wyboru wag w procesie inicjalizacji wag w ELM. Badania wykazały, że wybór generatora liczb losowych oraz funkcji rozkładu prawdopodobieństwa może znacząco wpłynąć na stabilność i dokładność modeli. Eksperymenty dowiodły, że rozkład Pareto często przewyższał najczęściej stosowane w ELM rozkłady jednostajne.

c) Ważnym elementem pracy jest również testowanie praktycznego zastosowania ELM w analizie obrazów biologicznych (tj. do klasyfikacji bakterii i Chromist zamieszkujących ryzosferę). W serii różnorodnych badań skoncentrowano się na identyfikacji mikroorganizmów glebowych z wykorzystaniem uczenia maszynowego oraz innych technik przetwarzania obrazów (np. filtrowanie, segmentacja, czy ekstrakcja). W jednej z prac porównano efektywność ELM, maszyn wektorów nośnych oraz lasów losowych w klasyfikacji bakte-

rii glebowych. Wyniki wykazały wyższość ELM, w kontekście dopasowania do danych oraz szybkości nauczania i testowania, w zadaniach klasyfikacji pojedynczych komórek wykorzystując stworzoną bibliotekę *TfELM* do obliczeń oraz usprawnienia sieci na bazie wiedzy ugruntowanej w publikacjach poświęconym optymalizacji sieci ELM. W kolejnych badaniach zastosowano konwolucyjne sieci neuronowe (eng. Convolutional Neural Network - CNN) do identyfikacji grzybów i Chromist potwierdzając efektywność podejścia opartego na analizie pojedynczych instancji nawet przy dużym zróżnicowaniu morfologicznym badanych danych. Najobszerniejsze studium poświęcono automatycznej identyfikacji mikroorganizmów glebowych hodowanych na pożywkach agarowych. Wykorzystując wcześniej stworzony pakiet *TfELM*, przeanalizowano skuteczność klasyfikatorów ELM i CNN, porównując cechy obliczone ręcznie w zestawieniu z tymi uzyskanymi przy użyciu CNN. Co istotne, cechy wyliczane z obrazu, które obejmowały cechy koloru, tekstury oraz kształtu, okazały się często bardziej skuteczne pod względem dopasowania modelu niż te uzyskane z głębokich sieci konwolucyjnych.

Podsumowując niniejsza rozprawa ma na celu wniesienie istotnego wkładu w rozwój ELM poprzez opracowanie nowoczesnych narzedzi programistycznych takich jak biblioteka *TfELM*, optymalizację algorytmiczną SSA i MRFO oraz praktyczne zastosowania w analizie obrazów biologicznych. Hipoteza badawcza niniejszej rozprawy zakłada, że ELM może zostać istotnie udoskonalona poprzez rozwój jej technik algorytmicznych - w szczególności dzięki zastosowaniu wspomnianych wcześniej rozszerzeń. Zmiany te nie tylko poprawiają wydajność ELM, ale także poszerzają zakres jego zastosowań w rozwiązywaniu rzeczywistych, opartych na danych problemów. Uzyskane wyniki stanowią solidną podstawę do dalszych badań nad wykorzystaniem ELM i jego wariantów w innych zadaniach uczenia maszynowego.

Acknowledgements

A full version of the acknowledgements is provided in Polish.

I express my sincere gratitude to my supervisor, Professor Ryszard Kozera, for his invaluable guidance and support throughout every stage of my research and publication process. I also wish to thank Professor Lidia Sas-Paszt, Professor Agnieszka Marasek-Ciołek and Dr. Paweł Trzciński from the Institute of Horticulture in Skierniewice for their essential contributions including biological insights, access to microscopy facilities, and the preparatory datasets for my experiments. Finally, I am grateful to all who supported me during my PhD research work and preparation of this dissertation.

Podziękowania

Pragnę wyrazić serdeczne podziękowania mojemu promotorowi, profesorowi Ryszardowi Kozerze, za nieocenione wsparcie, opiekę naukową oraz cenne wskazówki podczas realizacji każdej z publikacji składających się na niniejszą rozprawę. Zaangażowanie mojego Promotora, wiedza oraz doświadczenie były kluczowe na każdym etapie – od planowania badań, przez tworzenie, aż po końcową fazę tj. edycję publikacji. Dzięki Panu miałem możliwość poszerzania swojej wiedzy i umiejętności pod okiem wybitnego specjalisty, w tym również rozwinięcie moich umiejętności prezentacji wyników prac naukowych w publikacjach oraz podczas odczytów konferencyjnych.

Dziękuję wszystkim życzliwym osobom, które wspierały mnie w trakcie przygotowywania tej rozprawy. Wspacie, motywacja i zromienie tych osób były dla mnie nieocenionym źródłem siły w trakcie całego procesu pracy nad tą rozprawą.

Szczególne podziękowania kieruję również do profesor Lidii Sas-Paszt z Instytutu Ogrodnictwa w Skierniewicach za inspirację i pomysł na realizację badań związanych z identyfikacją mikroorganizmów na próbkach biologicznych przy użyciu metod uczenia maszynowego oraz zdjęć mikroskopowych. Pani ogromna wiedza, zaangażowanie oraz wsparcie w opracowaniu części biologicznych publikacji, a także pomoc w procesie odpowiadania na recenzje były nieocenione i miały kluczowe znaczenie dla wysokiej jakości tych prac. Jestem również wdzięczny za możliwość skorzystania z danych zgromadzonych w ramach grantu EcoFruits, realizowanego w programie BIOSTRATEG, dzięki któremu mogłem wykorzystać mikroskopowe zdjęcia mikroorganizmów do moich badań.

Dziękuję profesor Agnieszce Marasek-Ciołek z Instytutu Ogrodnictwa w Skierniewicach za udostępnienie mikroskopu, który umożliwił mi masowe wykonywanie zdjęć próbek i stworzenie największego zbioru obrazów organizmów ryzosferowych.

Słowa podziękowania kieruję także do doktora Pawła Trzcińskiego z Instytutu Ogrodnictwa w Skierniewicach za przeprowadzenie hodowli mikroorganizmów oraz wykonanie zdjęć wstępnego zbioru obrazów, które stanowiły fundament dalszych badań.

Badania finansowane przez Narodowe Centrum Badań i Rozwoju
w ramach programu BIOSTRATEG
numer umowy BIOSTRATEG3/344433/16/NCBR/2018

Tytuł projektu:
NOWE ROZWIĄZANIA BIOTECHNOLOGICZNE W DIAGNOSTYCE, ZWALCZANIU I MONITORINGU KLUCZOWYCH PATOGENÓW GRZYBOWYCH W EKOLOGICZNEJ UPRAWIE OWOCÓW MIĘKKICH

Contents

Chapter 1

Introduction

The foundations of Machine Learning (ML) stretch back to the 19th and early 20th centuries, with key advancements in mathematics, logic, and cognitive science [1, 2]. Statistical and probabilistic methods, such as Bayesian probability introduced by P. Laplace, the Gaussian distribution formalized by C.F. Gauss and F. Galton's concepts of correlation and regression analysis, established fundamental principles for data-driven learning [3]. At the same time, early theories of learning and cognition emerged, with H. Helmholtz proposing that both perception and cognition rely on inference, such concept that also resonates with modern probabilistic learning models [4]. W. James and I. Pavlov explored associative learning, emphasizing how experiences shape knowledge [5]. The evolution of ML was not solely reliant on statistical methods. Indeed the new concepts required computation power related to theoretical advancements. G. Boole's development of boolean algebra provided the logical framework for decision-making process in machines, while C. Babbage, alongside A. Lovelace, envisioned the first mechanical computer. Additionally, the contributions of D. Hilbert and B. Russell in formal logic helped to determine the principles of algorithmic reasoning [6]. All these intellectual advancements converged in 1943 with the pioneering work of W. McCulloch and W. Pitts, who proposed a model of artificial neurons resembling the biological ones [7]. Their work laid the foundation for ML, following decades of research in logic, probability, and cognitive science.

The proposed model was further expanded by D. Hebb's idea that neurons can be wired together that mimics the strengthening of synaptic connections based on experience [8]. Building upon these ideas F. Rosenblatt, a psychologist at Cornell University, developed in 1958 the perceptron as a practical model of learning based on McCulloch-Pitts neurons and Hebbian principles [9]. This model was capable to separate two linearly-separable classes using a linear decision boundary (upon finding appropriate weights). Initially, Rosenblatt claimed that it could eventually be extended to recognize complex patterns and even lead to machines with human-like intelligence (that nowadays is commonly called General Artificial Intelligence) [10]. The promises emerged U.S. Navy to finance the Mark I Perceptron hardware, to demonstrate its capabilities. The excitement surrounding halt in 1969 when M. Minsky and S. Papert mathematically proved the perceptron could only learn linearly separable functions [11]. The criticism in Perceptrons led to a decline in neural network research throughout the 1970s, a period often referred to as the first AI winter.

In 1967, S. I. Amari was the first to introduce the use of gradient-based method for training multilayer neural networks [12]. Noteworthy, his work remained largely unnoticed until 1986, when G. Hinton, D. Rumelhart, and R. J. Williams published a paper

demonstrating that backpropagation could effectively train multilayer perceptrons (MLP) [13] that hasn't cited the Amari work. His contribution was also overlooked by the 2024 Nobel Prize Committee that rewarded G. Hinton for Nobel Price in physics for foundational discoveries and inventions that enable machine learning with artificial neural networks. Despite its promise, backpropagation was computationally demanding, limiting its adoption that lead to the interest decline in neural networks during the second AI winter in the 1990s. With advancements in computing power, large datasets, and optimization techniques, backpropagation has since become the foundation of modern deep learning, enabling the training of today's most powerful AI models.

Convolutional Neural Networks (CNNs) originates is neuroscience, particularly the work of D. Hubel and T. Wiesel in the late 1950s [14]. Their experiments on the visual processing system of cats [15] revealed that neurons in the primary visual cortex respond selectively to specific visual patterns, such as edges, corners, and orientations. This insight laid the foundation for artificial neural architectures designed to imitate biological vision. The first fully trainable CNN was introduced by Yann LeCun in 1989 [16], demonstrating its effectiveness in handwritten digit recognition on the U.S. Modified National Institute of Standards and Technology (MNIST) database. Due to the computational limitations, CNNs remained largely unexplored until 2012, when AlexNet, developed by A. Krizhevsky, I. Sutskever, and G. Hinton, won the ImageNet competition [17], proving the immense potential of deep CNNs for large-scale image classification. Today, CNNs remain the gold standard for computer vision tasks, offering exceptional performance and ease of application. Unlike traditional methods that rely on handcrafted features, CNNs can automatically extract meaningful representations from images, making them the first choice for numerous real-world applications.

A pivotal theoretical breakthrough that preceded the development of the Single Hidden Layer Feedforward Network (SLFN) was the formulation of the *Universal Approximation Theorem*. This theorem provides a rigorous mathematical foundation for the expressive capabilities of neural networks in approximating real-valued functions. In a seminal result published in 1989, G. Cybenko [18] proved that a feedforward neural network with a single hidden layer employing a *sigmoidal activation function* is capable of approximating any continuous function $f : [0,1]^n \to \mathbb{R}$ to arbitrary precision, provided the network contains a sufficiently large number of hidden units.

Cybenko showed that for any continuous function f defined on the unit cube $[0,1]^n$ and for any $\varepsilon > 0$, there exists a finite set of weights $w_i \in \mathbb{R}^n$, biases $\theta_i \in \mathbb{R}$, and output coefficients $\alpha_i \in \mathbb{R}$, such that the function

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i \, \sigma(w_i^\top x + \theta_i)$$

approximates f uniformly on $[0,1]^n$ within an error ε, that is,

$$\sup_{x \in [0,1]^n} \left| f(x) - \hat{f}(x) \right| < \varepsilon.$$

Here, $\sigma : \mathbb{R} \to \mathbb{R}$ is any sigmoid function satisfying

$$\lim_{x \to -\infty} \sigma(x) = 0, \qquad \lim_{x \to \infty} \sigma(x) = 1.$$

This result established that SLFNs with a non-linear sigmoidal activation are *dense* in the space of continuous functions on compact subsets of $\mathbb{R}^n$, thus justifying their use

as universal function approximators. Shortly thereafter, in the same year, K. Hornik, along with M. Stinchcombe and H. White, extended and generalized Cybenko's theorem [19]. They showed that the universal approximation capability of feedforward networks is not limited to the sigmoid function. More precisely, any non-polynomial, bounded, and continuous activation function σ, under mild regularity assumptions, suffices for the network to retain the universal approximation property. These foundational results collectively demonstrated that even relatively shallow neural networks, those with only a single hidden layer, possess immense representational power. The significance of these theorems lies in their assertion of *existence*: they guarantee that, theoretically, a solution exists within the SLFN function space for any desired approximation accuracy, though they do not provide a constructive procedure to find the optimal weights and number of neurons in hidden layer. In particular, the universal approximation theorems make no assumptions about training algorithms, convergence behavior, or weight initialization strategies. Nonetheless, they laid the theoretical groundwork for the adoption of SLFNs in function approximation tasks and inspired a wealth of subsequent research into more efficient architectures and learning algorithms.

In 2004, G.-B. Huang introduced the Extreme Learning Machine (ELM), a fast and efficient alternative to traditional gradient-based MLPs [20]. Unlike MLPs, which were evolving into deeper and more complex architectures, ELM focused on a simpler SLFN. The key innovation of ELM was its training methodology: instead of iterative weight adjustments using backpropagation, it randomly initializes the input-to-hidden layer weights and biases, mapping input samples into a different feature space. Then, the hidden-to-output layer weights are computed in a single step using the Moore-Penrose pseudoinverse [21], providing the global optimal least-squares solution without iterative gradient-based optimization. This approach makes ELM significantly faster than traditional backpropagation-based networks, while still achieving comparable accuracy in many real-world applications [22]. Since it does not rely on gradient descent, ELM is inherently resistant to issues like vanishing or exploding gradients and local minima traps.

The concept of Extreme Learning Machines (ELMs) fundamentally revolves around dense single-layer feedforward neural networks (see Fig. 1.1), which offer robust solutions for both regression and classification tasks [23]. In a supervised setting, input data are represented as pairs $\eta = (x_i, t_i)_{i=1}^{N}$, where $X_{N\times d}$ denotes the input features for learning and $T_{N\times c}$ represents the corresponding targets. For classification, T consists of one-hot-encoded class labels, whereas for regression, it relies on one or more continuous target values. The architecture of an ELM typically involves input, single hidden, and output layers. The input layer of the network consists of d neurons, corresponding directly to the dimensionality of the input feature space. The hidden layer contains L neurons, where L is a hyperparameter that must be specified a priori. Despite its critical impact on model performance and generalization, there exists no universally efficient or theoretically grounded method for determining the optimal value of L, current practices rely primarily on heuristic strategies or empirical validation [24].. In classification tasks, *softmax* or *argmax* functions are frequently employed as activation functions of the output layer to identify the final class [25]. The uniform distribution function generates random weights α between the input and hidden layers, as well as biases b, resulting in a random mapping of the input data. The crux of ELM lies in the determination of weights β between the hidden and output layers. These weights are computed by solving the equation $Y = H\beta$, where H represents the output from the hidden layer, computed as $H = f(X\alpha + b)$, with $f(\cdot)$ denoting any activation function. Directly solving this system is infeasible due to the

irreversible nature of H [23]. Instead, β is estimated as the minimizer of the mean residual square error, given by the Moore-Penrose generalized inverse of H, denoted as $H^{\dagger}$, and rendering the final result as $\hat{\beta} = H^{\dagger}T$ [21]. The pseudoinverse matrix $H^{\dagger}$ uniquely determines $\hat{\beta}$ such that $H\hat{\beta}$ closely approximates Y in terms of mean square error [26]. ELMs offer a distinct advantage over traditional iterative learning methods by computing optimal weights in a single run, circumventing issues associated with vanishing or exploding gradients and suboptimal solutions [27, 28, 29]. This non-iterative approach ensures computational efficiency and mitigates common pitfalls encountered in iterative learning algorithms. Compared to methods such as the Multilayer Perceptron algorithm with Backpropagation, ELMs demonstrate remarkable speed and resource efficiency, with learning rates hundreds of times faster [30, 31]. This efficiency makes ELMs particularly well-suited for time-sensitive applications and large-scale datasets, offering a compelling alternative for rapid model training. Over time, several variants of ELM have been developed to enhance its capabilities. Notable extensions include Kernel-based ELM, which integrates kernel methods for improved non-linearity handling, and hybrid models that leverage evolutionary algorithms (e.g., Genetic Algorithms or Particle Swarm Optimization) to fine-tune parameters like weights or hyperparameters such as activation function parameters, further improving performance and adaptability across various tasks [32]. Following the formulation of the ELM, the process of modifying the original solution is initiated. The first idea involves incorporating the concept of ridge regression theory, which suggests adding a positive value to the diagonal of H^TH or HH^T to achieve a more stable solution and better generalization performance [33]. This concept is integrated into ELM through an additional hyperparameter C passed to the model. Subsequent efforts focus on optimizing the β weights after their calculation using Moore-Penrose pseudoinverse operations, minimizing the *l1*, *l2*, or combined *l12* loss with a given optimization method to form the Regularized ELM [34]. The *l1* loss function minimizes the sum of all absolute differences between the true value and the predicted value, whereas *l2* minimizes the sum of all squared differences between the true and predicted values. Still, the challenge of determining the number of neurons in the hidden layer L persists. This is addressed by the Kernel Extreme Learning Machine, which incorporates the evaluation of the matrix H^TH or HH^T by the kernel matrix, leveraging the Mercer kernel theorem [35]. Another approach involves tuning the randomly generated weights α using Metaheuristic Algorithms (MA), resulting in MA-ELM [36]. Specific ELM variants, such as constrained and weighted variants, are developed to handle imbalanced datasets by generating specially crafted α weights. In addition, receptive fields can change these weights to favor input in the center, as inspired by human vision in Receptive Fields ELM (RF-ELM) [37]. Moreover, researchers explore not only supervised but also semi-supervised and unsupervised methods, with the latter being ideal for data embedding and clustering tasks. To address the challenge of handling large-scale data, the Online Sequential ELM (OS-ELM) is introduced, allowing learning from data chunks with minimal memory consumption [38]. Furthermore, multilayer ELM structures are developed based on autoencoder ELM (AE-ELM), which maps input and target data into a different space defined by ELM parameters [39]. This enables the formulation of multilayer structures composed of stacked AE-ELMs and a typical ELM at the last layer.

One of the key applications of ML is image classification, where images or their fragments are analyzed and assigned to predefined categories based on the information they contain. This dissertation explores soil microorganism identification through microscopy imaging, comparing various ML approaches and evaluating different ELM optimization

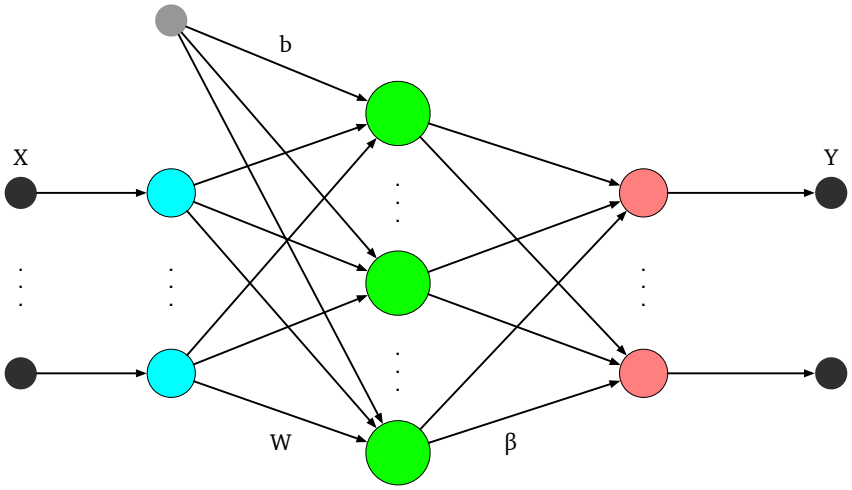


Figure 1.1: Extreme Learning Machine topology.

strategies for this task. Soil microorganisms play a crucial role in ecosystems, while some species are pathogenic or harmful, leading to significant agricultural losses, others contribute to plant health by suppressing pathogens and enhancing crop growth [40]. Traditional identification methods involve isolating microorganisms from soil, examining them under a microscope, and classifying them based on macro- and micromorphology, often supplemented by phenotypic and molecular biology techniques. This process is time-consuming, labor-intensive and depends often on expert knowledge. The goal is to streamline and automate microorganism identification using ML-based image analysis, relying solely on morphological traits that will facilitate the works of the other scientists enabling the large scale research in this scope.

1.1 Research problem

This dissertation has several key targets aimed at advancing the field of ELMs through optimization, performance evaluation, and application to real-world problems. These targets are as follows:

(a) **Open-Source Efficient Framework for ELM Calculation:** A primary target is the development of an open-source, efficient framework for ELM calculation utilizing *CUDA*, *TensorFlow*, and *scikit-learn*. The framework is designed to support various ELM variants, ensuring high efficiency and scalability. Written in Python, it enables seamless integration with already established machine learning libraries, taking full advantage of *CUDA* technology for optimized computations. The goal is to create a framework that is intuitive, user-friendly, and compatible with existing tools and frameworks, while remaining versatile and scalable for future use. Such approach supports researchers to implement ELM models easily, promoting transparency, reproducibility, and accessibility within the scientific community. Additionally, this open-source framework contributes to the open science movement by ensuring that the corresponding implementations and data sources are publicly available.

(b) **Performance of ELM on Apple M-Series Processors:** Another key thesis target is to investigate the performance of ELM on Apple's M-Series processors. With the introduction of the M1 chip, Apple has produced highly integrated processors that consume significantly less power than traditional PC-class devices, which makes them an

attractive option for machine learning tasks. This dissertation investigates whether these processors can effectively handle ELM models, particularly in the context of machine learning tasks that demand high computational resources. The performed evaluation accounts for trade-off between power consumption and performance, especially in light of the current global energy challenges.

(c) **Optimization of ELM:** The third thesis target is to optimize the performance of ELM through various strategies. One crucial area here is the selection of activation functions. The dissertation aims to address the open research question of determining the most suitable activation function for the specific tasks. Additionally, metaheuristic algorithms (MAs) will be explored to optimize ELM parameters, especially the activation function, as traditional approaches often rely on heuristic choices. Another focus is to refine the random weights generation between the input and hidden layers, investigating whether these weights can be fine-tuned using MAs or other optimization techniques. The dissertation also addresses the impact of different random distributions and generator implementations on the stability and performance of the model, ultimately seeking to improve the computational efficiency, stability, and predictive accuracy of ELM.

(d) **Applications of ELM in the Identification of Microorganisms:** A key application of the optimized ELM model aims on identification of soil microorganisms, which play a significant role in effective agriculture. Traditional methods of microorganism identification are labor-intensive and rely on manual observation of macro- and micromorphological traits. The target of this dissertation is to automate such identification process by developing an efficient machine learning model to accurately identify soil microorganism genera based solely on morphological traits observed under the microscope. The dissertation in question explores whether such a model is applicable for use across various datasets, including those generated through automated image capture. The successful development of such a system will enable rapid and precise identification of microorganisms, ultimately facilitating timely agricultural interventions to mitigate the harmful effects of pathogenic microorganisms on crops and laboratory colonies.

1.2 Overview of state of the art

Before presenting the results of the thesis in details, a short review of the current state of research related to the dissertation's objectives is provided. This section summarizes recent developments, identifies existing gaps, and highlights areas where further work is still needed.

1.2.1 Open-sourced efficient framework for ELM calculation utilizing *CUDA*, *TensorFlow* and *scikit-learn*

There are notable deficiencies in existing ELM implementations, particularly regarding the lack of a comprehensive framework that integrates diverse ELM variants while maintaining compatibility with modern ML tools like *TensorFlow* and *scikit-learn*. Currently, there is a scarcity of available ELM implementations, with only four variants provided in Python, none of which conform to modern *TensorFlow* 2 standards or to integrate seamlessly with *scikit-learn*.

Many existing implementations are outdated or incompatible with current *TensorFlow* versions. For example, a *TensorFlow*-based implementation from 2018 by Cornell covers

only the fundamental ELM concept and has not been updated since its release, making it difficult to run without executing several preparatory commands to adapt the code to modern *TensorFlow* distributions [41]. Another example is the Online Sequential Extreme Learning Machine (OS-ELM) from 2018 by Otenim, which relies on the numpy module for calculations, limiting its ability to leverage GPU acceleration by *CUDA* [42].

Additionally, the core ELM implementation by Putra [43] utilizes pure Python for computations, further restricting its performance and scalability. These limitations highlight the need for an open-sourced, efficient ELM framework that supports *CUDA* acceleration and ensures compatibility with *TensorFlow* 2 and *scikit-learn*, ultimately enhancing performance and usability for modern ML workflows.

1.2.2 Performance of ELM on Apple M-Series Processors

The M-Series Processors by Apple have piqued the interest of researchers due to their potential applications in scientific endeavors, particularly in optimization and ML calculations [44]. The main objective here is to evaluate different fundamental optimization and ML algorithms across various datasets. A significant gap in the current literature has been identified, where performance evaluations are often conducted on a single dataset [45] or focus solely on a single ML method, as seen in the work by Kasperek et al. [46]. This Ph.D. thesis contribution expands the research from [46] extending this work to *CUDA*-enabled devices. In the research process of this thesis, a *CUDA*-compatible RTX 3090 GPU was utilized. Previous experiments comparing the NVIDIA V100 and A100 GPUs with Apple's M1 and M1 Ultra showed promising results, with Apple Silicon outperforming both GPUs in several tasks [45]. While these GPUs delivered strong performance, they do not represent the most powerful GPUs currently available, with the NVIDIA V100 offering 14.13 TFLOPS in Float32 precision compared to the RTX 3090's 35.58, highlighting the importance of testing on more advanced hardware.

1.2.3 ELM Optimization

Huang et al. demonstrated that, unlike traditional gradient-based learning algorithms, which require differentiable activation functions, ELM can employ non-differentiable or piecewise differentiable activation functions [47]. In recent years, numerous new activation functions have been proposed (especially within deep learning scope), showing promising results in ML applications. The selection of an activation function depends on the specific input task and researchers continue to explore novel functions that could enhance ELM performance across diverse class of problems [48]. Despite the development of new activation functions, literature reviews indicate that sigmoid and hyperbolic tangent functions remain the most commonly used in practical ELM applications [49].

A notable gap in the field is the lack of comprehensive comparisons between activation functions across multiple datasets. For example, Ratnawati et al. [50] compared 11 activation functions but limited evaluation to a single dataset, raising concerns about the generalizability of their findings. The study hypothesizes that the optimal choice of activation function is highly dependent on dataset characteristics, suggesting that subsets of functions may consistently perform better or worse on specific types of data that also conforms to the fact that ELM random weights are representing the random mapping into the higher dimensional space. This operation is strictly related to the data characteristics itself and how the decision boundary is generated using a certain activation function.

The typical random initialization of weights in ELM can be further optimized through fine-tuning. Recent research has introduced hybrid Metaheuristic Algorithm-ELM (MA-ELM) models that combine ELM with various metaheuristic algorithms to improve performance Chia et al. [51] employed Particle Swarm Optimization, Moth-Flame Optimization, and Whale Optimization Algorithm to enhance ELM performance. Similarly, Wu et al. [52] utilized Genetic Algorithms, Ant Colony Optimization, Cuckoo Search Algorithm, and Flower Pollination Algorithm. These studies highlight the superiority of hybrid MA-ELM models over standard ELM approaches. However, most of this research focuses on practical applications, with a notable lack of comprehensive comparative analyses of metaheuristic algorithms within the ELM framework.

Another field of ELM optimization involves tuning the parameters of activation functions. While previous studies have explored this topic, the use of MAs to optimize activation function parameters remains unexplored. Traditionally, MA algorithms have been applied to fine-tune network weights, but my study proposes focusing on activation function parameters optimization. This approach offers potential advantages, such as improved generalization and reduced computational overhead, as it involves optimizing a limited set of parameters-typically between one and five-rather than the extensive search required for tuning numerous weights in traditional MA-ELM methods.

Additionally, the influence of random number generators and distribution functions on ELM performance has received limited attention in the literature. Despite the critical role of random number generators in initializing weights and biases between the input and hidden layers, there is a notable absence of studies evaluating their impact on ELM outcomes. R. Wang's 2012 study examined the influence of randomly assigned weights in ELM compared to kernel methods used in Kernel Extreme Learning Machine (KELM) [53]. The findings indicated that random initialization, which expands the feature space, often outperforms kernel mappings for various classification and regression tasks. In 2016, G. Dudek investigated how the type of activation function and the range of random weights and biases affect ELM's approximation capabilities [54]. Dudek's experiments demonstrated that ELMs perform better in function approximation tasks when input weights are randomly selected within a narrow range.

1.2.4 Applications of ELM and other Machine Learning methods in the identification of microorganisms

Extensive studies have explored the application of ML for detecting pathogenic fungi based on leaf images, achieving remarkable accuracy levels. CNN models such as AlexNet, GoogleNet, InceptionV3, and ResNets have reached AUC scores close to 99% in detecting fungal pathogens through leaf images [55, 56]. These studies highlight the potential of ML to accurately identify plant diseases. On the other hand, such approaches typically identify threats only after symptoms have manifested, leading to significant delays that can result in substantial agricultural losses, especially with diseases like *Verticillium wilt* [57]. Conventional methods for identifying microorganisms in soil samples, including phenotypic and molecular techniques, are often time-consuming and costly [58]. An alternative is to utilize microscopic imaging for the identification of microorganism genera. While soil fungi identification remains a niche area of study, initial insights can be drawn from research focusing on human-pathogenic fungi.

Zieliński et al. addressed a similar challenge, demonstrating that microscopic images can accelerate the identification process [59]. Their study showed that combining CNNs

with bag-of-words techniques significantly reduces the time and cost of fungal species identification by eliminating the need for extensive biochemical tests. Nevertheless, their research was limited by a small dataset of 180 images, all captured under uniform lighting conditions potentially introducing biases due to expert involvement.

Recent studies increasingly highlight the transformative potential of ML, particularly deep learning, in microbiology. Research by Treebupachatsakul [60] and Khasim [61] demonstrates the use of deep learning for microorganism recognition and classification, with Khasim specifically emphasizing the effectiveness of CNNs. Qu [62] and Jiang [63] provide comprehensive overviews of ML applications in microbiology, underscoring its utility in classification tasks and its potential to enhance the organization and application of microbiological knowledge. The popularity of CNNs in this domain is attributed to their straightforward processing pipeline, which eliminates the need for complex feature crafting and selection. Despite this, traditional ML methods involving feature engineering, such as Support Vector Machines, Random Forests and k-Nearest Neighbors [64], remain prevalent in microbial studies. Kotwal [65], for example, discusses their application in bacterial classification, demonstrating their continued relevance.

In a comprehensive review covering 100 studies on ML applications in microbial recognition from 1995 to 2021, Rani [66] notes that only 12.1% of these studies concentrate on fungi. One such focused study is by Liu et al. [67], which employs ML techniques to detect and count fungal microorganisms in microscopic images. Similarly, Tahir et al. [68] utilized SVMs for fungal spore detection, employing image patches, Gaussian filtering, and handcrafted features to achieve an accuracy of 88%.

Microscopic images of soil-dwelling microorganisms pose significant challenges for standard image processing methods [69]. These microorganisms display a wide range of complex, non-rigid, and irregular shapes and textures, complicating segmentation and analysis [70, 71]. Unlike bacteria, which exhibit more homogeneous structures [72], soil fungi often feature diverse and irregular components such as hyphae, phialides, micro- and macroconidia, conidia cells and zoospores.

1.2.5 Conclusions from literature review

The review of existing literature points out to several critical gaps and opportunities.

- Current ELM implementations lack compatibility with modern frameworks like *TensorFlow* 2 and *scikit-learn*, and do not effectively leverage GPU acceleration. Similarly, performance evaluations of optimization and ML algorithms on Apple M-Series processors remain limited, particularly in terms of hardware diversity and algorithmic variety.

- In the field of ELM optimization, although numerous metaheuristic-enhanced approaches have been proposed, comprehensive comparative studies are still missing, especially regarding activation function parameter tuning and the influence of random initialization strategies.

- While ML applications in microorganism identification have shown promising results, research on applying ELM methods to microscopic image analysis-particularly for soil fungi remains scarce.

These limitations underscore the need for the development of a modern and efficient ELM framework, accompanied by its deployment in novel application domains and supported

by comprehensive studies on performance evaluation and architectural optimization.

1.3 Thesis main contributions

The thesis research outcomes addressing the identified literature gaps and research objectives and achievements are presented in the following subsections.

1.3.1 Open-sourced efficient framework for ELM calculation utilizing *CUDA*, *TensorFlow* and *scikit-learn*

The development of *TfELM* [73] addresses a significant gap in the field of ELMs, offering a comprehensive and efficient solution for both researchers and practitioners. Among the 18 ELM variants implemented in *TfELM*, 14 were previously unavailable in Python, and 16 had not been integrated within the *TensorFlow* framework. By ensuring seamless compatibility with *TensorFlow* 2.15 and *scikit-learn*, *TfELM* aligns with contemporary software development standards, enhancing usability and accessibility. A key contribution of *TfELM* is its fully modular, object-oriented design that adheres to *scikit-learn* principles, making it approachable for both novice and experienced ML practitioners. This modular architecture not only simplifies usage but also enables researchers to experiment with novel ELM configurations, thereby broadening the potential applications of ELM in ML. To optimize computational efficiency without compromising flexibility, extensive preliminary experiments were conducted on five standard benchmark datasets. These evaluations covered both GPU-accelerated and CPU-based implementations. Performance assessments across diverse hardware configurations consistently demonstrated that *TfELM* outperforms existing ELM implementations, achieving substantial reductions in execution time. Detailed implementation insights and complete source code are provided in publication, supporting reproducibility and further development. *TfELM*'s contributions extend beyond Informatics, offering valuable tools for various disciplines that utilize ML thereby promoting interdisciplinary research and practical applications.

Paper related to this part of dissertation refers to [73].

1.3.2 Performance of ELM on Apple M-Series Processors

To comprehensively evaluate the performance of the selected ML classifiers across diverse hardware platforms and data types, six benchmark datasets from UCI [74] were utilized. They vary in the number of samples, features, and classes, ranging from particularly small to moderately sized datasets. The objective here is to measure the execution time of each classifier on three distinct hardware platforms: Apple's M1 with 8GB RAM, M2 with 16GB RAM, a high-performance NVIDIA RTX 3090 GPU with 24GB memory, and a mid-range laptop featuring an Intel Core i5 11500H processor paired with an NVIDIA RTX 3050 Ti graphics card. The experiment also aimed to compare the performance of mobile devices (M1/M2) against an i5-powered laptop. Unexpectedly, the unplugged i5 exhibited an average execution time four times longer than when plugged in, while the M1/M2 devices maintained consistent computational performance regardless of battery status. The comprehensive evaluation extends beyond previously explored GPUs, offering valuable insights into the real-world performance of ML classifiers across a range of hardware configurations. A variety of ML methods were evaluated, including ELM, k-Nearest

Neighbors, Multi-Layer Perceptron, Random Forest, and Support Vector Machine. Interestingly, the research outcomes clarify that applicability of M-Series Processors is the most prominent for the small or moderate datasets, where the SoC provides clear advancements in terms of limited data swapping operations between RAM and VRAM as it occurs in PC-class devices.

The research presented in this section is closely related to the study published in [75].

1.3.3 ELM optimization

To address the gaps identified in the literature concerning the application of various activation functions in ELM and their impact on performance, this thesis study presents a comprehensive evaluation of 36 distinct activation functions across 10 diverse datasets. The primary objective is to determine whether specific activation functions consistently outperform others and to assess how the optimal choice may vary based on the dataset's characteristics. Notably, this work incorporates a wide range of activation functions, including several that have not been previously explored in the context of ELM, such as Mish, introduced in 2019 [76]. These novel activation functions may offer superior generalization capabilities compared to traditionally options used, positioning them as adequate candidates for enhancing ELM performance in classification tasks.

As part of this dissertation, an in-depth evaluation of hybrid ELM combined with MA was conducted, focusing on the influence of MA selection and parameter tuning on model performance. The study investigated the performance of MA-ELM on two benchmark datasets: MNIST Handwritten Digits and Wine Quality White [77]. Multiple MAs were applied, and their effects on both classification accuracy and computational time were systematically analyzed. A core objective of the research was to investigate different parameter configurations impact on MA-ELM performance. Specifically, the study explored the influence of key parameters, including the number of neurons in the hidden layer of the ELM, the population size of the MA, and the stopping conditions used during optimization. The results revealed that the hybrid MA-ELM could achieve higher accuracy even with a reduced number of neurons in the hidden layer compared to traditional ELM, leading to significantly faster prediction times. One notable finding was the effect of termination criteria on MA performance. For the MNIST dataset, the most efficient results were obtained with a hard stop at just five iterations, whereas the Wine Quality White dataset benefited from a longer optimization process, with the best outcomes observed at fifty iterations. These insights highlight the importance of dataset-specific parameter tuning in hybrid MA-ELM systems and contribute valuable guidelines for future applications of metaheuristic optimization in ELM frameworks.

To validate the research question of whether MA can be used to optimize activation function parameters instead of fine-tuning the weights, this dissertation includes an extensive study evaluating the performance of 24 distinct activation functions across five diverse datasets. The selected benchmark datasets: Ionosphere, Breast Cancer, Australian Credit, Musk, and Banana are widely used for ML performance evaluations [74]. These datasets were intentionally chosen to represent a broad range of characteristics and complexities. The study aims to evaluate activation function performance in terms of accuracy and execution time while investigating whether integrating MA enhances overall results. A rigorous experimental setup was employed, utilizing a 50-times repeated 10-fold cross-validation to ensure reliable and robust outcomes. Mean accuracy across multiple runs served as the primary evaluation metric, providing an objective measure of model

performance. Significant variations in accuracy based on the chosen activation functions and their parameters were observed. The findings show that the MA-ELM approach is beneficial, especially when it achieves better or comparable results with fewer neurons in the hidden layer, resulting in more efficient models.

The experiments conducted within this study examined the impact of random number generators and distribution function choices on the performance and stability (obtaining comparable results every time the algorithm is run) of ELMs. A comprehensive evaluation was carried out, resulting in a total of 28,600 experimental setups that varied across datasets, distribution functions, random number generators and the number of neurons in the hidden layer. The study utilized five widely recognized datasets from the UCI repository [74], incorporating 13 different distribution functions, 22 random number generators from Python's randomgen package, and hidden layer sizes ranging from 50 to 10,000 neurons. Traditionally, ELMs employ uniform distribution functions for random initialization, but the findings of this research highlight significant performance variations across datasets when alternative distributions are used. The choice of randomness source in ELMs was found to substantially influence final accuracy, challenging established norms. Remarkably, certain distribution functions consistently outperformed others on specific datasets, indicating the necessity of considering dataset-specific characteristics when selecting randomness sources. The Pareto distribution, for example, demonstrated superior performance, particularly on the Australian and Banana datasets. These results underscore the need for a more nuanced approach to randomness in ELMs, promoting enhanced performance and stability tailored to the dataset in question.

The publications of the PhD Candidate referring to this part of the dissertation include [78, 79, 80, 81].

1.3.4 Applications of ELM in the identification of microorganisms

The study forming this dissertation core also addresses the need for rapid and cost-effective identification of soil microorganisms while establishing a robust testing framework. The proposed model was validated across multiple datasets, each prepared under various conditions, including scenarios where datasets were generated fully automatically, with the microscope capturing images sequentially and non-overlappingly to cover the sample area. The research utilized five distinct datasets specifically curated for this study. The initial dataset, comprising 128 images, was meticulously prepared by an expert microbiologist and served as a basis for fine-tuning the image processing pipeline. This foundational set was expanded into a second dataset with 303 images. Subsequently, three additional datasets are created using automated image acquisition techniques, collectively contributing to a total of 2,866 images. To ensure the findings are both robust and interpretable, the study integrated Explainable AI (XAI) techniques. This approach not only evaluated the model's performance but also elucidated the rationale behind the classification of individual instances and overall test sets. A key aspect of the research was the comparison between traditional handcrafted feature approaches and CNNs. Additionally, the research aimed to advance practical applications where prediction time is critical, especially during large-scale tests. The study employed the ELM known for its rapid training and exceptionally fast prediction times that suited for real-world applications where quick predictions are essential [82].

The following papers of the Candidate contribute to this section topic: [83, 84, 85].

Bibliography

[1] E. Z. Naeini and K. Prindle, "Machine learning and learning from machines," *The Leading Edge*, vol. 37, no. 12, p. 886–893, 2018.

[2] V. S. Subrahmanian, *Logic, Machine Learning, and Security*, p. 3–6. Springer International Publishing, 2019.

[3] D. Bacciu, P. J. Lisboa, A. Sperduti, and T. Villmann, *Springer Handbook of Computational Intelligence*. Springer Berlin Heidelberg, 2015.

[4] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, "The "Wake-Sleep" Algorithm for Unsupervised Neural Networks," *Science*, vol. 268, no. 5214, pp. 1158–1161, 1995.

[5] S. Jarius and B. Wildemann, "Pavlov's reflex before Pavlov: early accounts from the English, French and German classic literature," *European Neurology*, vol. 77, no. 5–6, p. 322–326, 2017.

[6] P. Mancosu, R. Zach, and C. Badesa, *The Development of Mathematical Logic from Russell to Tarski, 1900–1935*, p. 318–470. Oxford University Press, 2009.

[7] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[8] E. Kuriscak, P. Marsalek, J. Stroffek, and P. G. Toth, "Biological context of Hebb learning in artificial neural networks, a review," *Neurocomputing*, vol. 152, p. 27–35, 2015.

[9] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[10] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "Deep learning's diminishing returns: The cost of improvement is becoming unsustainable," *IEEE Spectrum*, vol. 58, no. 10, p. 50–55, 2021.

[11] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[12] S. Amari, "A theory of adaptive pattern classifiers," *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 3, pp. 299–307, 1967.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[14] G. W. Lindsay, "Convolutional neural networks as a model of the visual system: Past, present, and future," *Journal of Cognitive Neuroscience*, vol. 33, no. 10, p. 2017–2031, 2021.

[15] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurons in the cat's striate cortex," *Journal of Physiology*, vol. 148, pp. 574–591, 1959.

[16] J. S. D. Yann LeCun, Bernhard Boser *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[17] S. I. Krizhevsky A. and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1097–1105, 2012.

[18] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, p. 303–314, 1989.

[19] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, p. 359–366, 1989.

[20] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machine: a new learning scheme of Feedforward Neural Networks," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, vol. 2, pp. 985–990, IEEE, 2004.

[21] C. Rao and S. Mitra, *Generalized Inverse of Matrices and Its Applications*. Hoboken: John Wiley & Sons, 1971.

[22] J. Wang, S. Lu, S.-H. Wang, and Y.-D. Zhang, "A review on Extreme Learning Machine," *Multimedia Tools and Applications*, vol. 81, no. 29, pp. 41611–41660, 2022.

[23] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machine: A new learning scheme of feedforward neural networks," in *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks (IJCNN)*, vol. 2, pp. 985–990, 2004.

[24] A. L. Freire and G. D. Barreto, "A new model selection approach for the elm network using metaheuristic optimization," in *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, 2014.

[25] J. S. Denker and Y. LeCun, "Transforming neural-net output levels to probability distributions," in *Advances in Neural Information Processing Systems (NeurIPS)*, 1990.

[26] K. Struniawski, R. Kozera, and A. Konopka, "Performance of selected nature-inspired metaheuristic algorithms used for Extreme Learning Machine," in *Computational Science – ICCS 2023*, vol. 10475, (Cham), pp. 494–503, Springer, 2023.

[27] G. MacDonald, A. Godbout, B. Gillcash, and S. Cairns, "Volume-preserving neural networks: a solution to the vanishing gradient problem." arXiv preprint arXiv:1911.09576, 2019.

[28] J. Zhang, Q. Lei, and I. S. Dhillon, "Stabilizing gradients for deep neural networks via efficient SVD parameterization." arXiv preprint arXiv:1803.09327, 2018.

[29] J. Zhang, Y. Li, W. Xiao, and Z. Zhang, "Non-iterative and fast deep learning: Multilayer Extreme Learning Machines," *Journal of the Franklin Institute*, vol. 357, no. 13, pp. 8925–8955, 2020.

[30] J. Tang, C. Deng, and G.-B. Huang, "Extreme Learning Machine for multilayer perceptron," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 4, pp. 809–821, 2016.

[31] Y. Yang and Q. J. Wu, "Extreme Learning Machine with subnetwork hidden nodes for regression and classification," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2885–2898, 2016.

[32] A. Ashkzari and A. Azizi, "Introducing genetic algorithm as an intelligent optimization technique," *Applied Mechanics and Materials*, vol. 568–570, p. 793–797, 2014.

[33] A. E. Hoerl and R. W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[34] W. Deng, Q. Zheng, and L. Chen, "Regularized Extreme Learning Machine," in *2009 IEEE Symposium on Computational Intelligence and Data Mining*, (Nashville, TN, USA), pp. 389–395, 2009.

[35] M. Alizamir, S. Kim, M. Zounemat-Kermani, S. Heddam, N. W. Kim, and V. P. Singh, "Kernel Extreme Learning Machine: an efficient model for estimating daily dew point temperature using weather data," *Water*, vol. 12, no. 9, p. 2600, 2020.

[36] L. Wu, G.-B. Huang, J. Fan, X. Ma, H. Zhou, and W. Zeng, "Hybrid Extreme Learning Machine with meta-heuristic algorithms for monthly pan evaporation prediction," *Computers and Electronics in Agriculture*, vol. 168, p. 105115, 2020.

[37] G.-B. Huang, Z. Bai, L. L. C. Kasun, and C. M. Vong, "Local Receptive Fields based Extreme Learning Machine," *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, p. 18–29, 2015.

[38] J.-M. Park and J.-H. Kim, "Online Recurrent Extreme Learning Machine and its application to time-series prediction," in *2017 International Joint Conference on Neural Networks (IJCNN)*, p. 1983–1990, IEEE, 2017.

[39] K. Sun, J. Zhang, C. Zhang, and J. Hu, "Generalized Extreme Learning Machine Autoencoder and a new Deep Neural Network," *Neurocomputing*, vol. 230, p. 374–381, 2017.

[40] K. Nadarajah, "Role of microbiome in the generation of disease-suppressive soil for sustainable agriculture," *ScienceAsia*, vol. 50, no. 5, p. 1, 2024.

[41] S. Cornell, "Tf-ELM." `https://github.com/popcornell/tfelm`, 2018. Accessed: 2025-04-10.

[42] Otenim (GitHub's nickname), "TF-OS-ELM." Retrieved from `https://github.com/otenim/TensorFlow-OS-ELM`, 2018.

[43] V. Putra, "Simulasi paper Extreme Learning Machine: theory and applications." Retrieved from `https://github.com/virgantara/Extreme-Machine-Learning`, 2022.

[44] V. Dalakoti and D. Chakraborty, "Apple m1 chip vs intel (x86)," *EPRA International Journal of Research and Development (IJRD)*, vol. 7, no. 5, pp. 207–211, 2022.

[45] C. Kenyon and C. Capano, "Apple silicon performance in scientific computing," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–10, IEEE, 2022.

[46] D. Kasperek, M. Podpora, and A. Kawala-Sterniuk, "Comparison of the usability of apple m1 processors for various machine learning tasks," *Sensors*, vol. 22, no. 20, p. 8005, 2022.

[47] G.-B. Huang, Q.-Y. Zhu, K. Mao, C. Siew, P. Saratchandran, and N. Sundararajan, "Can threshold networks be trained directly?," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, pp. 187–191, 2006.

[48] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machines: a survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 10, pp. 2822–2840, 2019.

[49] G.-B. Huang, H. Zhou, X.-T. Ding, and R. Zhang, "Trends in Extreme Learning Machines: a review," *Neural Networks*, vol. 61, pp. 32–48, 2015.

[50] D. E. Ratnawati, W. Marjono, and S. Anam, "Comparison of activation function on Extreme Learning Machine (ELM) performance for classifying the active compound," in *Symposium on Biomathematics 2019 (Symomath 2019)*, vol. 2264, p. 140001, AIP Publishing, 2020.

[51] M. Y. Chia, Y. F. Huang, and C. H. Koo, "Swarm-based optimization as stochastic training strategy for estimation of reference evapotranspiration using Extreme Learning Machine," *Agricultural Water Management*, vol. 243, p. 106447, 2021.

[52] L. Wu, H. Zhou, X. Ma, J. Fan, and F. Zhang, "Daily reference evapotranspiration prediction based on hybridized Extreme Learning Machine model with bio-inspired optimization algorithms: Application in contrasting climates of China," *Journal of Hydrology*, vol. 577, p. 123960, 2019.

[53] R. Wang, S. Kwong, and X. Wang, "A study on random weights between input and hidden layers in Extreme Learning Machine," *Soft Computing*, vol. 16, no. 9, pp. 1465–1475, 2012.

[54] G. Dudek, "Extreme Learning Machine as a function approximator: initialization of input weights and biases," in *Advances in Intelligent Systems and Computing*, pp. 59–69, Springer International Publishing, 2016.

[55] V. Maeda-Gutiérrez, C. E. Galván-Tejada, L. A. Zanella-Calzada, *et al.*, "Comparison of convolutional neural network architectures for classification of tomato plant diseases," *Applied Sciences*, vol. 10, no. 4, p. 1245, 2020.

[56] M. R. Howlader, U. Habiba, R. H. Faisal, and M. M. Rahman, "Automatic recognition of guava leaf diseases using deep convolution neural network," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pp. 1–5, IEEE, 2019.

[57] A. G. Ayele, T. A. Wheeler, and J. K. Dever, "Impacts of verticillium wilt on photosynthesis rate, lint production, and fiber quality of greenhouse-grown cotton (gossypium hirsutum)," *Plants*, vol. 9, no. 7, p. 857, 2020.

[58] Z. Yunus, "Overview of techniques for the detection and identification of microorganisms," *Defence S and T Technical Bulletin*, vol. 1, no. 1, p. 46 – 59, 2008.

[59] B. Zieliński, A. Sroka-Oleksiak, D. Rymarczyk, A. Piekarczyk, and M. Brzychczy-Włoch, "Deep learning approach to describe and classify fungi microscopic images," *PLOS ONE*, vol. 15, no. 6, p. e0234806, 2020.

[60] T. Treebupachatsakul and S. Poomrittigul, "Microorganism image recognition based on deep learning application," in *2020 International Conference on Electronics, Information, and Communication (ICEIC)*, IEEE, 2020.

[61] S. Khasim, H. Ghosh, I. S. Rahat, K. Shaik, and M. Yesubabu, "Deciphering microorganisms through intelligent image recognition: machine learning and deep learning approaches, challenges, and advancements," *EAI Endorsed Transactions on Internet of Things*, vol. 10, 2023.

[62] K. Qu, F. Guo, X. Liu, Y. Lin, and Q. Zou, "Application of machine learning in microbiology," *Frontiers in Microbiology*, vol. 10, 2019.

[63] Y. Jiang, J. Luo, D. Huang, Y. Liu, and D.-d. Li, "Machine learning advances in microbiology: a review of methods and applications," *Frontiers in Microbiology*, vol. 13, 2022.

[64] P. Thanh Noi and M. Kappas, "Comparison of Random Forest, k-Nearest Neighbor, and Support Vector Machine classifiers for land cover classification using Sentinel-2 imagery," *Sensors*, vol. 18, no. 1, p. 18, 2017.

[65] S. Kotwal, P. Rani, T. Arif, J. Manhas, and S. Sharma, "Automated bacterial classifications using machine learning based computational techniques: architectures, challenges and open research issues," *Archives of Computational Methods in Engineering*, vol. 29, no. 4, pp. 2469–2490, 2021.

[66] P. Rani, S. Kotwal, J. Manhas, V. Sharma, and S. Sharma, "Machine learning and deep learning based computational approaches in automatic microorganisms image recognition: methodologies, challenges, and developments," *Archives of Computational Methods in Engineering*, vol. 29, no. 3, pp. 1801–1837, 2021.

[67] L. Liu, Y. Yuan, J. Zhang, H. Lei, Q. Wang, J. Liu, X. Du, G. Ni, and Y. Liu, "Automatic identification of fungi under complex microscopic fecal images," *Journal of Biomedical Optics*, vol. 20, no. 7, p. 076004, 2015.

[68] M. W. Tahir, N. A. Zaidi, R. Blank, P. P. Vinayaka, M. J. Vellekoop, and W. Lang, "Fungus detection through optical sensor system using two different kinds of feature vectors for the classification," *IEEE Sensors Journal*, vol. 17, no. 16, pp. 5341–5349, 2017.

[69] R. Gonzalez and R. Woods, *Digital Image Processing.* London: Pearson, 4th ed., 2018.

[70] Z. Feng, "A local invariant feature extraction and description method for microscopic image of bacteria," *Acta Microscopica*, vol. 29, pp. 1963–1970, 2020.

[71] C. Raghavendra, K. S. S. Reddy, M. Shanmugathai, and A. Devipriya, "Electron microscopy images for automatic bacterial trichomoniasis diagnostic classification separating and sorting of overlapping microbes," in *11th Annual International Conference (Aic) 2021: On Sciences and Engineering*, vol. 2613, p. 020089, AIP Publishing, 2023.

[72] A. Konopka, R. Kozera, L. Sas-Paszt, P. Trzcinski, and A. Lisek, "Identification of the selected soil bacteria genera based on their geometric and dispersion features," *PLOS ONE*, vol. 18, no. 10, p. e0293362, 2023.

[73] K. Struniawski and R. Kozera, "TfELM: Extreme Learning Machines framework with Python and TensorFlow," *SoftwareX*, vol. 27, p. 101833, 2024.

[74] D. Dheeru and E. Karra Taniskidou, "Uci machine learning repository." Available at `http://archive.ics.uci.edu/ml`, 2017.

[75] K. Struniawski, A. Konopka, and R. Kozera, *Exploring Apple Silicon's Potential from Simulation and Optimization Perspective*, p. 35–42. Springer Nature Switzerland, 2024.

[76] D. Misra, "Mish: A self regularized non-monotonic activation function," *arXiv*, 2019.

[77] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009.

[78] K. Struniawski, R. Kozera, and A. Konopka, "Performance of selected nature-inspired metaheuristic algorithms used for extreme learning machine," in *Computational Science – ICCS 2023* (J. Mikyška, C. de Mulatier, M. Paszynski, V. V. Krzhizhanovskaya, J. J. Dongarra, and P. M. A. Sloot, eds.), vol. 10475 of *Lecture Notes in Artificial Intelligence*, pp. 498–512, Springer, 2023.

[79] K. Struniawski, A. Konopka, and R. Kozera, "Performance evaluation of activation functions in extreme learning machine," in *ESANN 2023: Proceedings* (M. Verleysen, ed.), pp. 351–356, Ciaco, 2023.

[80] K. Struniawski, A. Konopka, and R. Kozera, "Metaheuristic algorithms in extreme learning machine for selection of parameters in activation function," in *Modelling and Simulation'2023. The 2023 European Simulation and Modelling Conference* (R. Vingerhoeds and P. D. Saqui-Sannes, eds.), pp. 239–244, EUROSIS-ETI, 2023.

[81] K. Struniawski, A. Konopka, and R. Kozera, "Credibility of randomness in extreme learning machine," in *Trust and Artificial Intelligence: Development and Application of AI Technology* (J. Paliszkiewicz and J. Gołuchowski, eds.), pp. 92–106, Routledge, 2025.

[82] Z.-H. Zhu, Q. Hong, L.-f. Wu, Q. Wang, and M. Ma, "Early identification of male and female embryos based on UV/Vis transmission spectroscopy and Extreme Learning Machine," *Spectroscopy and Spectral Analysis*, vol. 39, pp. 2780–2787, 2019.

[83] K. Struniawski, A. Konopka, and R. Kozera, "Identification of soil bacteria with machine learning and image processing techniques applying single cells' region isolation," in *Modelling and Simulation 2022. The European Simulation and Modelling Conference 2022* (I. Praca, E. Maia, and P. Geril, eds.), pp. 76–81, EUROSIS-ETI, 2022.

[84] K. Struniawski, R. Kozera, P. Trzciński, A. Lisek, and L. Sas-Paszt, "Automated identification of soil fungi and chromista through convolutional neural networks," *Engineering Applications of Artificial Intelligence*, vol. 127B, pp. 1–12, 2024.

[85] K. Struniawski, R. Kozera, P. Trzciński, A. Marasek-Ciołakowska, and L. Sas-Paszt, "Extreme learning machine for identifying soil-dwelling microorganisms cultivated on agar media," *Scientific Reports*, vol. 14, pp. 1–23, 2024.

Chapter 2

Overview of dissertation and publications

This dissertation is based on original research conducted by the Candidate, who served as the first author and primary contributor in all related publications. The specific contributions to each work are given below.

2.1 Publications constituting to the dissertation

1. *Identification of soil bacteria with machine learning and image processing techniques applying single cells' region isolation*
   Karol Struniawski, Aleksandra Konopka, and Ryszard Kozera.
   In: I. Praca, E. Maia, P. Geril (Eds.), *Modelling and Simulation 2022. The European Simulation and Modelling Conference 2022*, Porto (Portugal), EUROSIS-ETI, 2022, pp. 76–81. ISBN 9789492859242.
   Contribution: 60% — Concept development, method implementation, computational experiments, first manuscript draft, and conference presentation.
   Conference paper with presentation, 70 pkt. MNiSW (Polish Ministry of Science and Higher Education Points)

2. *Performance of selected nature-inspired metaheuristic algorithms used for Extreme Learning Machine*
   Karol Struniawski, Ryszard Kozera, and Aleksandra Konopka.
   In: J. Mikyška, C. de Mulatier, M. Paszynski, V. V. Krzhizhanovskaya, J. J. Dongarra, P. M. A. Sloot (Eds.), *Computational Science – ICCS 2023*, Prague (Czechia), Lecture Notes in Artificial Intelligence, vol. 10475, Springer, 2023, pp. 498–512. ISBN 978-3-031-36023-7. DOI: 10.1007/978-3-031-36024-4_38.
   Contribution: 70% — Concept development, method implementation, computational experiments, initial manuscript draft, and conference presentation.
   Conference paper with presentation, 140 pkt. MNiSW

3. *Performance evaluation of activation functions in Extreme Learning Machine*
   Karol Struniawski, Aleksandra Konopka, and Ryszard Kozera.
   In: M. Verleysen (Ed.), *ESANN 2023: Proceedings*, Brugge (Belgium),Ciaco, 2023, pp. 351–356. ISBN 9782875870872. DOI: 10.14428/esann/2023.es2023-31.
   Contribution: 70% — Study conceptualization, method implementation, computational experiments, first manuscript draft, and conference poster preparation.
   Conference paper with presentation, 70 pkt. MNiSW

4. *Metaheuristic algorithms in Extreme Learning Machine for selection of parameters in activation function*
   Karol Struniawski, Aleksandra Konopka, and Ryszard Kozera.
   In: R. Vingerhoeds, P. De Saqui-Sannes (Eds.), *Modelling and Simulation'2023. The 2023 European Simulation and Modelling Conference*, Toulouse (France), EUROSIS-ETI, 2023, pp. 239–244. ISBN 9789492859280.
   Contribution: 70% — Study conception, implementation, computational experiments, first manuscript draft, and conference presentation.
   Conference paper with presentation, 70 pkt. MNiSW

5. *Exploring Apple Silicon's potential from simulation and optimization perspective*
   Karol Struniawski, Aleksandra Konopka, and Ryszard Kozera.
   In: L. Franco, C. de Mulatier, M. Paszynski, V. V. Krzhizhanovskaya, J. J. Dongarra, P. M. A. Sloot (Eds.), *Computational Science – ICCS 2024*, Malaga (Spain), Lecture Notes in Computer Science, vol. 14836, Springer, 2024, pp. 35–42. ISBN 978-3-031-63775-9. DOI: 10.1007/978-3-031-63775-9_3.
   Contribution: 70% — Concept development, implementation, computational experiments, first manuscript draft, and conference poster design.
   Conference paper with presentation, 140 pkt. MNiSW

6. *TfELM: Extreme Learning Machines framework with Python and TensorFlow*
   Karol Struniawski, and Ryszard Kozera. *SoftwareX*, Elsevier, vol. 27, 2024, Article 101833, pp. 1–9. DOI: 10.1016/j.softx.2024.101833.
   Contribution: 90% — Full conceptualization, complete framework implementation, computational experiments, repository preparation, documentation, and first manuscript draft.
   Research article, 200 pkt. MNiSW, 2.4 Impact Factor (IF)

7. *Automated Identification of soil fungi and Chromista through Convolutional Neural Networks*
   Karol Struniawski, Ryszard Kozera, Paweł Trzciński, Anna Lisek, and Lidia Sas-Paszt.
   Engineering Applications of Artificial Intelligence, Elsevier, vol. 127B, 2024, Article 107333, pp. 1–12. DOI: 10.1016/j.engappai.2023.107333.
   Contribution: 60% — Study design, method implementation, computational experiments, and first manuscript draft.
   Research article, 140 pkt. MNiSW, 7.5 Impact Factor (IF)

8. *Credibility of randomness in Extreme Learning Machine*
   Karol Struniawski, Aleksandra Konopka, and Ryszard Kozera.
   In: J. Paliszkiewicz, J. Gołuchowski (Eds.), *Trust and Artificial Intelligence: Development and Application of AI Technology*, Routledge, 2025, pp. 92–106. ISBN 978-1-032-62632-1. DOI: 10.4324/9781032627236-10.
   Contribution: 70% — Study conceptualization, method implementation, computational experiments, and writing of the first manuscript version.
   Book chapter, 50 pkt. MNiSW

9. *Extreme Learning Machine for identifying soil-dwelling microorganisms cultivated on agar media*
   Karol Struniawski, Ryszard Kozera, Paweł Trzciński, and Agnieszka Marasek-Ciołakowska, and Lidia Sas-Paszt.
   Scientific Reports, Nature Publishing Group, vol. 14, 2024, Article 31034, pp. 1–23. DOI: 10.1038/s41598-024-82174-4.
   Contribution: 70% — Study conception, implementation, computational experiments, and preparation of the initial manuscript.
   Research article, 140 pkt. MNiSW, 3.8 Impact Factor (IF)

Summarizing:

- Number of publications contributing to PhD: 9 papers.
- Polish Ministerial points (MNiSW): 1020 points.
- Summarized Impact factor: 13.7.

2.2 Synopsis of research articles constituting the thesis

In publication *"TfELM: Extreme Learning Machines framework with Python and TensorFlow"* a flexible and accessible framework is discussed for applying Extreme Learning Machines (ELM) in Python, seamlessly integrating with *TensorFlow*, *CUDA*, and *scikit-learn* is introduced. To support open science, the framework is made publicly available via the Python Package Index, enabling researchers and practitioners to easily utilize it. *TfELM* is the only ELM framework fully compatible with *TensorFlow* 2 and *scikit-learn*, ensuring a smooth transition for users familiar with Python-based machine learning (ML) workflows. By adhering to *scikit-learn*, it facilitates essential functionalities such as automated metric computation through repeated cross-validation, model saving and loading, and Explainable Artificial Intelligence (XAI) techniques. The framework includes 18 ELM variants, offering a comprehensive set of options for researchers. Performance evaluations presented in the accompanying manuscript demonstrate that *TfELM* surpasses existing ELM implementations in both time execution, versatility and accuracy metrics.

In paper *"Exploring Apple Silicon's potential from simulation and optimization perspective"* the capabilities of Advanced Reduced Instruction Set Computing Machine System on a Chip architectures based on Apple's M-series processors were analyzed and compared to the NVIDIA RTX 3090 and a mid-range laptop equipped with an NVIDIA RTX 3050Ti. The study highlighted the power consumption differences, with M-series processors operating within a range of 10 to 31W, whereas PC-class systems with high-performance GPUs and CPUs can reach up to 800W at peak usage. Given the increasing importance of energy efficiency, the potential of such architectures for computational tasks was evaluated, considering their shared memory architecture and ability to allocate significant memory resources for intensive calculations. Various ML methods, including ELM, were tested across datasets with varying feature and sample sizes. The results demonstrated that the M2 processor is a viable option for ELM computations, offering an energy-efficient alternative for specific machine learning tasks.

ELM approach typically involves randomly initializing the weights between the input and hidden layers, though there are variants of ELM that are subject of further optimization processes. In publication *"Performance of selected nature-inspired metaheuristic algorithms used for Extreme Learning Machine"* various Metaheuristic Algorithms (MAs)

are applied to optimize these weights, and their performance is evaluated and compared across different benchmark datasets. One of the key findings is that fine-tuning the weights with MAs requires fewer iterations and for moderate population sizes, the hybrid MA-ELM model outperforms traditional ELM implementations. Conversely, researchers must carefully select the appropriate MA algorithm, as the results (measured in terms of accuracy) can vary significantly. Among the MAs tested, the Salp Swarm Algorithm and Manta Ray Foraging Optimization have shown particularly promising results.

While the tangent hyperbolic function is commonly used in ELMs, the ELM Theorem states that any activation function can be applied. In the paper *"Performance evaluation of activation functions in Extreme Learning Machine"* 36 different activation functions were compared across 10 datasets. The findings suggest that the choice of activation function should be closely tied to the specific dataset in use. However, the most versatile activation functions identified were those previously unexplored in ELM, specifically the Mish and Sexp functions, which are typically used in Deep Neural Networks.

The next paper, entitled *"Metaheuristic algorithms in Extreme Learning Machine for selection of parameters in activation function"* builds upon the MA-ELM framework by focusing not on optimizing the weights which require an extensive search of the feature space with thousands of parameters but on using MAs to optimize the parameters of the activation function itself. This study also extends the research on activation functions. Here 24 activation functions were compared across different datasets, with their parameters being subject to optimization. The results showed that, overall, the MA-driven optimization of activation function parameters led to an accuracy increase of the model.

Finally, concluding the study on optimizing ELM, the paper *"Credibility of randomness in Extreme Learning Machine"* examines the impact of random weight generation in ELM and how the choice of random generator implementation affects the results across different datasets. The study demonstrates that using a random generator can significantly impact the accuracy and stability of the results, as well as the selection of the distribution function for initializing the weights. Notably, the Pareto distribution function consistently delivered exceptional performance in the experiments.

The paper *"Identification of soil bacteria with Machine Learning and Image Processing Techniques applying Single Cells' Region Isolation"* explores the use of ELM in comparison to Support Vector Machine and Random Forest for soil bacteria classification. The study begins with microscopic images undergoing various image processing techniques to extract subimages containing single bacterial cells. These subimages enable the extraction of color, texture, and geometric features, which are then refined through a feature selection process. Classification is performed at the single-instance level, with results evaluated either individually or aggregated using a majority-voting rule. The findings highlight the efficiency of ELM in accurately classifying soil bacteria at the genus level, demonstrating its strong potential for single-instance processing based on statistical features.

Next, in the publication *"Automated identification of soil fungi and Chromista through Convolutional Neural Networks"* a newly curated dataset of soil fungi and Chromista was used to evaluate single-instance-driven identification with Convolutional Neural Networks (CNNs). The study tested classification at both the instance level and using the majority-voting method for genus identification. The CNNs demonstrated high accuracy in distinguishing classes, proving that the single-instance approach, which was effective for soil bacteria, is also applicable to soil fungi-despite their greater variation in shape. This work serves as a valuable reference point for ELM-based classification using statistical

image features.

Finally, the paper *"Extreme Learning Machine for identifying soil-dwelling microorganisms cultivated on agar media"* significantly expands the dataset of soil microorganisms, creating the most comprehensive collection of microscopic images in this field. Dataset 1, derived from previous research, allows for direct comparisons with earlier studies, while additional datasets were generated fully autonomously using an automated microscope, without expert biologist intervention. This study focuses on testing classification models against new data to assess their robustness. Various classifiers, including CatBoost, Random Forest, and ELM, were used alongside CNN-based feature extraction methods. The results were analyzed not only with standard classification metrics but also through Shapley Additive Explanations to provide insights into ELM's decision-making process and feature importance. Thanks to the *TfELM* framework, the study demonstrated that full-image classification, with preliminary background removal, is effective. Interestingly, the results showed that handcrafted features outperformed CNN-based features in this classification task.

Chapter 3

Content of thesis

3.1 *TfELM*: Extreme Learning Machines framework with Python and *TensorFlow*

Publication:

K. Struniawski and R. Kozera, *"TfELM: Extreme Learning Machines framework with Python and TensorFlow"*, SoftwareX, Vol. 27, 2024, doi: 10.1016/j.softx.2024.101833.

Abstract: *TfELM* introduces an innovative Python framework leveraging *TensorFlow* for Extreme Learning Machines (ELMs), offering a comprehensive suite for diverse machine learning (ML) tasks. Existing solutions in the ELM landscape lack comprehensive implementations. *TfELM* fills this gap by consolidating 18 ELM variants (including 14 so-far unimplemented in Python) into a unified framework. It conforms to established *scikit-learn* standards and emphasizes modularity, facilitating seamless integration into ML pipelines. Harnessing *TensorFlow*'s GPU acceleration, *TfELM* ensures rapid training and compatibility across varied computing environments. Notably, *TfELM* marks the inaugural ELM implementation in *TensorFlow* 2, featuring high-performance model saving/loading via HDF5 format, thus enhancing its novelty and alignment with contemporary standards. Performance evaluations demonstrate that *TfELM* outperforms other solutions, achieving significant speed enhancements across various computing platforms, with improvements of up to nine times tested on five standard UCI datasets.

SoftwareX 27 (2024) 101833

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

TfELM: Extreme Learning Machines framework with Python and *TensorFlow*

Karol Struniawski [a,*], Ryszard Kozera [a,b]

[a] *Institute of Information Technology, Warsaw University of Life Sciences - SGGW, ul. Nowoursynowska 166, Warsaw, 02-787, Masovian Voivodeship, Poland*
[b] *School of Physics, Mathematics and Computing, The University of Western Australia, 35 Stirling Highway, Perth, WA 6009, Western Australia, Australia*

ARTICLE INFO

Keywords:
Extreme Learning Machine
Machine learning
Artificial intelligence
Neural networks
Python
TensorFlow

ABSTRACT

TfELM introduces an innovative Python framework leveraging *TensorFlow* for Extreme Learning Machines (ELMs), offering a comprehensive suite for diverse machine learning (ML) tasks. Existing solutions in the ELM landscape lack comprehensive implementations. TfELM fills this gap by consolidating 18 ELM variants (including 14 so-far unimplemented in Python) into a unified framework. It conforms to established *scikit-learn* standards and emphasizes modularity, facilitating seamless integration into ML pipelines. Harnessing *TensorFlow's* GPU acceleration, TfELM ensures rapid training and compatibility across varied computing environments. Notably, TfELM marks the inaugural ELM implementation in *TensorFlow 2*, featuring high-performance model saving/loading via *HDF5* format, thus enhancing its novelty and alignment with contemporary standards. Performance evaluations demonstrate that TfELM outperforms other solutions, achieving significant speed enhancements across various computing platforms, with improvements of up to nine times tested on five standard UCI datasets.

Code metadata

Current code version	v1.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00239
Permanent link to Reproducible Capsule	N/A
Legal Code License	CC-BY-NC 4.0
Code versioning system used	git
Software code languages, tools, and services used	python
Compilation requirements, operating environments & dependencies	tensorflow-gpu (or tensorflow) >= 2.10.0, keras >= 2.10.0, numpy >= 1.24.0, h5py >= 3.7.0, scikit-learn >= 1.2.0, tqdm >= 4.66.0, mealpy >= 2.5.3
If available Link to developer documentation/manual	https://kstruniawski.github.io/TfELM/
Support email for questions	karol_struniawski@sggw.edu.pl

1. Motivation and significance

The concept of Extreme Learning Machines (ELMs) fundamentally revolves around dense single-layer feedforward neural networks, which offer robust solutions for both regression and classification tasks [1]. In a supervised setting, input data is represented as pairs $\eta = (x_i, t_i)_{i=1}^{N}$, where $X_{N\times d}$ denotes the input features for learning and $T_{N\times c}$ represents the corresponding targets. For classification, T consists of one-hot-encoded class labels, whereas for regression, it relies on one or more continuous target values. The architecture of an ELM typically involves input, single hidden and output layers. The input layer comprises neurons equal to the dimensionality d of the input data, while the number of neurons L in the hidden layer is determined beforehand, lacking a known efficient evaluation method [2]. In classification tasks, *softmax* or *argmax* functions are frequently employed as activation function of output layer to identify the final class [3].

The uniform distribution function generates random weights α between the input and hidden layers, as well as bias b, resulting in a random mapping of the input data. The crux of ELM lies in the determination of weights β between the hidden and output layers. These weights are computed by solving the equation $Y = H\beta$, where H represents the output from the hidden layer, computed as $H = f(X\alpha + b)$, with $f(\cdot)$ denoting any activation function. Directly solving

* Corresponding author.
E-mail address: karol_struniawski@sggw.edu.pl (Karol Struniawski).

https://doi.org/10.1016/j.softx.2024.101833
Received 26 April 2024; Received in revised form 17 July 2024; Accepted 22 July 2024
Available online 1 August 2024

this system is infeasible due to the irreversible nature of H [1]. Instead, β is estimated as the minimizer of the mean residual square error, given by the Moore–Penrose generalized inverse of H, denoted as $H^{\dagger}$ and rendering the final result as $\hat{\beta} = H^{\dagger}T$ [4]. The Pseudo-inverse of matrix $H^{\dagger}$ uniquely determines $\hat{\beta}$ such that $H\hat{\beta}$ closely approximates Y in terms of mean square error [5].

ELMs offer a distinct advantage over traditional iterative learning methods by computing optimal weights in a single run, circumventing issues associated with vanishing or exploding gradients and suboptimal solutions [6–8]. This non-iterative approach ensures computational efficiency and mitigates common pitfalls encountered in iterative learning algorithms. Compared to methods such as the Multilayer Perceptron algorithm with Backpropagation, ELMs demonstrate remarkable speed and resource efficiency, with learning rates hundreds of times faster [9,10]. This efficiency makes ELMs particularly well-suited for time-sensitive applications and large-scale datasets, offering a compelling alternative for rapid model training.

Following the formulation of the ELM, the process of modifying the original solution is initiated. The first idea involves incorporating the concept of ridge regression theory, which suggests adding a positive value to the diagonal of $H^T H$ or $H H^T$ to achieve a more stable solution and better generalization performance [11]. This concept is integrated into TfELM through an additional parameter C passed to the model. Subsequent efforts focus on optimizing the β weights after their calculation using Moore–Penrose pseudoinverse operations, minimizing the *l1*, *l2*, or combined *l12* loss with a given optimization method to form the Regularized ELM [12]. *l1* Loss Function is used to minimize the error which is the sum of the all the absolute differences between the true value and the predicted value, whereas *l2* is the sum of the all the squared differences between the true and the predicted value. Still, the challenge of determining the number of neurons in the hidden layer L persists. This is addressed by the Kernel Extreme Learning Machine, which incorporates the evaluation of the matrix $H^T H$ or $H H^T$ by the kernel matrix, leveraging the Mercer kernel theorem [13]. Various kernels, including combined kernels, are further explored and also implemented in TfELM. Another approach involves tuning the randomly generated weights α using Metaheuristic Algorithms (MA), resulting in MA-ELM [14]. Specific ELM variants, such as Constrained and Weighted variants, are developed to handle imbalanced datasets by generating specially crafted α weights. In addition, receptive fields can change these weights to favor input in the center, as inspired by human vision in Receptive Fields ELM (RF-ELM). Moreover, researchers explore not only supervised but also semi-supervised and unsupervised methods, with the latter being ideal for data embedding and clustering tasks. To address the challenge of handling large-scale data, the Online Sequential ELM (OS-ELM) is introduced, allowing learning from data chunks with minimal memory consumption. Furthermore, multilayer ELM structures are developed based on autoencoder ELM (AE-ELM), which maps input and target data into a different space defined by ELM parameters. This enables the formulation of multilayer structures composed of stacked AE-ELMs and a typical ELM at the last layer. The TfELM framework's modular architecture and object-oriented programming facilitate interchangeable usage and the creation of custom utilities. This flexibility permits the exploration of various ELM variant combinations not previously addressed in the literature, offering ample opportunities for future research endeavors.

The motivation for development of TfELM lies in response to the identified deficiencies within existing ELMs implementations.

1. Current solutions lack a comprehensive framework capable of integrating diverse ELM variants while maintaining compatibility with modern ML tools like *TensorFlow* and *scikit-learn*.
2. Compounding this challenge is the scarcity of available ELM implementations, with only four variants provided in Python, none of which is conforming to modern *TensorFlow 2* or is integrating seamlessly with *scikit-learn*.
3. The proposed tool effectively fills up this significant gap in the realm of ELMs. In Table 1 the summary of already existing solutions is presented.
4. Notably, 14 implemented variants covered by TfELM have yet to be implemented in Python, and 16 remain unexplored in *TensorFlow*.
5. TfELM for all of its 18 variants conforms to the Object Oriented Programming paradigm in Python with newest *TensorFlow* of 2.15 and *scikit-learn* modules compatibility (details with source code are provided in Code metadata).

Table 1
Summary of the existing implementations: TF denotes TensorFlow, PY Python, and TF2 signifies TensorFlow version 2, while SK refers to scikit-learn. All 18 variants of TfELM adhere to Python with the latest TensorFlow version 2.15 and scikit-learn.

Version of ELM	Reference	PY	TF	TF2	SK
Basic	[1]	✓	✓	✗	✗
Constrained	[17]	✗	✗	✗	✗
Deep	[18]	✗	✗	✗	✗
Deep Representation	[19]	✗	✗	✗	✗
Enhanced Deep Representation	[19]	✗	✗	✗	✗
Graph Regularized Autoencoder	[20]	✗	✗	✗	✗
Kernel	[13]	✗	✗	✗	✗
Local Receptive Field	[21]	✗	✗	✗	✗
Metaheuristic	[22]	✗	✗	✗	✗
Multi-layer	[23]	✓	✗	✗	✗
Online Sequential	[24]	✓	✗	✗	✗
Regularized	[12]	✓	✓	✗	✗
Receptive Fields	[25]	✗	✗	✗	✗
Residual Compensation	[26]	✗	✗	✗	✗
Semi-Supervised	[27]	✗	✗	✗	✗
Subnetwork	[10]	✗	✗	✗	✗
Unsupervised	[28]	✗	✗	✗	✗
Weighted	[29]	✗	✗	✗	✗

This application covers a significant need in publicly available ELM solutions, catering to a wide spectrum of users ranging from students to experts. Existing implementations are either outdated or lack compatibility with current *TensorFlow* versions. For instance, one *TensorFlow*-based implementation from 2018 by Cornell covers only the fundamental ELM concept and has not been updated since its release [15]. Another implementation, the OS-ELM from 2018 by Otenim [16], does not offer compatibility with modern *TensorFlow* versions.

TfELM's novelty stems from its adherence to *scikit-learn* and its fully modular nature, enabling new users to leverage it effortlessly within existing ML pipelines. The modular design empowers researchers to explore novel combinations of ELM variants, filling a crucial void in the ELM landscape.

TfELM leverages TensorFlow's GPU acceleration to ensure swift training and compatibility across diverse computing environments. Supported by the documentation, examples, and model persistence capabilities, TfELM streamlines the exploration and deployment of ELM-based solutions with remarkable efficiency. TfELM's compatibility with modern TensorFlow versions ensures its usability across various devices, from high-performance GPUs to CPU-based systems like Apple's M-series processors. Such versatility extends TfELM's applicability to a broad spectrum of computing environments, further enhancing its significance in the field of ML.

One of the crucial goals of TfELM is to optimize performance while maintaining versatility. Extensive preliminary experiments were conducted on five standard datasets to optimize operations within ELM, focusing on both GPU-compatible and non-GPU users. Performance evaluations conducted on different devices demonstrate TfELM's superiority over existing solutions, showcasing significant speedups in execution times. These observations underscore TfELM's potential to revolutionize ELM implementations. The primary research question tackled here is whether it is possible to develop a versatile framework that combines different ELMs, integrates the latest modules,

and still delivers high performance. TfELM goes beyond state-of-the-art solutions, being the first ELM implementation to conform to both TensorFlow 2 and scikit-learn.

2. Software description

TfELM is a Python and *TensorFlow* framework for ELMs, renowned for simplicity and efficiency. It offers efficient training, versatile applications, model persistence and a modular design for various ELM's types. With comprehensive documentation and examples, it facilitates seamless implementation of ELMs for various tasks.

2.1. Extreme learning machines

The concept of ELMs fundamentally revolves around dense single-layer feedforward neural networks, which offer robust solutions for both regression and classification tasks [1]. The uniform distribution function generates random weights between the input and hidden layers, as well as bias, resulting in a random mapping of the input data. The main crux of ELM lies in the determination of weights between the hidden and output layers by the Moore–Penrose generalized inverse [4]. ELMs offer a distinct advantage over traditional iterative learning methods by computing optimal weights in a single run, circumventing issues associated with vanishing or exploding gradients leading to potentially suboptimal solutions [6–8]. This non-iterative approach ensures computational efficiency and mitigates common pitfalls encountered in iterative learning algorithms.

Following the initial formulation of ELM in 2005, numerous variants emerged, from which we meticulously selected and eventually implemented 18. The comprehensive list of these variants is presented in Table 1. Notably, each variant encompasses numerous sub-variants, such as different methods for calculating Weighted ELMs, initially dispersed across various papers. In essence, TfELM encompasses more than 18 distinct versions, thanks to its expansive coverage of sub-variants and methodologies within the framework.

2.2. Software architecture

The framework is structured into distinct directories for ease of navigation and understanding. The **Data** directory contains sample datasets sourced from the UCI repository [30], serving as an initial resource for researchers engaging with the framework. Within the **Examples** directory, users can find 26 exemplary code snippets meticulously annotated to illustrate the application of specific methods on various datasets. For tasks involving feature embedding in Unsupervised ELM [28], these examples also provide visual representations of the results.

The core functionalities of the framework are encapsulated within the **Layers**, **Models** and **Optimizers** directories. **Layers** contain a diverse range of ELM layers, each comprising input, hidden and output layers. These modules offer fundamental utilities such as fitting, prediction, building and saving functionalities, allowing researchers to construct custom ELM variants not covered in the initial framework version thanks to Object Oriented Paradigm applied to TfELM.

The most crucial component i.e. the **Models** forms specifically crafted classes designed as subclasses of `BaseEstimator` and `ClassifierMixin` or `RegressorMixin` from *scikit-learn* package. These classes facilitate seamless integration with popular *scikit-learn* functions like `cross_val_score`, supporting also probability metrics through the implementation of the `predict_proba` method. TfELM's models also possess loading and saving capabilities, enhancing their versatility. The models can accept one or more layer instances from **Layers** (for multilayer ELMs, the `add` function facilitates the stacking of layers). This flexibility enables the combination of various ELM layer variants, including i.e. `Kernel` or `SubELMLayer`, across all multilayer models. Such adaptability is invaluable for accommodating future variants.

Optimizers directory encompasses optimizers tailored for Metaheuristics (MAs) and regularized ELMs, optimizing wages based on regularization norms such as l_1, l_2. Notably, TfELM uses the *mealpy* package for its MAs, which is the widely recognized golden standard for Python implementations. The module contains hundreds of MA implementations and is open to new ones, allowing users to implement any custom MA that conforms to *mealpy*'s requirements and easily utilize the newest ones for ELMs [31–33]. The **Resources** directory houses additional supporting functions, while the **docs** directory serves as the source for GitHub Pages documentation.

2.3. Software functionalities

TfELM forms a robust and a comprehensive solution for a myriad of tasks covering classification, regression, feature mapping and embedding. It refers to methods catering to *supervised, semi-supervised and unsupervised* learning modes, covering a wide spectrum of ELM techniques that seamlessly integrate with each other. The foundational Basic `ELMLayer` serves not only as a provider for `ELMModel` but also as one of the layers in the Multilayer ELM.

These algorithms underwent rigorous transformation for *TensorFlow* compatibility and optimization. For instance, the Nyström approximation (see - [34]) of the kernel matrix was employed to enhance the efficiency of Kernel and Multilayer Kernel ELM which is not covered in the literature. Noteworthy, the versatility of TfELM extends beyond the listed models. More specifically:

- Most layers and models are interchangeable.
- Users have the flexibility to mix and match various methods, adjust parameter values and even combine methods to suit their specific requirements.
- Some of these combinations might not have been previously fully explored in the literature, offering fertile ground for new research endeavors.

2.4. Performance evaluation

After constructing the module, close attention was paid to its performance, employing various optimization techniques to enhance calculation speed. Subsequently, comparisons were made, focusing on implementations suitable for benchmarking against basic ELM. Three specific implementations are considered:

- the initial implementation by Putra [35] utilizing core Python,
- the second by Otenim [16] employing the numpy module,
- the last utilizing TensorFlow 1 [15].

Following preliminary experiments, the TensorFlow 1 implementation was discarded due to its notably poorer performance. The remaining two implementations were evaluated against TfELM using well-established and used as reference datasets in ML field including *ionosphere*, *musk*, *abalone*, *navigation*, and *eyestate*. They are publicly available through UCI Repository [30], on two distinct devices. The first device, a PC running Windows 11 with specifications including Ryzen 7 3700X, 64 GB RAM, and RTX 3060 12 GB VRAM, utilized TensorFlow 2.10 and Python 3.10, with both GPU-enabled (TfELM-G) and disabled (TfELM-C) modes tested. The second device, a MacBook Air 15 from 2023 equipped with an M2 processor and 16 GB RAM, ran TensorFlow 2.15 and Python 3.10 on MacOS 14.2.1. The details about utilized datasets are provided in Table 2. Given the significance of hidden nodes amount in ELMs, the algorithm was executed with 10 repetitions of 10-fold cross-validation to ascertain optimal performance. This approach minimized random noise in measurements, ensuring statistically meaningful results. Final conclusions were drawn from the *ionosphere* and *eyestate* datasets, chosen to represent both ends of the spectrum in terms of size, with *ionosphere* being the smallest and *eyestate* being the largest.

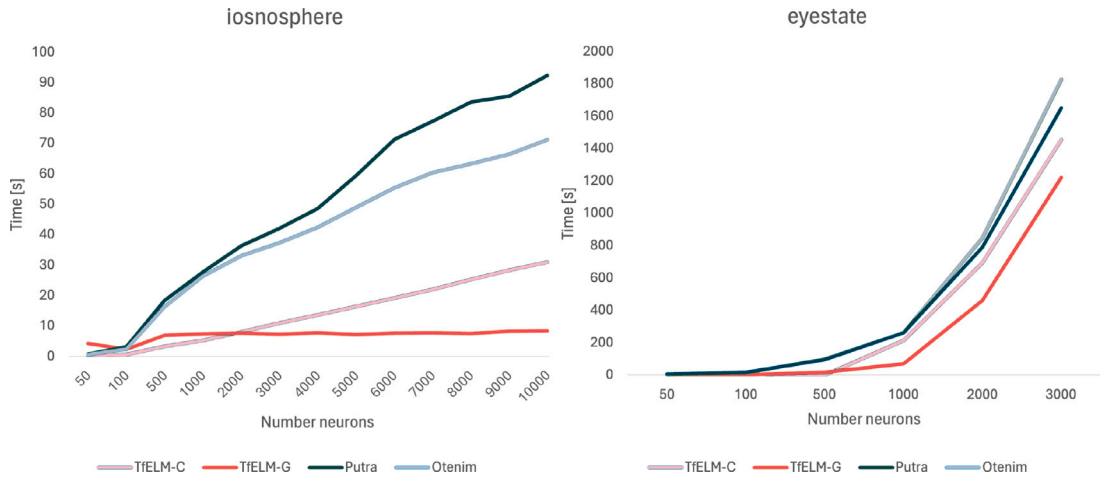


Fig. 1. Execution time comparison on PC for the ionosphere and eyestate datasets.

Table 2
UCI datasets: number of features and samples.

Dataset	No. features	No. samples	No. classes
Ionosphere	34	351	2
Abalone	8	4177	3
Navigation	14	5456	4
Musk	166	6598	2
EyeState	14	14,980	2

Table 3
Performance metrics for UCI datasets.

Dataset	ACC	F1-Macro	AUC
ionosphere	0.938	0.928	0.979
navigation	0.721	0.640	0.902
musk	0.967	0.964	0.991

The findings from *ionosphere* dataset, conducted on a Windows device, consistently demonstrate TfELM-C's superior performance over both Putra [35] and Otenim [16] implementations across all evaluated setups (see Fig. 1). The advantages of employing GPU calculations are evident, with execution times being up to *nine times faster* compared to alternative solutions. Additionally, results obtained from the *eyestate* dataset further underscore the remarkable computational efficiency achieved by both TfELM-C and TfELM-G.

Fig. 2 refers to the execution times of the code on an M2 processor for the ionosphere and eyestate datasets. The plot consistently demonstrates shorter execution times across all configurations for TfELM. This showcases the superior efficiency and optimization of TfELM for the given task, making it a favorable choice for machine learning tasks on the M2 processor.

Additionally, during the experiments we noted the classifier's accuracy, F1-macro and AUC for basic ELM algorithm with 1000 neurons and mish activation function — see Table 3. These experiments point out the ELM potential capabilities even in core version.

For a further comprehensive comparison, TfELM is also evaluated against other ML methods, demonstrating superior capabilities in high-performance computing. Indeed, that was one of the crucial aims in designing the proposed framework. TfELM outperforms all other ML methods, including high-performance ones like SVM, as shown in Table 4 and Fig. 3. Performance evaluation was performed on MacBook Air 15 M2 16 GB, Python 3.10, Tensorflow 2.15, Catboost 1.2.5, scikit-learn 1.5.0, xgboost 2.0.3 that are standard frameworks for a given model.

Table 4
Execution time of various ML methods (in seconds).

Method	Ionosphere	Navigation	Musk
SVM	0.1453	3.188	3.256
GradientBoosting	17.5287	212.7845	367.5672
RandomForest	6.8334	42.0421	81.4316
AdaBoost	5.1562	41.0533	72.8829
MLPClassifier	127.4873	524.6581	376.6111
XGBoost	4.3851	15.8917	11.0553
CatBoost	258.6639	557.0551	540.0941
ELM (TfELM)	1.8310	1.7619	2.0921

3. Illustrative examples

The highlighted example demonstrates the practical implementation of the Unsupervised Kernel ELM (USKELM) with a combination of multiple kernels. Utilizing the USKELMModel, the data embedding process is facilitated by the designated USKELMLayer, incorporating the predefined kernels. The objective is to leverage the provided dataset (in this case, G50C dataset) and project it into a two-dimensional space. Refer to Fig. 4 for the corresponding code snippet and Fig. 5 for the output visualization.

The second example (see Fig. 6) refers to the Multilayer ELM. TfELM facilitates the creation of custom `ELMLayer` instances, which are incorporated into the model using the `add` method. `ELMLayers` can vary from one another; for instance, some may be Kernel ELM Layers (`KELMLayer` — see Fig. 7), while others may apply denoising, with parameters passed to the `ELMLayer`. This flexibility makes the framework highly versatile. Following the algorithm's formulation, it is fitted to the example dataset – in this case, the ionosphere dataset – and evaluated using `RepeatedKFold` from *scikit-learn*, with accuracy serving as the metric. Subsequently, the saving and loading functionalities are demonstrated. Lastly, the Fuzzy variant of ELM [36] is represented as Fig. 8 showcasing versatility of the TfELM framework.

4. Impact

The TfELM addresses an urgent need for a comprehensive toolkit to implement and experiment with ELMs using Python and *TensorFlow*. The significance of the software lies in its ability to bridge the gap in ELM implementations by offering a comprehensive framework equipped with various ELM variants and utilities. By providing researchers and practitioners with easy access to powerful tools for building and evaluating ELM models, the software facilitates seamless experimentation and deployment of ELM-based solutions across diverse domains.

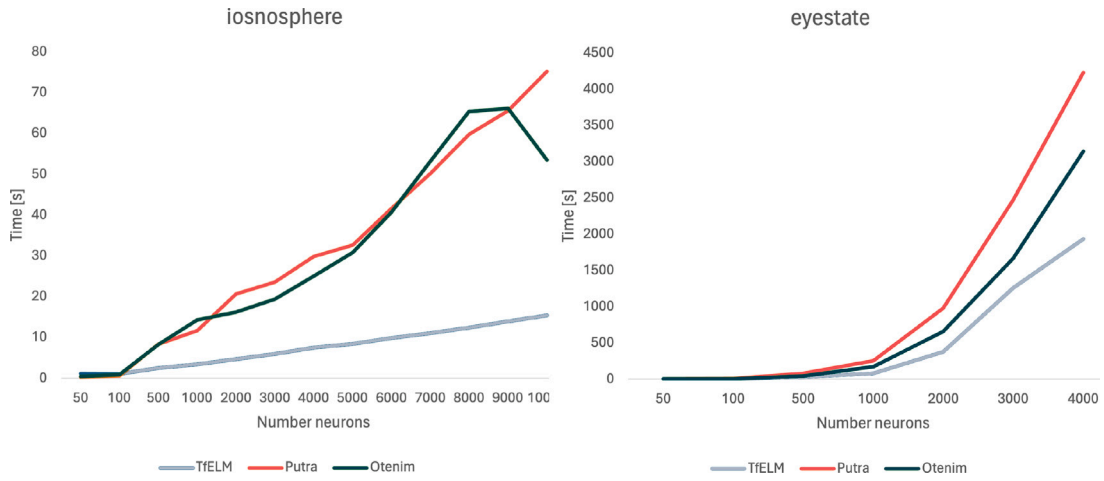


Fig. 2. Execution time comparison on M2 for the ionosphere and eyestate datasets.

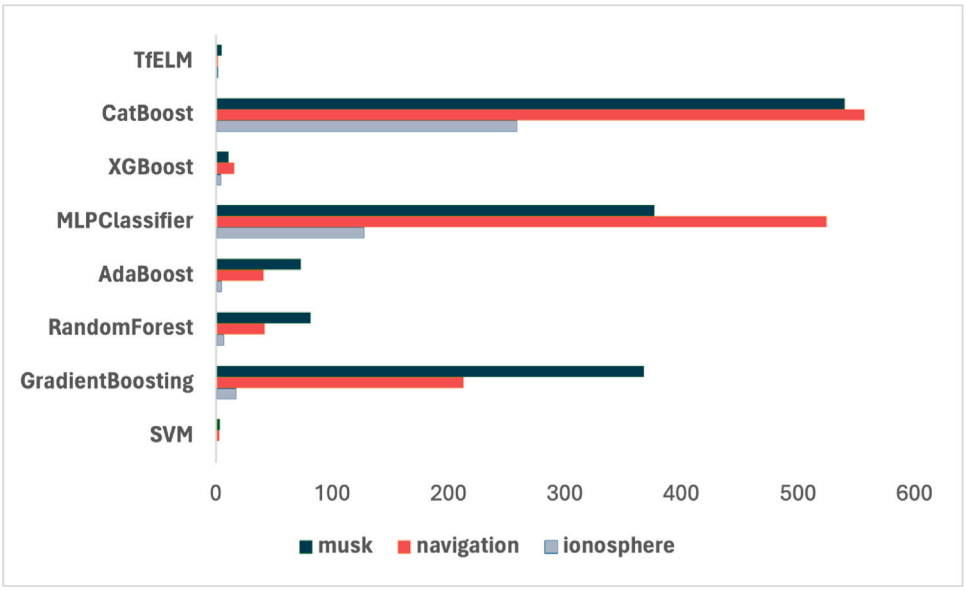


Fig. 3. Execution Time of Various ML Methods (in seconds).

```
# 2-dim embedding
kernel = CombinedProductKernel([Kernel("rational_quadratic"),
                                Kernel("exponential")])
layer = USKELMLayer(kernel=kernel, embedding_size=2, lam=0.001)
model = USKELMModel(layer=layer)
model.fit(X)

model.save("Saved Models/USKELM_Model_1.h5")
model = model.load("Saved Models/USKELM_Model_1.h5")

pred = model.predict(X)

# Create a scatter plot
plt.scatter(pred[:, 0], pred[:, 1], c=y.flatten())

# Show the plot
plt.show()
```

Fig. 4. Code example demonstrating the implementation of Unsupervised Kernel Extreme Learning Machine for embedding tasks into a two-dimensional space. This code is applied on the G50C dataset (data preprocessing steps omitted for clarity).

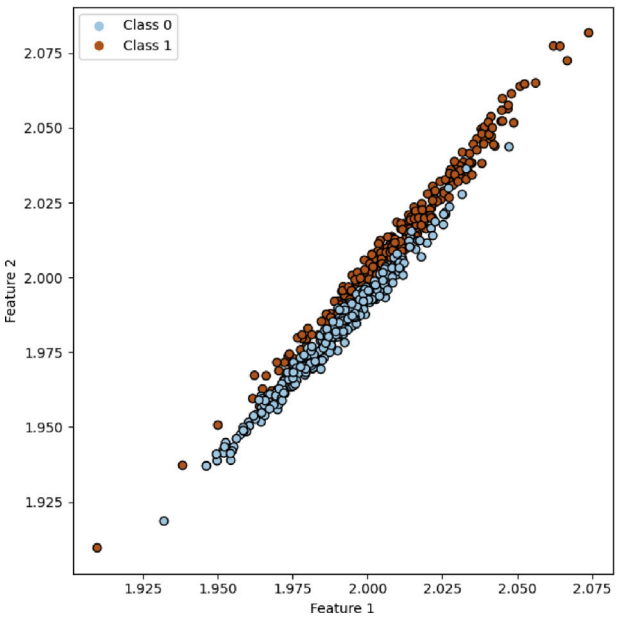


Fig. 5. Visualization depicting the effect of running the G50C data embedding task into a two-dimensional space using USKELM.

```
# Initialize a Multilayer Extreme Learning Machine model
model = ML_ELMModel(verbose=0)

# Add ELM layers to the Multilayer Extreme Learning Machine
model.add(ELMLayer(number_neurons=50))
model.add(ELMLayer(number_neurons=60))
model.add(ELMLayer(number_neurons=50))
model.add(ELMLayer(number_neurons=1000))

# Define a cross-validation strategy
n_splits = 10
n_repeats = 10
cv = RepeatedKFold(n_splits=n_splits, n_repeats=n_repeats)

# Perform cross-validation to evaluate the model performance
scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy')

# Print the mean accuracy score obtained from cross-validation
print(np.mean(scores))

# Fit the ML-ELM model to the entire dataset
model.fit(X, y)

# Save the trained model to a file
model.save('Saved Models/ML_ELM_Model.h5')

# Load the saved model from the file
model = model.load('Saved Models/ML_ELM_Model.h5')

# Evaluate the accuracy of the model on the training data
acc = accuracy_score(model.predict(X), y)
print(acc)
```

Fig. 6. Code example demonstrating the implementation of Multilayer Extreme Learning Machine.

```
# Initialize a Kernel (it can be instanced as Kernel class and its
# subclasses like CombinedProductKernel)
kernel = CombinedProductKernel([Kernel("rational_quadratic"),
                                Kernel("exponential")])

# Initialize a Multilayer Extreme Learning Machine model
model = ML_ELMModel(verbose=0)

# Add KELM layers to the Multilayer Extreme Learning Machine
model.add(KELMLayer(kernel=kernel))
model.add(KELMLayer(kernel=kernel))
model.add(KELMLayer(kernel=kernel))
model.add(ELMLayer(number_neurons=1000))

# Define a cross-validation strategy
n_splits = 10
n_repeats = 10
cv = RepeatedKFold(n_splits=n_splits, n_repeats=n_repeats)

# Perform cross-validation to evaluate the model performance
scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy')

# Print the mean accuracy score obtained from cross-validation
print(np.mean(scores))

# Fit the ML-ELM model to the entire dataset
model.fit(X, y)

# Save the trained model to a file
model.save('Saved Models/ML_KELM_Model.h5')

# Load the saved model from the file
model = model.load('Saved Models/ML_KELM_Model.h5')

# Evaluate the accuracy of the model on the training data
acc = accuracy_score(model.predict(X), y)
print(acc)
```

Fig. 7. Code example demonstrating the implementation of Kernel Multilayer Extreme Learning Machine.

```
# Fuzzify the imput
X = fuzzify_input(X)

# Initialize an Extreme Learning Machine (ELM) layer
elm = ELMLayer(number_neurons=num_neurons, activation='mish')

# Create an ELM model using the trained ELM layer
model = ELMModel(elm)

# Define a cross-validation strategy
cv = RepeatedKFold(n_splits=n_splits, n_repeats=n_repeats)

# Perform cross-validation to evaluate the model performance
scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy',
↪  error_score='raise')

# Print the mean accuracy score obtained from cross-validation
print(np.mean(scores))
```

Fig. 8. Code example demonstrating the implementation of Fuzzy Extreme Learning Machine.

ELMs find application across a wide array of domains, showcasing their versatility and importance in addressing real-world challenges. The availability of easily accessible frameworks for researchers is crucial for advancing the application of ELMs in practical tasks. These tasks span a multitude of domains, including disease forecasting [37], breast cancer detection [38], intrusion detection [39], water pipeline leak detection [40], corporate failure detection [41], speech emotion detection [42], wind power prediction [43], state of battery charge estimation [44] and countless others. The versatility of ELM software, coupled with its implementation of numerous methods and open-source nature as TfELM, plays a pivotal role in solving engineering problems. This accessibility and adaptability not only facilitate the development of ELM applications but also contribute to the advancement of various disciplines by providing efficient solutions to complex problems.

5. Conclusions

In conclusion, this paper presents a comprehensive framework — TfELM, for implementing and experimenting with ELMs using Python and *TensorFlow*. The framework offers a versatile toolkit for various tasks including *classification, regression, feature learning and embedding*. By leveraging *TensorFlow*'s computational graph and optimization algorithms, TfELM provides efficient training times, making it suitable for large-scale datasets. Together with extensive documentation and 18 implemented models, TfELM facilitates seamless integration with *scikit-learn* functions and offers flexibility for custom ELM variants. Ultimately, TfELM makes ELM approaches more accessible by providing an open-source option for researchers and practitioners.TfELM's performance assessments demonstrate its superiority over competing systems, showing significant time-execution speed-ups across x64 platform with GPU enabled/disabled and ARM based tested on five different datasets. This addresses the research question of whether a complex framework can still perform exceptionally well. The current limitations of TfELM are primarily related to the GPU memory of the device running the given algorithm which are in question. A significant performance boost is observed until the memory limit is reached, at which point swap memory is used, resulting in natural performance disadvantages. This drawback of GPU calculations is common across all frameworks that rely on GPU. Developers working with large datasets require substantial GPU memory. Future research and development directions may possibly refer to the issue of TfELM enhancement. The latter includes to enhance TfELM include implementing the framework in core C++ and NVIDIA CUDA, and creating a port for the Python environment to further boost performance.

CRediT authorship contribution statement

Karol Struniawski: Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ryszard Kozera:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: A new learning scheme of feedforward neural networks. In: 2004 IEEE international joint conference on neural networks, vol. 2. 2004, p. 985–90. http://dx.doi.org/10.1109/IJCNN.2004.1380068.

[2] Freire AL, Barreto G D. A new model selection approach for the ELM network using metaheuristic optimization. In: The European symposium on artificial neural networks. 2014.

[3] Denker JS, LeCun Y. Transforming neural-net output levels to probability distributions. Neural Inf Process Syst 1990.

[4] Rao CR, Mitra SK. Generalized inverse of matrices and its applications. John Wiley & Sons; 1971.

[5] Struniawski K, Kozera R, Konopka A. Performance of selected nature-inspired metaheuristic algorithms used for extreme learning machine. In: Computational science – ICCS 2023, vol. 10475. Cham: Springer; 2023, p. 494–503. http://dx.doi.org/10.1007/978-3-031-36024-4_38.

[6] Zhang J, Li Y, Xiao W, Zhang Z. Non-iterative and fast deep learning: Multilayer extreme learning machines. J Franklin Inst 2020;357(13):8925–55. http://dx.doi.org/10.1016/j.jfranklin.2020.04.033.

[7] MacDonald G, Godbout A, Gillcash B, Cairns S. Volume-preserving neural networks: A solution to the vanishing gradient problem. 2019, arXiv, abs/1911.09576.

[8] Zhang J, Lei Q, Dhillon IS. Stabilizing gradients for deep neural networks via efficient SVD parameterization. 2018, arXiv, abs/1803.09327.

[9] Tang J, Deng C, Huang G-B. Extreme learning machine for multilayer perceptron. IEEE Trans Neural Netw Learn Syst 2016;27(4):809–21. http://dx.doi.org/10.1109/TNNLS.2015.2424995.

[10] Yang Y, Wu Q MJ. Extreme learning machine with subnetwork hidden nodes for regression and classification. IEEE Trans Cybern 2016;46(12):2885–98. http://dx.doi.org/10.1109/TCYB.2015.2492468.

[11] Hoerl AE, Kennard RW. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics 1970;12(1):55–67. http://dx.doi.org/10.1080/00401706.1970.10488634.

[12] Deng W, Zheng Q, Chen L. Regularized extreme learning machine. In: 2009 IEEE symposium on computational intelligence and data mining. 2009, p. 389–95. http://dx.doi.org/10.1109/CIDM.2009.4938676.

[13] Alizamir M, Kim S, Zounemat-Kermani M, Heddam S, Kim NW, Singh VP. Kernel extreme learning machine: An efficient model for estimating daily dew point temperature using weather data. Water 2020;12(9):2600. http://dx.doi.org/10.3390/w12092600.

[14] Wu L, Huang G, Fan J, Ma X, Zhou H, Zeng W. Hybrid extreme learning machine with meta-heuristic algorithms for monthly pan evaporation prediction. Comput Electron Agric 2020;168:105115. http://dx.doi.org/10.1016/j.compag.2019.105115.

[15] Cornell S. Tf-ELM. 2018, Retrieved from https://github.com/popcornell/tfelm.

[16] Otenim (GitHub's nickname). 2018, TF-OS-ELM. Retrieved from https://github.com/otenim/TensorFlow-OS-ELM.

[17] Zhu W, Miao J, Qing L. Constrained extreme learning machine: A novel highly discriminative random feedforward neural network. In: 2014 international joint conference on neural networks. 2014, p. 800–7. http://dx.doi.org/10.1109/IJCNN.2014.6889761.

[18] Lendasse A, Ding S, Zhang N, Xu X, Guo L, Zhang J. Deep extreme learning machine and its application in EEG classification. Math Probl Eng 2015;129021. http://dx.doi.org/10.1155/2015/129021.

[19] Yu W, Zhuang F, He Q, Shi Z. Learning deep representations via extreme learning machines. Neurocomputing 2015;149(A):308–15. http://dx.doi.org/10.1016/j.neucom.2014.03.077.

[20] Sun K, Zhang J, Zhang C, Hu J. Generalized extreme learning machine autoencoder and a new deep neural network. Neurocomputing 2017;230:374–81. http://dx.doi.org/10.1016/j.neucom.2016.12.027.

[21] Huang G-B, Bai Z, Kasun LLC, Vong CM. Local receptive fields based extreme learning machine. IEEE Comput Intell Mag 2015;10(2):18–29. http://dx.doi.org/10.1109/MCI.2015.2405316.

[22] Bacanin N, Stoean C, Zivkovic M, Jovanovic D, Antonijevic M, Mladenovic D. Multi-swarm algorithm for extreme learning machine optimization. Sensors 2022;22(11):4204. http://dx.doi.org/10.3390/s22114204.

[23] Kale GA, Karakuzu C. Multilayer extreme learning machines and their modeling performance on dynamical systems. Appl Soft Comput 2022;122:108861. http://dx.doi.org/10.1016/j.asoc.2022.108861.

[24] Wibawa IPD, Machbub C, Rohman AS, Hidayat E. Modified online sequential extreme learning machine algorithm using model predictive control approach. Intell Syst Appl 2023;18:200191. http://dx.doi.org/10.1016/j.iswa.2023.200191.

[25] McDonnell MD, Tissera MD, Vladusich T, van Schaik A, Tapson J. Fast, simple and accurate handwritten digit classification by training shallow neural network classifiers with the 'extreme learning machine' algorithm. PLOS ONE 2015;10(8):e0134254. http://dx.doi.org/10.1371/journal.pone.0134254.

[26] Zhang J, Xiao W, Li Y, Zhang S. Residual compensation extreme learning machine for regression. Neurocomputing 2018;311:126–36. http://dx.doi.org/10.1016/j.neucom.2018.05.057.

[27] Abuassba AO, Dezheng Z, Mahmood Z. Semi-supervised multi-kernel extreme learning machine. Procedia Comput Sci 2018;129:305–11. http://dx.doi.org/10.1016/j.procs.2018.03.080.
[28] Huang G, Song S, Gupta JND, Wu C. Semi-supervised and unsupervised extreme learning machines. IEEE Trans Cybern 2014;44(12):2405–17. http://dx.doi.org/10.1109/TCYB.2014.2307349.
[29] Ban X, Liu R, Shen Q, Wang Y. Weighted extreme learning machine for balance and optimization learning. In: 2016 4th international conference on cloud computing and intelligence systems. 2016, p. 6–10. http://dx.doi.org/10.1109/CCIS.2016.7790215.
[30] Dua D, Graff C. UCI machine learning repository. Irvine, CA: University of California, School of Information and Computer Science; 2019, Retrieved from https://archive.ics.uci.edu/ml.
[31] Gajic L, Cvetnic D, Zivkovic M, Bezdan T, Bacanin N, Milosevic S. Multi-layer perceptron training using hybridized bat algorithm. In: Computational vision and bio-inspired computing. Advances in intelligent systems and computing, vol. 1318, Singapore: Springer; 2021, http://dx.doi.org/10.1007/978-981-33-6862-0_54.
[32] Strumberger I, Bacanin N, Tuba M. Enhanced firefly algorithm for constrained numerical optimization. In: 2017 IEEE congress on evolutionary computation. Spain: Donostia; 2017, p. 2120–7. http://dx.doi.org/10.1109/CEC.2017.7969561.
[33] Bacanin N, Stoean R, Zivkovic M, Petrovic A, Rashid TA, Bezdan T. Performance of a novel chaotic firefly algorithm with enhanced exploration for tackling global optimization problems: Application for dropout regularization. Mathematics 2021;9(21):2705. http://dx.doi.org/10.3390/math9212705.
[34] Nystrom EJ. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. Acta Math 1930;54:185–204.
[35] V. Putra O. Simulasi paper extreme learning machine: Theory and applications. 2022, Retrieved from https://github.com/virgantara/Extreme-Machine-Learning.
[36] Zhang WB, Ji HB. Fuzzy extreme learning machine for classification. Electron Lett 2013;49:448–50. http://dx.doi.org/10.1049/el.2012.3642.
[37] Sibi SA, Bushra SN, Revathi M, Godbin AB, Akshaya S. Multitudinous disease forecasting using extreme learning machine. In: Challa RK, et al., editors. Artificial intelligence of things, vol. 1929. Cham: Springer; 2024, p. 256–67. http://dx.doi.org/10.1007/978-3-031-48774-3_22.
[38] Muduli D, Kumar RR, Pradhan J, et al. An empirical evaluation of extreme learning machine uncertainty quantification for automated breast cancer detection. Neural Comput Appl 2023. http://dx.doi.org/10.1007/s00521-023-08992-1.
[39] Wang K, Li J, Wu W. A novel transfer extreme learning machine from multiple sources for intrusion detection. Peer-to-Peer Network Appl 2024;17:33–47. http://dx.doi.org/10.1007/s12083-023-01569-8.
[40] Liu M, Guo G, Xu Y, Yang Y, Liu N. Performance of improved Gaussian extreme learning machine for water pipeline leak recognition. IEEE Sens J 2024;24(6):8474–83. http://dx.doi.org/10.1109/JSEN.2024.3360185.
[41] Veganzones D. Predicting corporate failure using ensemble extreme learning machine. In: Novel financial applications of machine learning and deep learning. vol. 336, Cham: Springer; 2023, p. 123–36. http://dx.doi.org/10.1007/978-3-031-18552-6_7.
[42] Koti VM, Murthy K, Suganya M, Sarma MS, Seshu Kumar GVS, Balamurugan N. Speech emotion recognition using extreme machine learning. EAI Endorsed Trans Internet Things 2023;10. http://dx.doi.org/10.4108/eetiot.4485.
[43] Wang Z-C, Niu J-C. Wind power output prediction: A comparative study of extreme learning machine. Front Energy Res 2023;11:1267275. http://dx.doi.org/10.3389/fenrg.2023.1267275.
[44] Zhang Y, Zhang Z, Chen J, et al. The adaptive kernel-based extreme learning machine for state of charge estimation. Ionics 2023;29:1863–72. http://dx.doi.org/10.1007/s11581-023-04903-5.

3.2 Extreme Learning Machine evaluated on M-series System on a Chip processors from Apple

Publication:

K. Struniawski, A. Konopka, and R. Kozera, *"Exploring Apple Silicon's potential from simulation and optimization perspective"*, in Computational Science – ICCS 2024. 24th International Conference, Malaga, Spain, July 2–4, 2024, Proceedings, Part V, 2024, vol. 14836, p. 35–42. doi: 10.1007/978-3-031-63775-9_3.

Abstract:

This study explores the performance of Apple Silicon processors in real-world research tasks, with a specific focus on optimization and Machine Learning applications. Diverging from conventional benchmarks, various algorithms across fundamental datasets have been assessed using diverse hardware configurations, including Apple's M1 and M2 processors, NVIDIA RTX 3090 GPU and a mid-range laptop. The M2 demonstrates competitiveness in tasks such as BreastCancer, liver and yeast classification, establishing it as a suitable platform for practical applications. Conversely, the dedicated GPU outperformed M1 and M2 on the eyestate1 dataset, underscoring its superiority in handling more complex tasks, albeit at the expense of substantial power consumption. With the technology advances, Apple Silicon emerges as a compelling choice for real-world applications, warranting further exploration and research in chip development. This study underscores the critical role of device specifications in evaluating Machine Learning algorithms.

Exploring Apple Silicon's Potential: A Simulation and Optimization Perspective

Karol Struniawski[1][0000−0002−4574−2986], Ryszard Kozera[1,2][0000−0002−2907−8632] and Aleksandra Konopka[1][0000−0003−1730−5866]

[1] Institute of Information Technology,
Warsaw University of Life Sciences - SGGW,
ul. Nowoursynowska 159, 02-776 Warsaw, Poland
{karol_struniawski, ryszard_kozera, aleksandra_konopka}@sggw.edu.pl
[2] School of Physics, Mathematics and Computing,
The University of Western Australia,
35 Stirling Highway, Crawley, WA 6009, Perth, Australia
ryszard.kozera@uwa.edu.au

Abstract. This study explores the performance of Apple Silicon processors in real-world research tasks, with a specific focus on optimization and machine learning applications. Diverging from conventional benchmarks, various algorithms across fundamental datasets have been assessed using diverse hardware configurations, including Apple's M1 and M2 processors, NVIDIA RTX 3090 GPU and a mid-range laptop. The M2 demonstrates competitiveness in tasks such as *breast cancer, liver* and *yeast*, establishing it as a suitable platform for practical applications. Conversely, the dedicated GPU outperformed M1 and M2 on the *eyestate1* dataset, underscoring its superiority in handling more complex tasks, albeit at the expense of substantial power consumption. With the technology advances, Apple Silicon emerges as a compelling choice for real-world applications, warranting further exploration and research in chip development. This study underscores the critical role of device specifications in evaluating machine learning algorithms.

Keywords: Machine Learning, Optimization, Simulation, Apple Silicon, Extreme Learning Machine

1 Introduction

In November 2020, Apple introduced a new line of processors, starting with the M1 chip that adopts a System on a Chip (SoC) design with unified memory. The M1 processor, built using 5nm process technology and containing 16 billion transistors, also integrated the task specific modules like Apple Neural Engine. Over subsequent years, Apple released upgraded versions like the M1 Pro/M1 Max in 2021, Ultra in 2022, M2 in 2022, M2 Pro/Max/Ultra in 2023, and M3, M3 Pro/Max in 2023, all promising improved performance. The Apple M1 SoC is a highly integrated processor unit that includes all of the necessary components for a fully working computer while consuming less power in general, making it

available to customers in the markets without losing performance. The technology innovations in this field have promising future and grant more investigation and research toward the development of the chips [10, 14].

In addition to noteworthy features such as prolonged battery life and fanless design in MacBook Air models, contributing to their quiet and portable nature, these devices have piqued the interest of researchers due to their potential applications in scientific endeavors. More specifically, researchers can harness the computational capabilities of Apple Silicon for tasks like optimization and Machine Learning (ML) calculations [3]. This study aims to evaluate the practicality and effectiveness of Apple Silicon-powered devices in tasks commonly undertaken by researchers, with a focus on performing calculations.

The primary objective in this study is to evaluate various fundamental optimization and ML algorithms across diverse datasets. The notable gap in the existing literature is detected, where performance assessments are conducted on a single dataset [9] or are limited to a single ML method [8]. This paper is an extension of the work by Kasperek et al., who suggested the possibility of further expanding their research to CUDA-enabled devices. In this study, a CUDA device, the RTX 3090, was utilized.

2 Apple Silicon Overview

The Apple Silicon processors, such as the M1, utilize a Unified Memory Architecture (UMA) that allows for shared memory access across different modules of the SoC [6]. This means that the RAM is a single pool of memory that all parts of the processor can access, enabling the GPU to utilize more system memory while other parts of the SoC ramp down, without the need to shuttle data between different memory spaces [9]. In contrast, traditional CPU devices have separate memory spaces for the GPU and CPU, requiring data movement between these spaces, which can be inefficient. The benefits of UMA are particularly evident in the context of ML tasks, where the Apple Silicon chip offers hardware acceleration support, making it a tempting option for researchers. Additionally, the use of Unified Memory has been found to be beneficial when only a small random portion of data is accessed for a set of benchmarks, highlighting its efficiency [13]. In the realm of SoC architectures, many-core architectures with shared memory are preferred for flexible and programmable solutions in computationally intensive application domains, including ML and embedded processing [11].

The MacOS operating system leverages the concept of shared memory to enhance performance by expanding UMA with swap memory, albeit with a trade-off in effectiveness [12]. This approach allows a more flexible allocation of memory resources, particularly in the context of Apple Silicon devices, where different RAM sizes are available. Therefore, comparing the performance of devices without considering the RAM utilization may lead to incorrect conclusions.

The low-energy SoC chip offers clear advantages, notably in terms of extended battery life and optimal performance per watt. Significantly, the operational efficiency of Apple Silicon devices remains consistent whether operating on battery

power or when connected to an external power source, a capability not commonly observed in conventional computing systems.

These advantages become even more pronounced given the escalating energy prices in Europe following the aftermath of the conflict in Ukraine [2]. For instance, the M1-powered Mac mini demonstrates an average power consumption ranging from 10W to 31W [5]. In contrast, a PC-class device equipped with an AMD R9 or Intel i9 CPU and dedicated GPU like the NVIDIA RTX 3090 can consume up to 800W at peak performance (calculated based on the cumulative peak power consumption of individual PC components as per manufacturer specifications). The significance of power consumption is underscored by the current global scenario, where electricity demand is outpacing the growth of renewable sources [7]. Highlighting this, the Cinebench R23 Single Package Power Efficiency metric reflects favorably on the SoC, registering 297 Points per Watt. In comparison, competitors such as the Ryzen 5 5600U score 90.8 and the Intel i5-1240P scores 64 points [1]. This underlines the efficiency and energy-conscious performance of the low-energy SoC chip in a landscape where power consumption considerations are paramount.

3 Methodology

In preceding experiments that compared the NVIDIA V100 and A100 GPUs with the M1 and M1 Ultra, the obtained results were promising, showcasing the superior performance of Apple Silicon over both GPUs [9]. While the aforementioned GPUs produced impressive results, they do not represent the pinnacle of current GPU capabilities, with the NVIDIA V100 providing 14.13 TFLOPS Float32 precision to the RTX 3090's 35.58. To comprehensively assess the performance of selected ML classifiers across diverse hardware platforms and data types the six benchmark datasets are employed [4], wherein the number of samples, features and classes for each task is specified (see Tab. 1). The objective is to measure the execution time of each classifier on three distinct hardware platforms: Apple's M1 with 8GB RAM and M2 with 16GB RAM, a high-performance NVIDIA RTX 3090 GPU with 24GB memory and a mid-range laptop configuration featuring an Intel Core i5 11500h processor and an NVIDIA RTX 3050ti graphics card. The intended experiment also aimed to compare the performance of mobile devices (M1/M2) with an i5-powered laptop. Surprisingly, the unplugged i5 showed four times longer performance on average compared to the plugged-in scenario. On the other hand, the M1/M2 devices maintained consistent computational power on battery. Regrettably, the i5's limited battery life led to the decision to forgo the experiment before completion.

This approach ensures a comprehensive evaluation that extends beyond the previously explored GPUs, providing insights into the real-world performance of the classifiers across a spectrum of hardware configurations. A variety of ML methods, including Extreme Learning Machine (ELM), k Neighbors Classifier (kNN), Multi-Layer Perceptron (MLP), Random Forest (RF) and Support Vector Machine (SVM), are employed for the datasets.

	BreastCancer	eyestate1	liver	musk	waveform	yeast
Samples	699	762	345	1682	500	150
Features	9	14	6	166	21	8
Classes	2	3	4	2	10	10

Table 1. Details of datasets utilized in performance evaluations.

4 Experimental Results

All experiments were conducted in Python 3.11 using Tensorflow 2.15, scikit-learn 1.4.0 and numpy 1.26.3 on MacOS 14.2.1 or Windows 11. The outcomes, illustrated in Figures 2-6, represent the time taken for training and testing during 10 times repeated 10-fold cross-validation, ensuring result significance by mitigating odd observations.

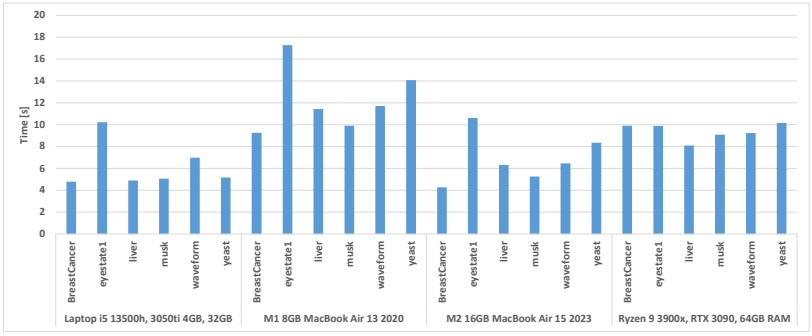


Fig. 1. Extreme Learning Machine with 100 neurons in hidden layer time of execution.

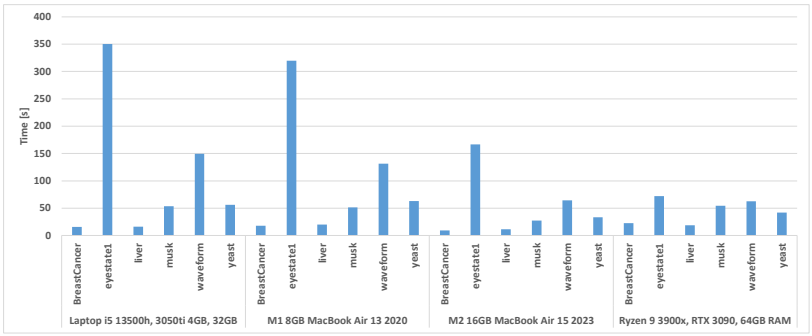


Fig. 2. Extreme Learning Machine with 1000 neurons in hidden layer time of execution.

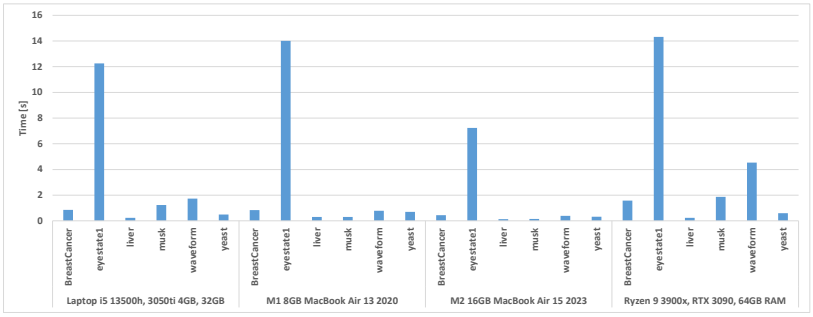


Fig. 3. K Neighbours classifier time of execution.

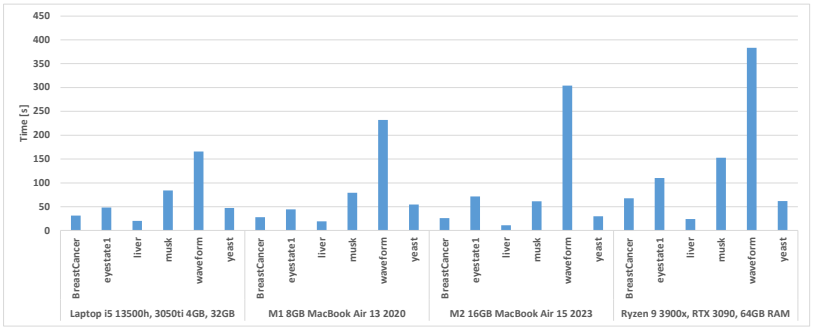


Fig. 4. Multi Layer Perceptron time of execution.

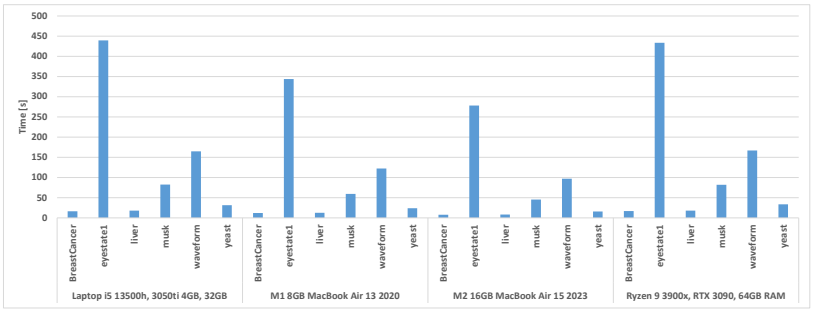


Fig. 5. Random Forest time of execution.

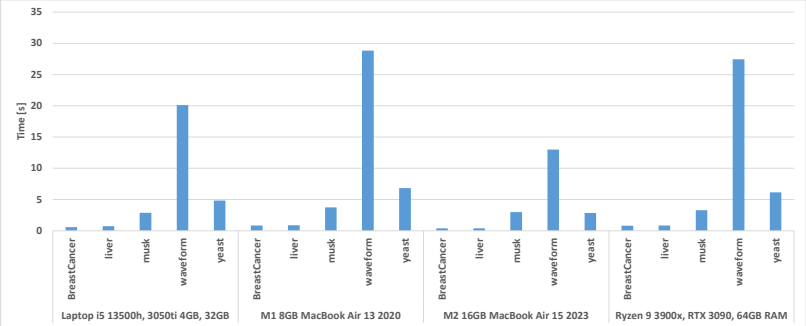


Fig. 6. Support Vector Classifier time of execution.

Device	*bc*	*eyestate1*	*liver*	*musk*	*waveform*	*yeast*	sum
Laptop i5 11500h, 3050ti, 32GB	657	16210	670	2516	46898	3360	70311
M1 8GB MacBook Air 13 2020	661	69000	700	17777	38621	3034	129792
M2 16GB MacBook Air 15 2023	331	38477	342	3055	22585	1582	66372
Ryzen 9 3900x, RTX 3090, 64GB	401	6058	376	1170	17816	1633	27454

Table 2. Performance Evaluation Results, where *bc* states as *BreastCancer*.

5 Discussion

In the case of ELM with 100 hidden layer units, subtle differences emerge, with the most notable discrepancy found in the execution time on the M1 8GB device, which is approximately twice as long as the M2 16GB counterpart (see Tab. 1). Surprisingly, the i5 laptop yields comparable results to the M2 16GB. Notably, the performance of the RTX 3090 is unexpectedly inferior to the M2 on each dataset. The differences in results across various datasets for the RTX 3090 are minimal, indicating that the GPU's memory allocation necessitated longer processing time. Despite this, the RTX 3090's rapid CUDA cores and ample 24GB memory mitigate the dataset size impact for this classifier. The *eyestate1* dataset requires the most time for processing by the classifier, with both the i5 and M1 machines struggling for ELM with 1000 neurons (see Tab. 2). Specifically, the M1 requires times longer than the M2 and the M2 takes twice as long to execute compared to the RTX 3090. Conversely, for the other datasets, the differences between M2 and the RTX 3090 are less pronounced and appear comparable.

Moving to kNN, M2 emerges as the fastest across all devices, surpassing the RTX 3090 by a few times for the *waveform* dataset. Inexplicably, the RTX 3090 delivers suboptimal results despite having updated drivers and configurations, consistent across repeated experiments (see Tab. 3). Similar patterns are observed with MLP with topology (10,10), where the RTX 3090 consistently produces the worst results, yet for the *waveform* dataset, the i5 device is the fastest, followed by the M1 and then the M2 (see Tab. 4).

In the context of RF, the RTX 3090 once again yields subpar results, while the M2 proves to be the fastest (see Tab. 5). Similar trends persist for the SVM method, with the RTX 3090 delivering suboptimal results and the M2 demonstrating the fastest performance (see Tab 6).

Considering the real-world scenario where all classifiers run on a given dataset, the aim is to compare the models' overall performance. Combining the total time required for a device to run all classifiers on different datasets, along with the ultimate sum of running all classifiers on all datasets, reveals interesting insights (see Tab. 4). The RTX 3090 emerges as the leader with a combined time of 27454 seconds, whereas the M2 is twice as slow. The i5 device demonstrates comparable performance to the M2, while the M1 lags as the slowest due to memory-intensive tasks. A closer examination highlights the substantial impact of the *eyestate1* dataset, where the RTX 3090 outperforms the M2 sixfold, showcasing the dedicated GPU's potential for more complex datasets. Conversely, M2 excels in tasks such as *BreastCancer*, *liver* and *yeast*, underscoring its competitive edge in certain scenarios against an 800W machine with a 30W device.

6 Conclusion

In conclusion, the results reveal nuanced variations in the performance of ML classifiers across diverse datasets and hardware configurations. The ELM with 100 hidden layer units showcases subtle differences, with notable disparities in execution times between devices. ELM with 1000 neurons introduces new dynamics, impacting performance across datasets.

In specific algorithms like kNN, MLP, RF and SVM, the Apple M2 processor consistently demonstrates promising performance compared to the Nvidia RTX 3090 GPU, highlighting the efficacy of Apple Silicon in real-world applications, especially taking into account the performance per watt power.

These findings underscore the importance of considering device specifications and configurations when assessing the practicality and effectiveness of ML algorithms. The competitive edge of Apple Silicon, particularly the M2 processor, is evident in various scenarios, showcasing its potential for tasks such as BreastCancer, liver and yeast, even against higher-power GPU counterparts.

References

1. Cinebench scores, https://nanoreview.net/en/cpu-list/cinebench-scores
2. Ari, A., et al.: Surging energy prices in europe in the aftermath of the war: How to support the vulnerable and speed up the transition away from fossil fuels. IMF Working Papers **2022**(152), A001 (2022). https://doi.org/10.5089/9798400214592.001.A001
3. Dalakoti, V., Chakraborty, D.: Apple M1 chip vs Intel (X86). EPRA International Journal of Research and Development (IJRD) **7**(5), 207–211 (2022)
4. Dheeru, D., Karra Taniskidou, E.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml

5. Frumusanu, A.: The 2020 mac mini unleashed: Putting apple silicon m1 to the test. AnandTech (2020), https://www.anandtech.com/show/16252/mac-mini-apple-m1-tested
6. Hart, C.: Cdram in a unified memory architecture. In: Proc. COMPCON. pp. 261–266 (1994). https://doi.org/10.1109/CMPCON.1994.282913
7. IEA: Electricity market report - july 2021 (2021), https://www.iea.org/reports/electricity-market-report-july-2021
8. Kasperek, D., Podpora, M., Kawala-Sterniuk, A.: Comparison of the usability of Apple M1 processors for various Machine Learning tasks. Sensors **22**(20) (2022). https://doi.org/10.3390/s22208005, https://www.mdpi.com/1424-8220/22/20/8005
9. Kenyon, C., Capano, C.: Apple silicon performance in scientific computing (2022)
10. Liao, X., Li, B., Li, J.: Impacts of Apple's M1 SoC on the Technology Industry. In: Proceedings of the 2022 7th International Conference on Financial Innovation and Economic Development (ICFIED 2022). pp. 355–360. Atlantis Press (2022). https://doi.org/10.2991/aebmr.k.220307.056
11. Luan, H., Gatherer, A.: Combinatorics and geometry for the many-ported, distributed and shared memory architecture. In: 2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS). pp. 1–6 (2020). https://doi.org/10.1109/NOCS50636.2020.9241708
12. Richard, G.G., Case, A.: In lieu of swap: Analyzing compressed ram in mac os x and linux. Digi. Invest. **11**, S3–S12 (2014). https://doi.org/https://doi.org/10.1016/j.diin.2014.05.011
13. Xu, H., Lin, P.H., Emani, M., Hu, L., Liao, C.: Xunified: A framework for guiding optimal use of gpu unified memory. IEEE Access **10**, 82614–82625 (2022). https://doi.org/10.1109/ACCESS.2022.3196008
14. Zhang, Z.: Analysis of the advantages of the M1 CPU and its impact on the future development of Apple. In: 2021 2nd International Conference on Big Data and Artificial Intelligence and Software Engineering (ICBASE). pp. 732–735 (2021). https://doi.org/10.1109/ICBASE53849.2021.00143

3.3 ELM optimization

3.3.1 Performance of selected nature-inspired metaheuristic algorithms used for Extreme Learning Machine

Publication:

K. Struniawski, R. Kozera, and A. Konopka, *"Performance of Selected Nature-Inspired Metaheuristic Algorithms Used for Extreme Learning Machine"*, in Computational Science – ICCS 2023. 23rd International Conference, Prague, Czechia, July 3–5, 2023, Proceedings, Part III, 2023, vol. 10475, p. 498–512. doi: 10.1007/978-3-031-36024-4_38.

Abstract: This work presents a research on Nature Inspired Metaheuristic Algorithms (MA) used as optimizers in training process of Machine Learning method called Extreme Learning Machine (ELM). We tested 19 MA optimizers measuring their performance directly on sample datasets. The impact of input parameters such as number of hidden layer units, optimization stopping conditions and population size on the accuracy results, training and prediction time is evaluated here. Significant differences in performance of applied methods and their parameters' values are detected. The most meaningful outcome of this paper shows that an increase of the number of MA iterations does not yield significant boost in accuracy with a huge increase in training time. Indeed a cap on number of MA iterations ranging from 1 to 5 is sufficient for analyzed machine learning tasks. In our research the best results are obtained for population size ranging between 50 and 100. Hybridized ELM outperforms classical implementation of ELM as higher accuracy is reached for the same number of neurons.

Performance of selected Nature-Inspired Metaheuristic Algorithms used for Extreme Learning Machine

Karol Struniawski[1][0000−0002−4574−2986], Ryszard Kozera[1,2][0000−0002−2907−8632] and Aleksandra Konopka[1][0000−0003−1730−5866]

[1] Institute of Information Technology,
Warsaw University of Life Sciences - SGGW,
ul. Nowoursynowska 159, 02-776 Warsaw, Poland
{karol_struniawski, ryszard_kozera, aleksandra_konopka}@sggw.edu.pl
[2] School of Physics, Mathematics and Computing,
The University of Western Australia,
35 Stirling Highway, Crawley, WA 6009, Perth, Australia
ryszard.kozera@uwa.edu.au

Abstract. This work presents a research on Nature Inspired Metaheuristic Algorithms (MA) used as optimizers in training process of Machine Learning method called Extreme Learning Machine (ELM). We tested 19 MA optimizers measuring their performance directly on sample datasets. The impact of input parameters such as number of hidden layer units, optimization stopping conditions and population size on the accuracy results, training and prediction time is evaluated here. Significant differences in performance of applied methods and their parameters' values are detected. The most meaningful outcome of this paper shows that an increase of the number of MA iterations does not yield significant boost in accuracy with a huge increase in training time. Indeed a cap on number of MA iterations ranging from 1 to 5 is sufficient for analyzed machine learning tasks. In our research the best results are obtained for population size ranging between 50 and 100. Hybridized ELM outperforms classical implementation of ELM as higher accuracy is reached for the same number of neurons.

Keywords: Computational Optimization · Metaheuristic Algorithms · Bio-inspired computing · Extreme Learning Machine · Machine Learning

1 Introduction

Mathematical optimization algorithms play a vital role in many contemporary technology applications such as e.g. GPS or IT banking sector tools. In addition, the optimization driven behavior is also prevalent within all living organisms commonly relying on it while e.g. hunting or trying to move more efficiently. In fact, searching for the efficiency in nature can be a matter of life and death. Among all the latter is physically demonstrated by the reproduction capabilities.

Thus, organisms that perform life activities more efficiently are better adopted to the environment and are more likely to pass these abilities to their offsprings by genes according to the Darwin's Theory [7].

Scientists attempt to describe "optimized" activities of organisms in terms of mathematical modeling [27] commonly called Metaheuristic Algorithms (MA). The combination of bio-inspired optimization algorithms with machine learning models may improve their performance [30]. Here one of such approaches is called Extreme Learning Machine (ELM) - the Machine Learning method with growing popularity since its formulation in 2004 [14].

Recently, a hybrid MA-ELM that combines ELM with Metaheuristic Algorithms (MA) is proposed and evaluated in practical applications. In doing so, Chia et al. [8] used particle swarm (PSO), moth-flame (MFO) and whale optimization algorithm (WOA). In other practical related context, Wu et al. [30] applied genetic algorithm (GA), ant colony optimization (ACO), cuckoo search algorithm (CSA) and flower pollination algorithm (FPA). The above research demonstrates superiority of hybridized ELM over the regular one. Nevertheless, most of the works in this topic deal with the practical applications of these methods. There is a shortage in literature on comprehensive comparison of metaheuristic algorithms used in ELMs. In this paper we evaluate hybrid ELM on MNIST handwritten and Wine Quality White datasets [9] for different MA. The comparison analysis for a separate set of parameters to investigate their impact on attained accuracy and registered computational time is also performed for each examined algorithm. The experiment is carried out on a single machine in MATLAB R2021b, Ryzen 9 3900X CPU, 64GB RAM, GTX 1660TI GPU.

2 Extreme Learning Machine

Extreme Learning Machine (ELM) is a dense feed-forward neural network classifier and regressor introduced by Huang et al. in 2004 [14]. The network's topology consists of input layer, a single hidden-layer and an output layer of neurons. The numbers of selected neurons in input and output layer depends on the task characteristics. The number of hidden layer units requires an empirical determination as a consequence of the theoretical method scarcity permitting to determine upfront its optimal numbers controlling the topology of the ELM.

2.1 Classification

Input data regarding supervised classification task with N observations can be described as pairs of values $\{(x_i, t_i)\}_{i=1}^{N}$, where x_i is i-th vector of d features and t_i is i-th label of class to which selected x_i belongs. Here, $t_i = 0, \ldots, M-1$, where M is the amount of distinctive classes in the classification task in question. Note here that for multiclass classification (when object belongs to more than one class) t_i is a vector. Based on the latter, matrix $X = (x_1, x_2, \ldots, x_N) \in \mathbb{M}_{d \times N}(\mathbb{R})$

is formed, where $x_i \in \mathbb{R}^d$ with vector $T = \{t_i\}_{i=1}^N$:

$$X = \begin{bmatrix} x_{11} \dots x_{1N} \\ \vdots \quad \ddots \quad \vdots \\ x_{d1} \dots x_{dN} \end{bmatrix} \quad \text{and} \quad T = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}.$$

The ELM input layer comprises of d neurons and its output layer consists of units number equal to M. As an output of the network, the corresponding N values $\{y_i\}_{i=1}^N$ are calculated forming the matrix $Y = (y_1, y_2, \dots, y_N) \in \mathbb{M}_{N\times M}(\mathbb{R})$, where $y_i \in \mathbb{R}^M$. The recognition of a given input x_i is performed based on extracting the maximal value of y_i observed on the p-th index which assigns x_i to the p-th class. Thus, matrix Y is actually reformatted as N values $\{y_i\}_{i=1}^N$, where $y_i = [0, \dots, \overset{p}{1}, \dots, 0]$. Subsequently, one has to properly format T in order to facilitate comparison with Y. In the next step $1 - of - K$ scheme is applied to the vector T - see [6]. Such procedure is designed to reformat t_i as $\{t_{ij}\}_{j=0}^M$ that yields all values set to zero except one element at s-th index that in turn is set to one. Consequently, t_i can be written as $t_i = [0, \dots, \overset{s}{1}, \dots, 0]$, where s-th element indicates that i-th input vector of X affiliates to the s-th class. Correct classification is observed if and only if $s = p$ for a given input vector x_i.

Let L be a number of neurons in hidden layer that is chosen a priori. Weights between input and hidden layer determine the matrix $W \in \mathbb{M}_{d\times L}(\mathbb{R})$, where w_{ij} represent the weights associated with the connection of i-th input layer neuron with j-th in hidden layer (see left equation (1)). Bias connections are represented by a vector $b = \{b_i\}_{i=1}^N$. In learning process of ELM coefficients of W and b are computed using uniform distribution function $U(-1, 1)$. The outputs of hidden layer neurons are stored in matrix $H \in \mathbb{M}_{N\times L}(\mathbb{R})$ (see right equation (1)):

$$W = \begin{bmatrix} w_{11} \dots w_{1L} \\ \vdots \quad \ddots \quad \vdots \\ w_{d1} \dots w_{dL} \end{bmatrix}, H = \begin{bmatrix} f(\sum_{i=1}^d x_{i1}w_{i1} + b_1) & \dots & f(\sum_{i=1}^d x_{i1}w_{iL} + b_1) \\ \vdots & \ddots & \vdots \\ f(\sum_{i=1}^d x_{iN}w_{i1} + b_N) & \dots & f\sum_{i=1}^d x_{iN}w_{iL} + b_N) \end{bmatrix}. \tag{1}$$

The activation function $f : \mathbb{R} \to \mathbb{R}$ represents in our investigation a sigmoid function $f(x) = f_\alpha(x) = \frac{1}{1+e^{-\alpha x}}$, with $\alpha = 1$. The weights β between hidden and output layer can be computed upon solving the following equation $Y = H\beta$. The system cannot be directly solved since H with probability equal to 1 is irreversible and $||H\beta - Y|| = 0$ (see Huang et al. [14]). We estimate β as a minimizer of mean residual square error:

$$\hat{\beta} = \underset{\beta}{argmin} \, \|H\beta - T\|^2 = H^\dagger T, \tag{2}$$

where $H^\dagger$ defines a Moore-Penrose generalized inverse of H [26]. The Pseudo-inverse of matrix $H^\dagger$ is uniquely determined and in the case of a non-singular matrix H it coincides with an ordinary inverse i.e. $H^\dagger = H^{-1}$. The matrix $H^\dagger$ gives solution $\hat{\beta}$ so that $H\hat{\beta}$ is close to Y in terms of mean square error (MSE).

Assigning random values to weights and bias between input and hidden ELM layer makes the network not susceptible to overtraining. Most importantly, the computed solution $\hat{\beta}$ is a global minimizer of (2). The latter contrasts with Multi-Layer Perceptron (MLP) supervised training procedure. Indeed Backpropagation Algorithm finds generically only a local minimizer of the given network's loss function that measures how well the neural network classifies the training data [12]. In addition, the optimal value of $\hat{\beta}$ is found here upon performing a non-iterative procedure in (2). The learning speed of ELM can be thousands times faster than other methods like MLP (see [15]).

3 Genetic Extreme Learning Machine

The original concept of ELM relies on selecting weights between input and hidden layer together with bias values as randomly generated. This principle has a remarkable advantage in terms of computational efficiency [15]. Still such randomness in weights generation in ELM can lead to the unstable performance [4]. The idea here is to somehow estimate weights and bias values in order to maximize the accuracy and stability of the model. A possible remedy to this problem is to combine the Genetic Algorithm (GA) (see [13]) with ELM to form the so-called hybridized Genetic Extreme Learning Machine (GELM). GAs are created as a computational representation of Darwinian evolution theories to search for the optimal solution of global non-linear optimization task by simulating the process of biological natural selection concept [16]. Our hope is that reflecting the natural processes of selection, crossover and mutation the fittest individuals are selected for reproduction that will provide better offspring in terms of improving an appropriate fitness evaluation function [5].

4 Nature-Inspired Metaheuristic Algorithms

In general, the constrained optimization problem can be formulated in terms of minimizing some objective function: $minimize f(x)$ with $x = (x_1, \ldots, x_n)$ admitted to fulfill either some equality(ies) and/or inequality(ies) [31]. All modern nature-inspired algorithms are called Metaheuristic Algorithms [17]. Up to now there is no commonly accepted definition of MA, but one can outline the following selected principles of MA adopted in the literature (see [27, 31, 24]): a strategy that the main aim is to guide the search process avoiding the disadvantages of iterative improvement allowing the local search to escape from local optima; starting to find solutions in more intelligent way than just providing random initial solutions; dealing with randomness in an biased form incorporating search experience (in a form of memory) to guide the search; in the simulation stage considered as a set of assumptions about the natural environment.

The search strategies of different MA are highly dependent on the philosophy of the metaheuristic itself. In this paper, as a comparison of the MA applied in ELM learning process, we use methods simulating behaviors of living organisms in terms of the following optimization processes: Artificial Ecosystem-based

Optimization (AEO) [35], Artificial Hummingbird Algorithm (AHA) [34], Artificial Rabbits Optimization (ARO) [29], African Vultures Optimization Algorithm (AVOA) [2], Coyote Optimization Algorithm (COA) [25], Dandelion Optimizer (DO) [32], Fast Cuckoo Search (FCS) [23], Gorilla Troops Optimizer (GTO) [3], Grey Wolf Optimizer (GWO) [20], Hybrid Grey Wolf and Cuckoo Search Optimization Algorithm (GWO-CS) [11], Improved Grey Wolf Optimizer (I-GWO) [21], Leader Harris Hawks Optimization (LHHO) [22], Mountain Gazelle Optimizer (MGO) [1], Manta Ray Foraging Optimization (MRFO) [36], Northern Goshawk Optimization (NGO) [10], Pelican Optimization Algorithm (POA) [28], Hybrid Particle Swarm Optimization and Gravitational Search Algorithm (PSOGSA) [19], Sea-horse Optimizer (SHO) [33] and lastly Salp Swarm Algorithm (SSA) [18].

5 Experiments and Results

The metaheuristic algorithms (briefly outlined in the previous section) used in this work have common prerequisites. In particular, from now on, the term MA directly refers to the algorithms exclusively used in this paper (see Section 4).

MA define a concept of population as a set S of S_n candidate solutions, where s_i, $i = 1, \ldots, S_n$ is a solution vector called also an individual and implement the concept of intelligent iterative ransacking search space taking as an input dimension of the vector $S_d = dim(s_i)$, number of population S_n and constraints applied to s_i. A termination condition for the algorithm and appropriate fitness function must be determined. As MA fall into iterative methods, in k-th iteration the set S^k called generation is produced with $s_i^k \in S^k$ representing generation's individual, where $k = 1, \ldots, k_n$. The output of MA $s^{min} = s_i^{k_n}$ yields a minimal value of a given fitness function in the last generation of the algorithm.

To integrate MA with ELM we first need to specify input parameters for MA. Analogously to GELM our aim is to evaluate optimal values of weights between input and hidden layer including bias. In fact, the output of the MA is a vector $s^{min} \in \mathbb{R}^{S_d}$, where $S_d = dN + N$. As a consequence we can reformat s^{min} properly constructing W and b:

$$W = \begin{bmatrix} s_{11}^{min} & \ldots & s_{1N}^{min} \\ \vdots & \ddots & \vdots \\ s_{d1}^{min} & \ldots & s_{dN}^{min} \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} s_{dN+1}^{min} \\ \vdots \\ s_{dN+N}^{min} \end{bmatrix}.$$

Vector s^{min} forms the final, optimal value of $W^{s^{min}}$ and $b^{s^{min}}$. Fitness function g is prepared based on the response of the ELM network represented as Y_i^k for a given s_i^k (forming W_i^k and b_i^k) compared to the expected results T, $g(X, W_i^k, b_i^k, T) = \frac{1}{N}\sum_{j=1}^{N}(Y_{ij}^k - T_j)^2$, where $Y_i^k = H\beta$, $\beta = H^{\dagger}T$ and $H = f(X^T W_i^k + b_i^k)$ (see also (1)). The inequality constraints $-1 < s_{ij} < 1$ (with $j \in [1, \ldots S_d]$ for each $i \in [1, \ldots, S_n]$) enforce $s_i \in [-1; 1]^{S_d} \subseteq \mathbb{R}^{S_d}$. The impact of parameters S_n and the termination condition on the algorithm performance is investigated in this research. The admitted values for S_n are equal

to 50, 100 or 200. It is noteworthy that lowering the vales of S_n led to unstable results, while higher S_n values resulted in impractically long evaluation times for our experiment. Two different approaches of selecting stopping conditions of MA are here considered. *First*, the stopping flag is activated once one of two conditions is fulfilled. More specifically, the upper limit on k iterations is a priori set (here $k_n = 10000$). In conjunction with the latter, the optimization procedure terminates once the following a posteriori condition is met $\left|g(X, W_i^k, b_i^k, T) - g(X, W_i^{k+1}, b_i^{k+1}, T)\right| < \varepsilon$ holding for longer than 200 iterations (here $\varepsilon = 0.0001$). In further presentation of calculation results the first variant of stopping condition is marked as *"Limit 0"*. *Second*, the impact of fixing ad hock an upper bound k on number of iterations is also analyzed here for $k_n = 1$, $k_n = 5$ and $k_n = 50$ that can be recognized in further considerations as *"Limit 1"*, *"Limit 5"* and *"Limit 50"*, respectively.

Another parameter taken also here into consideration is the number of neurons L in hidden layer of ELM. At this point one should mention a dilemma of evaluating results applying testing and training sets once MA is used for optimizing W and b. A core principle of the Machine Learning (ML) is to examine results returned by a selected method on data that cannot be used for training process. To enforce the latter the data is usually a priori divided into training and testing sets or alternatively one resorts to a cross-validation method [12]. Cross-validation is an iterative method that uses different portions of data to test and to train a model applying randomness. Thus, matrices W and b are optimized upon using MA on training data exclusively and cannot be specified as optimal on testing set. A similar approach should be adopted for β evaluation while computing weights between hidden and output layer of ELM. It is implicitly assumed here that dependencies for both training and testing sets are similar. Therefore the optimized W and b based on training set can equally successfully operate on testing set. The case of unbalanced number of observations obtained on training and testing sets deserves a short note. Indeed, should the latter occurs, the matrices W and b generated by MA on training set cannot be directly applied to estimate Y on testing set. In ML there exists an implicit assumption that the testing set should be essentially smaller than a training one. Typically, the proportion of observations abides from 9:1 to 7:3 ratio. Consequently, s^{min} is too large to be properly re-formatted to W and b which can still act on testing set. Assuming testing set contains N^{test} observations we solve this problem by taking $k = N^{test} \times d$ first elements of s^{min} transforming them into matrix $W^{s^{min}}_{N^{test} \times d}$ and last N^{test} elements of s^{min} creating vector $b^{s^{min}}$:

$$W^{s^{min}} = \begin{bmatrix} s_1^{min} & \dots & s_N^{min} \\ \vdots & \ddots & \vdots \\ s_{k-N}^{min} & \dots & s_k^{min} \end{bmatrix} \quad \text{and} \quad b^{s^{min}} = \begin{bmatrix} s_{S_d - N^{test}}^{min} \\ \vdots \\ s_{S_d}^{min} \end{bmatrix}.$$

The entire calculation process is presented in the flowchart (see Fig. 5).

First, we evaluate the model in question for a different number of neurons L in hidden layer, population size S_n and MA termination condition taken as *"Limit 0"*. Unfortunately, even for $S_n = 50$ and $L = 100$ computation time for

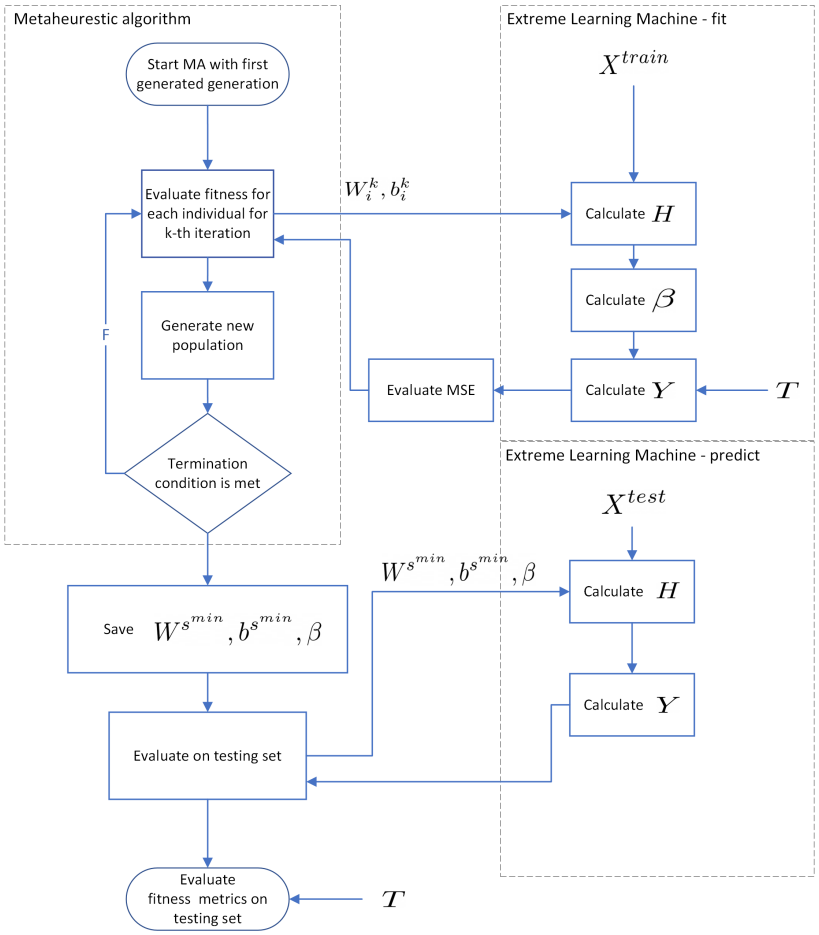


Fig. 1. Flowchart of MA-ELM training and testing process.

most of the methods exceeded a few hours. Then, a full comparison to the other Limits is impossible. Therefore we discarded *"Limit 0"* calculations and leave it for a future investigation. For our research two exemplary datasets are used. The first set is called MNIST handwritten digits. The dataset contains 60000 training and 10000 testing samples that are handwritten digits saved as greyscale images of size 28×28 pixels. Thus, input vectors' size is 784 after flatten operation is applied to the original image transforming image to the row by row vector. The dataset is a typical example of classification task where accuracy of the classifier is evaluated. The second set is called Wine Quality White which is composed of features describing chemical and physical parameters of white wine i.e. fixed acidity, volatile acidity, pH etc. Our task is to assign to a given wine

sample the quality measure ranging from 0 to 10. Here classification performance is measured with MSE as we recognize differences between inappropriate class assignments i.e. attaching a wine which is truly categorized as 0 to class 9 is a graver mistake than assigning this wine to class 1. The dataset does not contain separate training and testing subsets and because of that to obtain the most meaningful results a 20% cross-validation method is applied that is repeated 50 times to properly estimate a statistically significant value of MSE and to discard randomness influence on the final results.

ACC [%]	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
AEO	78.31	79.04	82.05	85.19	85.06	86.14	84.95	78.74	88.89	88.00
AHA	77.08	81.02	83.80	84.17	86.77	85.22	87.89	88.29	85.29	89.61
ARO	73.74	83.43	81.06	83.17	83.76	85.96	88.27	89.20	88.08	88.02
AVOA	68.50	82.11	84.06	81.66	84.46	85.78	86.55	88.85	89.84	89.60
COA	72.75	78.67	83.83	84.62	86.36	86.04	86.28	89.36	88.47	90.09
DO	73.48	81.41	83.61	80.31	87.55	81.95	89.22	86.20	87.54	85.98
FCS	72.89	80.43	80.94	86.15	85.77	87.17	86.76	89.25	88.02	88.80
GTO	78.54	81.73	78.35	84.67	75.38	88.85	84.99	80.26	83.8	84.41
GWO	69.42	74.79	81.01	83.28	83.95	88.52	87.58	85.05	88.65	90.08
GWO-CS	46.43	51.76	63.67	68.60	62.92	70.41	68.98	70.54	73.66	71.76
I-GWO	71.36	79.02	82.68	80.52	86.51	86.38	88.23	86.47	84.77	85.78
LHHO	72.48	81.02	80.57	87.39	84.42	86.93	85.53	87.43	87.88	87.84
MGO	71.59	78.65	78.06	80.13	86.27	86.93	86.93	88.36	88.42	88.92
MRFO	73.52	81.21	86.47	83.36	82.47	88.8	88.34	89.16	90.01	89.25
NGO	69.89	81.43	84.35	84.63	84.9	86.17	87.09	88.54	89.72	90.70
POA	69.81	81.06	83.16	84.73	85.89	87.31	87.83	88.09	87.36	89.60
PSOGSA	74.91	78.97	83.23	84.25	85.34	86.70	86.36	84.90	86.89	90.19
SHO	72.87	81.92	81.92	84.57	86.44	84.13	83.34	89.73	89.49	85.07
SSA	74.92	80.61	81.97	82.41	82.41	86.66	88.8	87.78	87.21	87.77

Table 1. Accuracy of ELM with selected MA applied for a given number of neurons in hidden layer, population size 50 and limit of iterations equal to 1 used directly as a classifier on MNIST handwritten digits dataset.

For MNIST similarly to the ELM (see Tab. 7) for MA-ELM we obtain better results upon increasing number of neurons (see Tab. 1). Then it was decided to fix L to the value of 1000 in further calculations of MA-ELM. In contrast, for Wine Quality White dataset the best results are observed for $L = 100$ (see Tab. 2), so this value will be fixed analyzing the dataset. Fitting process (see Tab. 3) for MNIST, $L = 1000$, $S_n = 50$ and selected limit of iterations combined with a method in question may take from a few seconds to some hours. The fastest methods turned out to be AVOA, DO, PSOGSA, SHO and SSA. In particular, the last one yields the most prominent results with 8s, 284s and 2784s fit time for limit k_n set to 1, 5 and 50, respectively. One can also notice that raising the number of iterations more or less linearly increases the resulting computation time. Differences in fitting time between various methods testify

MSE	100	200	300	400	500	600	700	800	900	1000
AEO	0.641	0.647	0.659	0.679	0.692	0.699	0.725	0.734	0.755	0.775
AHA	0.634	0.645	0.661	0.669	0.685	0.701	0.718	0.731	0.759	0.768
ARO	0.648	0.650	0.654	0.670	0.684	0.697	0.716	0.739	0.747	0.765
AVOA	0.645	0.647	0.663	0.675	0.688	0.706	0.722	0.739	0.753	0.776
COA	0.639	0.647	0.659	0.680	0.690	0.712	0.721	0.738	0.751	0.771
DO	0.653	0.652	0.661	0.681	0.692	0.708	0.725	0.743	0.754	0.768
FCS	0.647	0.650	0.659	0.677	0.694	0.707	0.727	0.738	0.762	0.761
GTO	0.650	0.650	0.659	0.661	0.674	0.719	0.712	0.723	0.740	0.756
GWO	0.648	0.647	0.661	0.678	0.690	0.709	0.719	0.741	0.757	0.774
GWO-CS	0.625	0.628	0.641	0.642	0.671	0.672	0.726	1.105	0.722	1.056
I-GWO	0.644	0.651	0.668	0.672	0.693	0.709	0.723	0.747	0.749	0.770
LHHO	0.630	0.638	0.660	0.676	0.695	0.681	0.722	0.727	0.758	0.794
MGO	0.635	0.637	0.656	0.668	0.695	0.694	0.721	0.726	0.726	0.752
MRFO	0.641	0.642	0.656	0.657	0.674	0.711	0.720	0.730	0.751	0.765
NGO	0.647	0.654	0.664	0.674	0.686	0.708	0.726	0.739	0.751	0.771
POA	0.656	0.651	0.664	0.676	0.691	0.704	0.724	0.745	0.756	0.773
PSOGSA	0.647	0.648	0.662	0.677	0.695	0.703	0.732	0.737	0.758	0.772
SHO	0.651	0.649	0.659	0.677	0.696	0.710	0.723	0.738	0.756	0.772
SSA	0.647	0.652	0.663	0.675	0.696	0.708	0.720	0.742	0.755	0.767

Table 2. MSE of Extreme Learning Machine with selected Metaheuristic Algorithms applied for a given number of neurons in hidden layer, population size 50 and limit of iterations equal to 1 used directly as a classifier on Wine Quality White dataset.

their practical applicability i.e. MGO needs fourfold more time for *"Limit 1"* to achieve comparable results with SSA. It should be emphasized here that there is no correlation between more computational time involved versus achieving better results. Indeed, a GWO method surpasses in terms of ACC the other methods that still need twice longer time to be executed. In previous section MA are defined as methods that tend to create the new generations with individuals characterized by lower fitness function value. Simultaneously, as we stated the low MSE value for a given individual being MA solution cannot be directly recognized as better in terms of accuracy upon applying on a testing set, because of the fact that in training and testing different subsets of dataset are used. Surprisingly, for many of the methods increasing number of iterations does not improve ACC (see Tab. 3). For most of them we observe a slight increase of ACC between *"Limit 1"* and *"Limit 5"*. The decrease of ACC between *"Limit 5"* and *"Limit 50"* was not expected. For some of the methods the lowest ACC is obtained for *"Limit 50"*. The classifier performance on Wine Quality White confirms results obtained on MNIST. For the majority of methods we do not observe significant changes of MSE. Setting k_n to higher value even increases MSE in the case of AHA, COA, GWO-CS, LHHO, MGO and PGOGSA. For the remaining methods change of k_n from 1 to 5 results in a slight decrease of MSE. Methods AEO, AVOA, DO, FCS, GWO, MGO, MRFO and NGO can be characterized by decreasing MSE once k_n changes from 1 to 5. In terms of

	MNIST						Wine Quality White					
	Fit time [s]			ACC [%]			Fit time [s]			MSE		
Method / k_n	1	5	50	1	5	50	1	5	50	1	5	50
AEO	237	571	5697	88.00	86.79	84.95	0.6	2.4	21.6	0.641	0.635	0.641
AHA	155	312	3989	89.61	88.98	84.62	0.3	1.7	10.7	0.634	0.646	0.631
ARO	156	309	4028	88.02	88.15	81.87	0.3	1.4	11.0	0.648	0.637	0.636
AVOA	78	258	4050	89.60	84.94	77.36	0.1	1.0	10.9	0.645	0.629	0.634
COA	278	692	7274	90.09	82.7	88.83	0.6	3.2	26.6	0.639	0.643	0.624
DO	77	273	2666	85.98	88.46	86.9	0.1	1.3	11.5	0.653	0.649	0.650
FCS	233	564	5041	88.80	86.51	77.86	0.5	2.5	20.9	0.647	0.647	0.651
GTO	103	567	5062	84.41	89.99	89.89	0.5	2.6	20.5	0.650	0.637	0.633
GWO	250	277	2705	90.08	88.57	87.78	0.2	1.1	11.8	0.648	0.648	0.653
GWO-CS	270	311	3046	71.76	72.99	87.88	0.2	1.3	13.7	0.625	0.643	0.630
I-GWO	445	608	5466	85.78	88.99	86.67	0.6	3.2	24.7	0.644	0.643	0.635
LHHO	240	896	8879	87.84	89.87	76.34	0.6	3.6	36.7	0.630	0.635	0.623
MGO	236	1218	11486	88.92	90.55	89.89	1.0	5.6	52.6	0.635	0.633	0.640
MRFO	232	566	5069	89.25	88.14	83.28	0.5	2.9	21.3	0.641	0.642	0.643
NGO	141	566	5038	90.70	91.45	87.42	0.5	2.6	21.5	0.647	0.645	0.647
POA	212	556	4974	89.60	81.25	86.32	0.5	2.8	20.6	0.656	0.647	0.638
PSOGSA	82	507	4800	90.19	85.8	90.81	0.5	2.6	24.6	0.647	0.655	0.649
SHO	22	471	4152	85.07	87.37	72.69	0.5	2.4	17.0	0.651	0.644	0.634
SSA	8	284	2784	87.77	87.71	88.90	0.2	1.5	12.8	0.647	0.647	0.643

Table 3. Fit time and accuracy of Extreme Learning Machine with selected Metaheuristic Algorithms applied for $L = 1000$ neurons in hidden layer, population size $S_n = 50$ and $k_n = 1$, 5 or 50 used directly as a classifier on MNIST handwritten digits. For $L = 100$, $S_n = 50$, $k_n = 1$, 5 or 50 on Wine Quality White dataset.

fitting time we observe more or less a linear growth of computational time when k_n is enlarged. In Tab. 4 we tested a different population size S_n. It shows that increasing S_n expands the fitting time, but to a lesser extent with exceptions like SSA method that needs almost 14× computation time for $S_n = 100$ as compared to $S_n = 50$. Such tendency is similar for SHO method. Fitting time between $S_n = 100$ and $S_n = 200$ for most of the methods expand twice. In terms of accuracy, we do not observe a significant increase when population size is enlarged. The methods that are beneficial to this increase are DO, FCS, GTO, GWO-CS, I-GWO, LHHO, MGO, MRFO and SSO. Noticeably, for DO, FCS, I-GWO, MGO and MRFO the highest accuracy is registered for $S_n = 100$ and the lowest for $S_n = 200$. The observed dependencies on MNIST are even more visible for Wine Quality dataset. The lowest MSE for all methods is obtained for $S_n = 50$ and increases substantially when $S_n = 100$ or $S_n = 200$ is used.
For standard ELM classifier applied on MNIST handwritten dataset the highest accuracy 91.41% is generated for 4000 and 5000 neurons in hidden layer (see Tab. 7). In comparison, for MA-ELM the highest ACC of 91.45% is reached for NGO metaheuristic algorithm, 1000 neurons, limit of 5 iterations, population size 50 and 91.43% ACC for MRFO with 1000 neurons, limit of 1 iteration and

	MNIST						Wine Quality White					
	Fit time [s]			ACC [%]			Fit time [s]			MSE		
Method / S_n	50	100	200	50	100	200	50	100	200	50	100	200
AEO	237	331	675	88.00	86.47	88.50	0.6	30	57	0.641	0.743	0.741
AHA	155	211	428	89.61	86.46	86.07	0.3	19	36	0.634	0.804	0.788
ARO	156	208	435	88.02	84.53	86.26	0.3	19	36	0.648	0.732	0.771
AVOA	78	104	215	89.60	88.13	88.87	0.1	9.	18	0.645	0.745	0.712
COA	278	369	785	90.09	89.42	88.25	0.6	35	68	0.639	0.717	0.744
DO	77	102	209	85.98	89.93	87.88	0.1	9	17	0.653	0.704	0.734
FCS	233	313	654	88.80	90.06	88.67	0.5	29	57	0.647	0.737	0.711
GTO	103	321	649	84.41	80.56	89.26	0.5	28	54	0.650	0.804	0.716
GWO	250	115	231	90.08	88.99	88.85	0.2	11	21	0.648	0.750	0.731
GWO-CS	270	142	315	71.76	84.23	89.93	0.2	16	34	0.625	0.733	0.759
I-GWO	445	345	721	85.78	89.06	87.23	0.6	33	65	0.644	0.747	0.736
LHHO	240	422	845	87.84	88.39	89.68	0.6	39	71	0.630	0.715	0.749
MGO	236	657	1574	88.92	89.59	87.32	1.0	69	153	0.635	0.731	0.729
MRFO	232	318	673	89.25	91.43	82.92	0.5	30	60	0.641	0.774	0.788
NGO	141	312	651	90.70	86.34	88.46	0.5	29	56	0.647	0.737	0.749
POA	212	306	628	89.60	87.06	89.96	0.5	28	53	0.656	0.732	0.717
PSOGSA	82	296	996	90.19	87.65	88.91	0.5	42	140	0.647	0.664	0.747
SHO	22	286	593	85.07	88.07	89.39	0.5	28	56	0.651	0.717	0.743
SSA	8	110	229	87.77	90.69	88.18	0.2	10	20	0.647	0.687	0.746

Table 4. Fit time and accuracy of Extreme Learning Machine with selected Metaheuristic Algorithms applied for 1000 neurons in hidden layer, limit of iterations equal to 1 and a different population size $S_n = 50$, 100 or 200 used directly as a classifier on MNIST handwritten digits dataset and 100 neurons in hidden layer, limit of iterations equal to 1 and a different population size $S_n = 50$, 100 or 200 used directly as a classifier on Wine Quality White dataset.

Method	S_n	L	k_n	**Fit MSE**	**Fit time [s]**	**Prediction time [s]**	**ACC [%]**
NGO	50	1000	5	0.059	566	0.074	91.45
MRFO	100	1000	1	0.057	318	0.082	91.43
GTO	50	900	5	0.058	471	0.081	90.88
GTO	200	900	5	0.057	2928	0.069	90.85
SSA	50	900	5	0.062	236	0.065	90.81
PSOGSA	50	1000	50	0.060	4800	0.068	90.81

Table 5. The 6 highest ACC for Extreme Learning Machine with selected Metaheuristic Algorithms used directly as a classifier on MNIST handwritten digits dataset.

population size 100 (see Tab. 5). Here we obtained comparable results of ELM and MA-ELM but for a different number of neurons. Training time, that is stated as a fit time for MA-ELM, is a lot longer than in case of typical ELM. Note here that in many practical application cases of ML a short prediction time is crucial. The time is extended when net is composed of more hidden layer units. Focusing on prediction time we should compare nets with 1000 hidden layer neurons, then

Method	S_n	L	k_n	**Fit MSE**	**Fit time [s]**	**Prediction time [s]**	**MSE**
LHHO	50	100	50	0.417	36.706	0.008	0.623
COA	50	100	50	0.414	26.659	0.008	0.624
GWO-CS	50	100	1	0.425	0.285	0.005	0.625
GWO-CS	50	200	1	0.398	0.561	0.010	0.628
AVOA	50	100	5	0.421	1.073	0.008	0.629

Table 6. The 5 lowest MSE for Extreme Learning Machine with selected Metaheuristic Algorithms used directly as a classifier on Wine Quality White dataset.

Neurons	**Fit Time [s]**	**Prediction Time [s]**	**ACC [%]**
1000	3	0.069	88.69
2000	7	0.134	90.57
3000	14	0.256	91.26
4000	26	0.268	91.41
5000	45	0.412	91.41
6000	71	0.511	91.30
7000	115	0.527	90.58
8000	165	0.707	90.83

Table 7. Extreme Learning Machine used directly as a classifier on MNIST handwritten digits dataset results.

Neurons	**Fit Time [s]**	**Prediction Time [s]**	**MSE**
100	0.004	0.0003	0.646
200	0.008	0.0014	0.652
300	0.021	0.0032	0.660
400	0.028	0.0034	0.677
500	0.025	0.0025	0.691
600	0.040	0.0033	0.710
700	0.052	0.0037	0.721
800	0.053	0.0041	0.744
900	0.105	0.0061	0.757
1000	0.130	0.0084	0.770

Table 8. Extreme Learning Machine used directly as a classifier on Wine Quality White results.

MA-ELM achieve ACC higher by 3pp (percentage points) over EML. Prediction time is highly dependent on L, then there is no difference of prediction time between MA-ELM and ELM. When we compare a similar ACC from the both methods (for MA-ELM $L = 1000$ and ELM $L = 4000$) prediction time for 1000 is 6 times shorter which can be very beneficial in models that require short classification time. Summing up the results obtained for the Wine Quality with ELM classifier applied leads to a rise of MSE when number of neurons in hidden layer increases (see Tab. 8). The lowest MSE=0.646 is generated for $L = 100$ with training time of the net equal to $0.004s$ and prediction coinciding with $0.0003s$.

According to Tab. 6 which presents the top 5 lowest MSE across all exploited parameters' values of MA-ELM the best results are produced for $S_n = 50$, $L = 100$ and $k_n = 50$ for LHHO and COA methods. The lowest observed MSE=0.623 for MA-ELM is a better result than using core ELM method for which MSE=0.646 is detected. Both classifiers for this dataset have comparable prediction time as the best results are reached for the same value of L.

6 Conclusions

In this paper the concept of hybridized ELM with MA is introduced. Subsequently, the influence of the parameters' value selection on final results is examined. More precisely, the impact on the results of the number of neurons in hidden layer of ELM, the size of the population and the stopping conditions for MA are investigated. Based on this research we conclude that higher accuracy of the hybridized ELM can be detected even for lower number of neurons in hidden layer than in typical ELM. The latter leads to a significant fall in prediction time of the model. Surprisingly, the best results assessing MA termination condition of MA are registered as a hard limit of 5 iterations for MNIST handwritten and of 50 iterations for Wine Quality White dataset. Unfortunately, we were not able to examine termination condition of MA as a limit of 10000 iterations or changes of fitness less than $\varepsilon = 0.0001$ because of the computational complexity involved. This aspect should be further investigated. The population sizes examined in our study were set to 50, 100, and 200, as lower values led to unstable results, and higher values resulted in impractically long evaluation times for our experiment. In total 19 MA methods are tested and across all SSA and MRFO stand out for their high accuracy combined with shorter training times compared to other methods. Notwithstanding, one ought to emphasize that there is no method that yields excellent results on both datasets. It is worth noting that there is no direct correlation between increased computational time and improved results. The selection of the appropriate MA algorithm for a particular task should be based on comprehensive evaluation. However, in our work, we observed that certain algorithms exhibit a high computational complexity without a significant improvement in classification accuracy. Therefore, MGO, AEO, COA, and LHHO may not be suitable for hybridized ELM and can be discarded from further consideration.

References

1. Abdollahzadeh, B., Gharehchopogh, F.S., Khodadadi, N., Mirjalili, S.: Mountain gazelle optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems. Adv. Eng. Softw. **174**, 103282 (2022). https://doi.org/10.1016/j.advengsoft.2022.103282
2. Abdollahzadeh, B., Gharehchopogh, F.S., Mirjalili, S.: African vultures optimization algorithm: a new nature-inspired metaheuristic algorithm for global optimization problems. Comput. Ind. Eng. **158**, 107408 (2021). https://doi.org/10.1016/j.cie.2021.107408

3. Abdollahzadeh, B., Soleimanian Gharehchopogh, F., Mirjalili, S.: Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. Int. J. Intell. Syst. **36**(10), 5887–5958 (2021). https://doi.org/10.1002/int.22535
4. Albadr, M.A.A., Tiun, S., AL-Dhief, F.T., Sammour, M.A.M.: Spoken language identification based on the enhanced self-adjusting extreme learning machine approach. PLOS One **13**(4), 1–27 (2018). https://doi.org/10.1371/journal.pone.0194770
5. Banzhaf, W., Francone, F.D., Keller, R.E., Nordin, P.: Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann Publishers Inc. (1998)
6. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
7. Brooks, C.M.: The nature of life and the nature of death. Journal of UOEH **5**(2), 133–145 (1996)
8. Chia, M.Y., Huang, Y.F., Koo, C.H.: Swarm-based optimization as stochastic training strategy for estimation of reference evapotranspiration using extreme learning machine. Agric. Water Manag. **243**, 106447 (2021). https://doi.org/10.1016/j.agwat.2020.106447
9. Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J.: Modeling wine preferences by data mining from physicochemical properties. Decis. Support Syst. **47**(4), 547–553 (2009). https://doi.org/10.1016/j.dss.2009.05.016
10. Dehghani, M., Hubalovsky, S., Trojovsky, P.: Northern goshawk optimization: a new swarm-based algorithm for solving optimization problems. IEEE Access **9**, 162059–162080 (2021). https://doi.org/10.1109/ACCESS.2021.3133286
11. Gupta, A.: Hybrid grey wolf and cuckoo search optimization algorithm (2023), https://www.mathworks.com/matlabcentral/fileexchange/69392-hybrid-grey-wolf-and-cuckoo-search-optimization-algorithm
12. Hassoun, M.H.: Fundamentals of Artificial Neural Networks. MIT Press, 1st edn. (1995)
13. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
14. Huang, G.B., Zhu, Q.Y., Siew, C.: Extreme learning machine: A new learning scheme of feedforward neural networks. In: IEEE Int. Conf. Neural Netw. vol. 2, pp. 985 – 990 (2004). https://doi.org/10.1109/IJCNN.2004.1380068
15. Li, H.T., Chou, C.Y., Chen, Y.T., Wang, S.H., Wu, A.Y.: Robust and lightweight ensemble extreme learning machine engine based on eigenspace domain for compressed learning. IEEE TCAS-I **66**(12), 4699–4712 (2019). https://doi.org/10.1109/TCSI.2019.2940642
16. Liu, W.C., Chung, C.E.: Enhancing the predicting accuracy of the water stage using a physical-based model and an artificial neural network-genetic algorithm in a river system. Water **6**(6), 1642–1661 (2014). https://doi.org/10.3390/w6061642
17. lreau, M., Potvin, J.Y.: Handbook of Metaheuristics. Springer Publishing Company, Incorporated, 2nd edn. (2010)
18. Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H., Mirjalili, S.M.: Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. Adv. Eng. Softw. **114**, 163–191 (2017). https://doi.org/10.1016/j.advengsoft.2017.07.002
19. Mirjalili, S., Hashim, S.Z.M.: A new hybrid PSOGSA algorithm for function optimization. In: ICCASM 2010. pp. 374–377 (2010). https://doi.org/10.1109/ICCIA.2010.6141614

20. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014). https://doi.org/10.1016/j.advengsoft.2013.12.007
21. Nadimi-Shahraki, M.H., Taghian, S., Mirjalili, S.: An improved grey wolf optimizer for solving engineering problems. Expert Syst. Appl. **166**, 113917 (2021). https://doi.org/10.1016/j.eswa.2020.113917
22. Naik, M.K., Panda, R., Wunnava, A., Jena, B., Abraham, A.: A leader harris hawks optimization for 2-D masi entropy-based multilevel image thresholding. Multimed. Tools. Appl. **80**(28), 35543–35583 (2021). https://doi.org/10.1007/s11042-020-10467-7
23. Naik, M.K., Swain, M., Panda, R., Abraham, A.: Novel square error minimization-based multilevel thresholding method for COVID-19 x-ray image analysis using fast cuckoo search. Int. J. Image Graph. (2022). https://doi.org/10.1142/s0219467824500049
24. Osman, I.H., Laporte, G.: Metaheuristics: A bibliography. Ann. Oper. Res. **63**, 511–623 (1996)
25. Pierezan, J., Dos Santos Coelho, L.: Coyote optimization algorithm: A new metaheuristic for global optimization problems. In: IEEE-CEC 2018. pp. 1–8 (2018). https://doi.org/10.1109/CEC.2018.8477769
26. Rao, C.R., Mitra, S.K.: Generalized Inverse of Matrices and its Applications. John Wiley & Sons (1971)
27. Stützle, T.: Local search algorithms for combinatorial problems - analysis, improvements, and new applications. In: DISKI (1999)
28. Trojovsky, P., Dehghani, M.: Pelican optimization algorithm: A novel nature-inspired algorithm for engineering applications. Sensors **22**(3) (2022). https://doi.org/10.3390/s22030855
29. Wang, L., Cao, Q., Zhang, Z., Mirjalili, S., Zhao, W.: Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. Eng. Appl. Artif. Intell. **114**, 105082 (2022). https://doi.org/10.1016/j.engappai.2022.105082
30. Wu, L., Zhou, H., Ma, X., Fan, J., Zhang, F.: Daily reference evapotranspiration prediction based on hybridized extreme learning machine model with bio-inspired optimization algorithms: Application in contrasting climates of China. J. Hydrol. **577**, 123960 (2019). https://doi.org/10.1016/j.jhydrol.2019.123960
31. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms. Luniver Press (2010)
32. Zhao, S., Zhang, T., Ma, S., Chen, M.: Dandelion optimizer: A nature-inspired metaheuristic algorithm for engineering applications. Eng. Appl. Artif. Intell. **114**, 105075 (2022). https://doi.org/10.1016/j.engappai.2022.105075
33. Zhao, S., Zhang, T., Ma, S., Wang, M.: Sea-horse optimizer: a novel nature-inspired meta-heuristic for global optimization problems. Appl. Intell. (2022). https://doi.org/10.1007/s10489-022-03994-3
34. Zhao, W., Wang, L., Mirjalili, S.: Artificial hummingbird algorithm: a new bio-inspired optimizer with its engineering applications. Comput. Methods Appl. Mech. Eng. **388**(1), 114194 (2022). https://doi.org/10.1016/j.cma.2021.114194
35. Zhao, W., Wang, L., Zhang, Z.: Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm. Neural. Comput. Appl. **32**, 1–43 (2020). https://doi.org/10.1007/s00521-019-04452-x
36. Zhao, W., Zhang, Z., Wang, L.: Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. Eng. Appl. Artif. Intell. **87**, 103300 (2020). https://doi.org/10.1016/j.engappai.2019.103300

3.3.2 Performance evaluation of activation functions in Extreme Learning Machine

Publication:

K. Struniawski, A. Konopka, and R. Kozera, *"Performance Evaluation of Activation Functions in Extreme Learning Machine"*, in ESANN 2023 : Proceedings, 2023, p. 351–356. doi: 10.14428/esann/2023.es2023-31.

Abstract: This study investigates the performance of 36 different activation functions applied in Extreme Learning Machine on 10 distinct datasets. Results show that Mish and Sexp activation functions exhibit outstanding generalization abilities and consistently perform well across most datasets, while other functions are more dependent on the characteristics of the task at hand. The selection of an activation function is intricately linked to the applied dataset and novel activation functions may possess superior generalization capabilities comparing to commonly employed alternatives. This study provides valuable insight for researchers and practitioners seeking to optimize Extreme Learning Machine performance for solving classification tasks.

Performance Evaluation of Activation Functions in Extreme Learning Machine

Karol Struniawski[1], Aleksandra Konopka[1] and Ryszard Kozera[1,2]

1 - Warsaw University of Life Sciences - SGGW
Institute of Information Technology
ul. Nowoursynowska 159, Warsaw, Poland

2 - The University of Western Australia
School of Physics, Mathematics and Computing
35 Stirling Highway, Crawley, Perth, Australia

Abstract. This study investigates the performance of 36 different activation functions applied in Extreme Learning Machine on 10 distinct datasets. Results show that Mish and Sexp activation functions exhibit outstanding generalization abilities and consistently perform well across most datasets, while other functions are more dependent on the characteristics of the task at hand. The selection of an activation function is intricately linked to the applied dataset and novel activation functions may possess superior generalization capabilities comparing to commonly employed alternatives. This study provides valuable insight for researchers and practitioners seeking to optimize Extreme Learning Machine performance for solving classification tasks.

1 Introduction

Extreme Learning Machine (ELM) is a type of neural network that was introduced by Huang et al. in 2004 [1]. The ELM architecture comprises of an input layer, a single hidden layer and an output layer of neurons. The number of neurons in the input and output layer are adapted to the specific task at hand. Due to the scarcity of theoretical methods, it is difficult to determine upfront the optimal number of hidden units for the ELM. Consequently, the latter is usually established through empirical evaluations. ELM has been widely applied in various fields, including image classification [2], medical diagnosis [3] and soil microorganism identification [4]. It is shown to be highly computationally efficient in both classification and regression tasks [5]. The ELM has become an increasingly popular Machine Learning (ML) technique in recent years as its versatility and effectiveness make it a valuable tool for a wide range of applications.

Extreme Learning Machine utilizes the McCulloch-Pitts neurons [6] for which an activation function needs to be determined. Huang et al. proved that in contrast to conventional gradient-based learning algorithms that are exclusively applicable to differentiable activation functions, ELM may also employ non-differentiable or piecewise differentiable activation functions [7].

In recent years, new activation functions are proposed that yield promising results in ML. The Rectified Linear Unit (ReLU) [8] is one of the activation functions that is adopted in Convolution Neural Networks (CNN). Another activation function that has gained popularity in recent years is the Exponential

Linear Unit (ELU) function that can achieve better performance than the ReLU in certain scenarios e.g. in networks with more than 5 layers [9].

The choice of activation function depends on the specific input task and researchers are continuously exploring new activation functions that can improve the performance of ELM applied to a wide range of problems [10]. In practical applications of ELM, literature overview shows that despite novel activation functions being developed the sigmoid and hyperbolic tangent functions remain the most widely used in ELM [11]. The insufficient treatment of complex comparison of activation functions applied for different datasets has been detected in the field in question. In one of the works that provide comparison between 11 different activation functions only one dataset is examined [12]. The observed activation function's performance on a single dataset raises concerns regarding its generalizability. To address this issue, we present a comprehensive performance investigation of the 36 different activation functions on 10 distinct datasets. The aim of this study is to determine whether certain activation functions outperform others and to assess whether the optimal selection varies depending on the dataset. Our hypothesis is that the activation function selection is intricately linked to the characteristics of the dataset used for which subset of functions can be identified that consistently exhibit superior or inferior performance. Noteworthy, this paper has utilized a diverse set of activation functions for the ELM, many of which have not been previously investigated in the literature like Mish, originally introduced in 2019 [13]. Novel activation functions may possess superior generalization capabilities in comparison to the commonly employed alternatives, rendering remarkable candidates for enhancing the performance of classification tasks using ELMs.

2 Extreme Learning Machine Classifier

In a supervised classification task N observations are represented as pairs of values denoted by $\{(x_i, t_i)\}_{i=1}^{N}$. The i-th vector x_i is composed of d features, while the corresponding i-th label t_i identifies the class to which the vector belongs. For a classification problem with M distinctive classes, t_i ranges from 0 to $M-1$. The input data is used to construct a matrix $X = (x_1, x_2, \ldots, x_N) \in \mathbb{M}_{d\times N}(\mathbb{R})$ with each $x_i \in \mathbb{R}^d$, along with a vector $T = (t_1, \ldots, t_N)$.

The input layer of an ELM neural network is composed of d neurons, while its output layer has a number of units equal to M. The network calculates N values $\{y_i\}_{i=1}^{N}$ as its output, which are then used to form the matrix $Y = (y_1, y_2, \ldots, y_N) \in \mathbb{M}_{N\times M}(\mathbb{R})$. To recognize a given input x_i, the maximal value of y_i observed on the p-th index is extracted. This assigns x_i to the p-th class. Suppose that a fixed number of neurons, denoted by L is selected for the hidden layer in advance. The weights connecting the input and hidden layers define the matrix $W \in \mathbb{M}_{d\times L}(\mathbb{R})$, where w_{ij} corresponds to the weight associated with the connection between the i-th input layer neuron and the j-th neuron in the hidden layer. The bias connections are represented by a vector $b = (b_1, \ldots, b_N)$. During the learning process of the ELM, the coefficients of W and b are determined

using a uniform distribution function $U(-1, 1)$. The outputs of the hidden layer neurons are stored in the matrix H. In ELM, the activation function $f : \mathbb{R} \to \mathbb{R}$ introduces non-linearity to the hidden layer output, which is crucial for the network's performance on tasks that involve complex relationships between input and output variables [1]. The weights β between the hidden and output layer in ELM can be calculated by solving the equation $Y = H\beta$. This system cannot be directly solved since H is non-invertible and $\|H\beta - Y\| = 0$ (see Huang et al. [1]). Instead, we estimate β as the minimizer of the mean residual square error: $\hat{\beta} = \arg\min_{\beta} \|H\beta - T\|^2 = H^{\dagger}T$, where $H^{\dagger}$ is the Moore-Penrose generalized inverse of H [14]. The pseudo-inverse of matrix $H^{\dagger}$ is uniquely determined and in the case of a non-singular matrix H, it coincides with an ordinary inverse, i.e. $H^{\dagger} = H^{-1}$. The matrix $H^{\dagger}$ gives the solution $\hat{\beta}$ so that $H\hat{\beta}$ is close to Y in terms of mean square error.

3 Methodology and results

The use of ELM's techniques necessitates the selection of an appropriate activation function and requires determination of the number of hidden layer units denoted by L. To enable clear comparison of results, we propose a lucidity experiment involving running ELM on a specific dataset using a fixed activation function and conducting 50 repetitions of 10% cross-validation. A search over the range of L values, from 100 to 5000 in increments of 100, is then performed to identify the optimal classification accuracy. The reported results reflect the highest accuracy obtained using a particular dataset and activation function at various values of L. In light of the extensive use of various activation functions in this study, it is recommended that each applied activation function should be referenced to the relevant scientific literature. Activation functions taken here into consideration are: identity $f(x) = x$, Binary Step Function (***BSF***) $f(x) = \{1 : x \geq 0; 0 : x < 0\}$, ***TanhRe*** $f(x) = tanh(x) + x$, ***HTan1*** $f(x) = \min(\max(x, -1), 1)$, ***Sine*** $f(x) = sin(x)$, ***ASin*** (Inverse Sine) $f(x) = \arcsin(\min(\max(x, -1), 1))$, ***Cosine*** $f(x) = cos(x)$, Soft Exponential (***Sexp***) $f(x) = \max(x, 0) + \ln(1 + exp\{-|x|\})$, Inverse Square Root Linear Unit (***ISRLU***) $f(x) = \frac{x}{1+e^{-1.5x}}$, Inverse Square Root Linear Units (***ISRLUs***) $f_{\alpha}(x) = \frac{x}{\sqrt{1+\alpha x^2}}$, Asymmetric Rectified Linear Unit (***AReLU***) $f(x) = \{x : x \geq 0; 0.1x : x < 0\}$, Bent's Exponential Linear Units (***BELUs***) $f(x) = \frac{\sqrt{x^2+1}-1}{2} + x$, Exponential Linear Units with Maxout (***Max-ELUs***) $f_{\alpha}(x) = \max(x, \alpha e^x - 1)$, Tilted Exponential Linear Units (***TELUs***) $f_{\alpha}(x) = \{x : x \geq 0; \alpha e^x - 1 : x < 0\}$, Soft Clip Exponential Linear Units (***SCELU***) $f_{\alpha}(x) = \{x : x \geq 0; \alpha e^x : x < 0\}$, Scaled Exponential Sine Linear Units (***SESLUs***) $f_{\alpha,\beta}(x) = \{x : x \geq 0; \alpha sin(\beta x) : x < 0\}$, Square Non-Linearity (***SQNL***) $f(x) = \{-1 : x < -2; \frac{x+x^2}{4} : x < 0; \frac{x-x^2}{4} : x \geq 2\}$, ***Soft Clipping*** $f(x) = \{-1 : x \leq -1; x : x > -1 \text{ and } x > 1; 1 : x \geq 1\}$, ***SineReLU*** $f(x) = \max(0, \sin(x))$, Rectified Square Root (***ReSQRT***) $f(x) = \sqrt{\max(0, x)}$,

# Samples	690	569	1k	6k	208	14k	13k	351	3k	846
# Features	14	30	11	166	60	14	16	34	180	6
Activation	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
identity	**77**	**95**	*77*	**96**	74	64	90	82	**95**	76
BSF	*60*	*57*	*2*	*56*	*55*	*55*	*16*	*74*	*39*	*31*
Sigmoid	76	94	84	**95**	74	88	91	84	**94**	77
Swish	77	94	84	93	**77**	88	91	83	93	**79**
ELiSH	76	93	81	94	74	77	90	85	94	75
TanH	75	94	82	94	*72*	86	90	*80*	**93**	75
HardTanH	70	88	*76*	86	72	*61*	*64*	81	90	*62*
ReLU	72	90	79	87	74	62	89	**88**	88	67
TanhRe	75	94	82	**94**	71	86	90	81	94	75
ELUs	76	93	81	94	74	77	90	85	94	75
***Soft*-Plus**	**77**	**95**	**84**	94	77	**89**	**91**	86	94	**79**
LReLU	71	90	79	88	74	63	90	**88**	88	68
SeLU	76	94	83	93	72	86	90	81	93	74
ReLU6	71	90	79	87	74	62	89	**88**	88	66
HTan1	73	90	79	92	74	63	74	82	93	70
Sinusoidal	75	93	**84**	94	**77**	**90**	**91**	*80*	93	77
Asin	*69*	*88*	78	89	73	*61*	*67*	83	93	63
Cosine	*47*	*48*	*1*	*40*	*37*	*41*	*1*	*34*	*12*	*14*
Sexp	**78**	**95**	**84**	**95**	**77**	**89**	**91**	85	94	**79**
Mish	**78**	**95**	**84**	94	**77**	**89**	**91**	83	**94**	**80**
ISRLU	77	94	83	92	76	87	91	82	92	77
RReLU	73	92	80	87	75	73	90	**86**	*86*	70
GELU	76	94	83	92	76	87	91	83	92	78
SELU	74	91	79	94	*71*	72	90	82	94	72
ISRLU	75	94	82	94	72	85	90	*80*	93	74
AReLU	72	91	79	89	74	62	90	**87**	90	69
BELU	77	94	83	**95**	**77**	82	91	84	**95**	78
Max-ELU	**79**	91	80	94	74	**91**	91	81	90	**80**
TELUs	76	93	81	94	74	74	90	85	94	74
SCELU	*61*	*86*	*24*	*71*	*66*	67	*21*	84	*56*	*37*
SESLU	76	94	82	94	72	78	90	82	94	76
SQNL	76	93	81	94	73	70	87	81	94	74
Soft Clip	73	90	79	92	73	63	74	82	93	70
SineReLU	71	89	79	*86*	72	65	88	84	*86*	64
ReSQRT	*68*	*88*	*77*	84	73	*61*	84	86	89	*58*
SiLU	77	**95**	**84**	93	77	86	91	83	93	79

Table 1: The accuracy (ACC) [%] of the Extreme Learning Machine (ELM) for a particular activation function and dataset with given samples and features number. The bold values indicate the top five ACC for a selected dataset among activation functions, while the italicized values represent the five functions with the lowest ACC.

Sigmoid Linear Unit (***SiLU***) $f(x) = \frac{x}{1+e^{-x}}$,. For the following activation functions details can be obtained in [15]: ***Sigmoid***, ***Swish***, Exponential Linear Squashing (***ELiSH***), ***TanH***, ***HTanH***, Rectified Linear Unit (***ReLU***), Exponential Linear Units (***ELUs***), ***SoftPlus***, Leaky ReLU (***LReLU***), Scaled Exponential Linear Unit (***SeLU***), ***ReLU6***, ***Mish***, Gaussian Error Linear Units (***GELUs***), Scaled Exponential Linear Units (***SELU***), Randomized Leaky ReLU (***RReLU***). In this paper, α and β are set to 1 as the selection of activation function parameters is beyond the scope of this research.

To provide the meaningful comparison between various activation functions 10 different datasets were used. The datasets are made publicly available by the UCI Machine Learning Repository for the purpose of classification tasks and are commonly used in ML [16]. For simplicity in further considerations datasets are marked as $A, \ldots, J$, where A - Australian credit card applications, B - Breast Cancer, C - Wine-Red, D - Musk, E - Sonar, F - EyeState, G - Dry Bean, H - Ionosphore, I - DNA and J - Vehicle. The experiment's results are presented in Tab. 1 and are analyzed in the conclusions section below.

4 Conclusions

The results (see Tab.1) indicate that some activation functions performed poorly, including BSF, Cosine and SCELU, which exhibited the worst results for all datasets (being 10 times in bottom 5 ACC). With high confidentiality we can exclude these functions from practical usage for ELM. On the other hand, the Mish (9 times), Sexp (8), SoftPlus (6), Maxout-ELUs (4) exhibited superior performance across datasets being in top 5. Noteworthy, each of these functions has never been chosen as the worst 5 for a given dataset. Identity and sinusoidal functions were in the top 5 for the 4 times, but simultaneously each of these was also once noted as the bottom 5. Their usage may be beneficial only for some of the assignments. Our experimental results demonstrate the exceptional generalization capabilities of Mish and Sexp activation functions for ELM. These functions have consistently performed well across 10 diverse datasets, encompassing various classification tasks. They have exhibited strong performance even when applied to datasets with varying numbers of features and samples. In contrast, the effectiveness of other activation functions has been more reliant on the specific characteristics of the tasks. Notably, the accuracy of the ELM classifier can differ significantly, up to 80 percentage points, depending on the chosen activation function. Therefore, it is crucial to evaluate the efficacy of activation functions for each task to ensure optimal classifier performance. Our experimental results highlight the promising potential of Mish activation function for ELM. Mish, a relatively new activation function in Deep Learning, has not been extensively utilized in ELM until now. Based on the obtained results we can straightforward conclude that the ELM's performance with selected activation cannot be measured on a single dataset as there is no guarantee that the activation function's generalization abilities will be sufficient for a given task. The analysis did not reveal any significant correlations between the number of

features or samples and the performance of activation functions. Instead, the results suggest that the performance is closely tied to the characteristics of the specific classification task. This observation opens up avenues for future research, particularly in exploring the implications of Universal Approximation theorems for ELM. Further research on the Mish and Sexp activation functions applied in ELM should be conducted to examine their performance on more datasets. It would also be beneficial to introduce optimization strategies to the values of α and β for specific activation functions.

References

[1] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme Learning Machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[2] Wei Bao, Yuan Lin, and Ming Cheng. Deep Convolutional Extreme Learning Machine and its application in image classification. *IEEE Trans. Cybern.*, 48(6):1886–1898, 2018.

[3] Guodong Li, Yanyu Zhao, Jie Liu, and Huaiqing Wang. A new Extreme Learning Machine with application to medical diagnosis. *J. Med. Syst.*, 41(3):1–8, 2017.

[4] Aleksandra Konopka, Karol Struniawski, Ryszard Kozera, Paweł Trzciński, Lidia Sas-Paszt, Anna Lisek, Krzysztof Górnik, Edyta Derkowska, Sławomir Głuszek, Beata Sumorok, and Magdalena Frac. Classification of soil bacteria based on machine learning and image processing. In *ICCS 2022*, pages 263–277, 2022.

[5] Gang Chen, Xuerong Mao, and Yingkai Zhang. Extreme Learning Machine: a review. In *NeurIPS*, pages 477–490, 2019.

[6] Mohamad H. Hassoun. *Fundamentals of Artificial Neural Networks*. MIT Press, 1st edition, 1995.

[7] Guang-Bin Huang, Qin-Yu Zhu, Kudo Mao, Chee Siew, P. Saratchandran, and Narasimhan Sundararajan. Can threshold networks be trained directly? *IEEE Trans. Circuits Syst. II Express Briefs*, 53:187 – 191, 04 2006.

[8] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. *Proc. Int. Conf. Electron. Comput. Artif. Intell.*, 15:315–323, 2011.

[9] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by Exponential Linear Units (ELUs). In *ICLR*, 2016.

[10] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme Learning Machines: A survey. *IEEE Trans. Neural Netw. Learn Syst.*, 30(10):2822–2840, 2019.

[11] Guang-Bin Huang, Hong Zhou, Xiao-Tong Ding, and Rui Zhang. Trends in Extreme Learning Machines: a review. *Neural Netw.*, 61:32–48, 2015.

[12] Dian Eka Ratnawati, Marjono, Widodo, and Syaiful Anam. Comparison of activation function on Extreme Learning Machine (ELM) performance for classifying the active compound. *AIP Conf. Proc.*, 2264(1), 2020.

[13] Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *CoRR*, abs/1908.08681, 2019.

[14] C. Radhakrishna Rao and Sujit Mitra. *Generalized Inverse of Matrices and its Applications*. John Wiley & Sons, 1971.

[15] An overview of activation functions. paperswithcode.com/methods/category/activation-functions.

[16] Dheeru Dua and Casey Graff. UCI machine learning repository. archive.ics.uci.edu/ml, 2017.

3.3.3 Metaheuristic Algorithms in Extreme Learning Machine for selection of parameters in activation function

Publication:

K. Struniawski, A. Konopka, and R. Kozera, *"Metaheuristic Algorithms in Extreme Learning Machine for Selection of Parameters in Activation Function"*, in Modelling and Simulation'2023. The 2023 European Simulation and Modelling Conference, 2023, p. 239–244.

Abstract: This research investigates the fusion of Metaheuristic Algorithms (MAs) with the Extreme Learning Machine (ELM) model to optimize parameters of activation function. While MAs have traditionally been employed for weights selection, a methodology that utilizes MA for the selection of activation function parameters was proposed. The performance of 24 distinctive activation functions was evaluated on diverse and widespread benchmark datasets: Ionosphere, Breast Cancer, Australian Credits, Musk and Banana. The results demonstrate a strong dependence on selecting an optimal activation function for each task, with variations in accuracy ranging up to 60 percentage points. The MA-ELM approach shows promising results, providing improved accuracy and reducing the number of required neurons in certain cases. The approach offers an efficient alternative to the typical MA-ELM method, requiring evaluation of only a few parameter values compared to the optimization of hundreds or thousands of weights. This approach enhances the generalization abilities of core ELM method and reduces computational time in comparison to typical MA-ELM. These findings validate the effectiveness of the proposed MA-ELM approach, contributing to the understanding of integrating MA with activation functions in ELM and offering insights for enhancing model performance in various applications.

METAHEURISTIC ALGORITHMS IN EXTREME LEARNING MACHINE FOR SELECTION OF PARAMETERS IN ACTIVATION FUNCTION

Karol Struniawski[1], Aleksandra Konopka[1] and Ryszard Kozera[1,2]

[1]Institute of Information Technology
Warsaw University of Life Sciences - SGGW
ul. Nowoursynowska 166, 02-776, Warsaw, Poland
email: {karol_struniawski, aleksandra_konopka, ryszard_kozera}@sggw.edu.pl
[2]The School of Physics, Mathematics and Computing
The University of Western Australia
35 Stirling Highway, Perth, Australia
email: ryszard.kozera@uwa.edu.au

KEYWORDS
Machine Learning, Metaheuristic Algorithms, Extreme Learning Machine, Activation Function Optimization, Computational Efficiency

ABSTRACT

This research investigates the integration of Metaheuristic Algorithms (MAs) with the Extreme Learning Machine (ELM) model to optimize parameters of activation function. While MAs have traditionally been employed for weights selection, a methodology that utilizes MA for the selection of activation function parameters was proposed. The performance of 24 distinctive activation functions was evaluated on diverse and widespread benchmark datasets: *Ionosphere, Breast Cancer, Australian Credits, Musk* and *Banana*. The results demonstrate a strong dependence on selecting an optimal activation function for each task, with variations in accuracy ranging up to 60 percentage points. The MA-ELM approach shows promising results, providing improved accuracy and reducing the number of required neurons in certain cases. The approach offers an efficient alternative to the typical MA-ELM method, requiring evaluation of only a few parameter values compared to the optimization of hundreds or thousands of weights. This approach enhances the generalization abilities of core ELM method and reduces computational time in comparison to typical MA-ELM. These findings validate the effectiveness of the proposed MA-ELM approach, contributing to the understanding of integrating MA with activation functions in ELM and offering insights for enhancing model performance in various applications.

INTRODUCTION

Metaheuristic Algorithms (MAs) have proven to be effective in solving optimization problems in a variety of domains. In the field of Machine Learning (ML), combining MA with the parameter selection of activation function has the potential to improve the performance of learning models (Wu et al. 2019). The Extreme Learning Machine (ELM) is a ML model that has gained popularity due to its computational efficiency and versatility (Huang et al. 2004) making it an attractive choice for a wide range of ML applications. Its unique approach, characterized by random initialization of hidden layer weights and a single-pass learning process, has simplified training while maintaining competitive performance.

In recent research, a hybrid approach combining ELM with MA has been proposed and evaluated in practical applications. The work (Chia et al. 2021) utilized particle swarm, moth-flame and whale optimization algorithm (WOA) to improve weight selection in ELM. Similarly, (Wu et al. 2019) applied genetic algorithms: ant colony optimization, cuckoo search algorithm and flower pollination algorithm in their practical context. A recent study successfully estimated the uniaxial compressive strength of rocks using the hybrid MA-ELM approach (Qiu et al. 2022). The results demonstrated the superiority of the WOA-powered MA-ELM model over the conventional ELM. The WOA-ELM model exhibited lower relative, minimum and mean residual errors with higher performance indices. These findings collectively indicate the enhanced performance of the hybridized ELM approach.

The choice of an appropriate activation function plays a crucial role in the performance and generalization capabilities of the ELM (Huang et al. 2015). Thoughts traditional sigmoid and hyperbolic tangent functions have been widely used in ELM, recent advancements have introduced novel activation functions (Glorot et al. 2011) primarily designed for Convolutional Neural Networks (CNNs). The effectiveness and suitability of these novel activation functions in the context of ELM are still largely unexplored. Additionally, many activation

functions require the specification of parameter values, typically fixed to a default value such as 1.

In this study, we propose a novel methodology that utilizes MA to optimize activation function parameters in ELM. While MAs have been traditionally employed for weights selection, our approach focuses on the evaluation of optimal activation function parameters. This solution offers potential benefits, including improved generalization abilities and reduced computational time compared to typical MA-ELM methods. Unlike the core method that seeks to find hundreds or even thousands of optimal weights through the MA process, our approach requires evaluating a maximum of only 5 parameter values (number of examined activation function's parameters is equal to 1, 2 or 5).

To validate our proposed concept, we conducted an extensive study evaluating the performance of 24 distinctive activation functions on five diverse datasets. The chosen widespread benchmark datasets: *Ionosphere, Breast Cancer, Australian Credits, Musk* and *Banana* are commonly used for ML performance evaluation (Kelly et al. 2023). Intentionally the datasets are chosen to yield a broad range representation and to exhibit diverse characteristics. We aim to assess the performance of activation functions in terms of accuracy, execution time and investigate whether the integration of Metaheuristic Algorithms improves the overall results. To accomplish this, a rigorous experimental setup was employed. A 50-times repeated cross-validation technique with 10 folds was utilized to ensure reliable and robust results. The evaluation metric of mean accuracy is used across multiple runs, providing an objective measure of model's performance.

In addition to evaluating the activation functions, we integrated five different metaheuristic algorithms from the python's mealpy package (Nguyen and Seyedali 2023) into the ELM framework. These algorithms, namely *OriginalHBA, OriginalSMA, OriginalTSO, OriginalARO* and *OriginalTSA* were specifically chosen due to their exceptional performance in terms of accuracy and execution time. Among the over 175 different methods implemented in the mealpy package, these algorithms showed promising results in the pre-experimental setup. By leveraging the optimization capabilities of MAs, our aim was to investigate whether they can further enhance the performance of ELM. The research necessitates utilization of MA with different parameters such as population size or number of iterations, and ELM that solely requires the quantity of neurons in hidden layer.

EXTREME LEARNING MACHINE

ELM is a dense feed-forward neural network classifier and regressor (Huang et al. 2004). It consists of an input, a single hidden and an output layer of neurons. The number of hidden layer units is empirically established as there is no theoretical method to determine the optimal number upfront. In a supervised classification task, observations are represented as pairs (x_i, t_i), where x_i is the input vector with d features, t_i is the corresponding one-hot encoded class label Potdar et al. (2017). The input data is used to construct matrices X and T. The bias and weights between input and hidden layer are determined using a uniform distribution function $U(-1, 1)$. The activation function introduces non-linearity to the hidden layer, which is essential for handling complex relationships (Hassoun 1995). The weights between hidden and output layer, denoted by β, are estimated by Moore-Penrose generalized inverse $H^{\dagger}$ (Rao and Mitra 1971) of the hidden layer matrix H. The solution $\hat{\beta}$ minimizes mean residual square error between $H\beta$ and T.

METAHEURISTIC ALGORITHMS

MAs have gained significant attention in the field of mathematical optimization and have found applications in various technological domains. The concept of optimization is not only prevalent in technology but also observed in the natural world, where organisms rely on efficient behaviors for survival and reproduction, as explained by Darwin's Theory of evolution (Brooks 1996). Scientists have attempted to model and describe these optimized activities of organisms using mathematical techniques, which are nowadays commonly referred to as MAs (Stützle 1999). The performance of models can be improved (Wu et al. 2019) combining bio-inspired optimization algorithms with ML.

In general, a constrained optimization problem involves minimizing an objective function, subjected to certain equality and/or inequality constraints (Venkatraman and Yen 2005). MAs are a class of nature-inspired algorithms that aim to guide the search process to avoid getting trapped in local optima (Yang 2010). They employ strategies such as intelligent initialization of solutions, incorporating randomness in a biased manner and utilizing search experience or memory to moderate the process (Lan and DePuy 2006). These algorithms simulate various natural behaviors and make assumptions about the environment to optimize the algorithm (Larrau and Potvin 2010). The search strategies employed by different MAs vary based on their underlying philosophies. In this study, we compare different MAs applied in the learning process of ELM.

RESULTS

The previous research highlighted the effectiveness of using a low number of algorithm iterations and small population size to optimize the weights in typical MA-ELM setup (Struniawski et al. 2023). This work performs a detailed investigation on parameter selection for

Table 1: Results of the experiment on the application of MA method for a given dataset with different activation functions. The table is in details described in Results section.

Dataset	Method	Activation	act_p	n	e	p	ACC [%]	t [s]
Ionosphere	*OriginalHBA*	cubic_spline	5	5000	1000	200	95.69	39
	OriginalARO	cubic_spline	5	5000	1000	200	95.63	39
	OriginalARO	cubic_spline	5	5000	5	50	95.57	10
	OriginalHBA	cubic_spline	5	5000	1000	50	95.54	10
	no_method	srelu	-	4000	-	-	95.31	-
	OriginalSMA	selu	2	400	1000	200	64.00	16
Breast Cancer	*no_method*	softshrink	-	400	-	-	93.51	-
	OriginalARO	sqrelu	2	500	5	50	93.41	7
	OriginalSMA	sqrelu	2	100	5	50	93.40	4
	no_method	softshrink	-	600	-	-	93.37	-
	OriginalHBA	aqrelu	2	500	1000	50	93.34	6
	OriginalARO	aq	2	100	1000	100	37.14	8
Australian	*OriginalHBA*	aq	2	400	1000	50	71.70	9
	OriginalHBA	sqrelu	2	300	5	50	71.60	7
	OriginalSMA	sqrelu	2	400	5	100	71.60	24
	OriginalTSO	sqrelu	2	400	5	100	71.60	20
	no_method	aq	-	700	-	-	71.60	-
	OriginalHBA	cubic_spline	5	800	5	50	53.08	11
Musk	*no_method*	isru	-	3000	-	-	96.70	-
	no_method	softshrink	-	3000	-	-	96.70	-
	no_method	elu	-	3000	-	-	96.70	-
	OriginalSMA	aqrelu	2	1000	5	200	96.60	85
	no_method	prelu	-	3000	-	-	96.60	-
	no_method	smooth_sigmoid	-	5000	-	-	48.10	-
Banana	*OriginalARO*	cubic_spline	5	800	1000	100	95.90	29
	OriginalHBA	cubic_spline	5	900	5	50	95.90	22
	OriginalTSA	cubic_spline	5	900	1000	50	95.80	19
	OriginalTSO	cubic_spline	5	800	1000	200	95.70	56
	no_method	hard_shrink	-	2000	-	-	93.60	-
	OriginalSMA	selu	2	200	5	200	40.63	1

MA within the context of ELM. The primary focus in this study was to explore the performance of MA-ELM once applied to search for optimal values of activation functions. The latter has received limited attention in prior literature. The approach involved designing a comprehensive set of experiments by varying different activation functions and their parameters. The activation functions applied in our study included: *cubic spline, aqrelu, selu, softshrink, leaky relu, isru, smooth sigmoid, prelu, sqrelu, isrlu, smooth hard tanh, elu, hard shrink, hard sigmoid, hardshrink, mish, sine, aq, swish, softexp, srelu, leaky softplus, softclip* and *gswish*. Their formulas can be found in the corresponding references in the field of ML that provide detailed explanations of each activation function. Noteworthy, most of these activation functions were originally developed and widely used for CNNs. However, they have not been extensively explored or applied in the context of ELM. To ensure a reliable evaluation, we conducted multiple runs and utilized appropriate assessment metric of mean accuracy. The objective was to testify the impact of MA on the performance of activation functions, particularly in terms of achieving high accuracy.

The experimental setup involved dividing the dataset randomly into training and testing sets using 10-fold 50-times repeated cross-validation (resulting in total 500 sets). The first set was utilized for MA evaluation, where we optimized the activation function parameter value by changing it during the MA process and evaluating the current fitness using mean square error. The MA algorithm searched for the optimal parameter or parameters, depending on the requirements of the activation function. Selected MA iterated for a specific number of iterations and population size, resulting in an optimal parameter value that minimizes given fitness function within a maximal number of epochs. This optimal value was stored in a memory and utilized in the remaining 499 folds of the cross-validated set (see flowchart Fig. 1). This approach ensured statistically meaningful results that were resistant to the randomness of cross-validation. As the final result, we observed the mean accuracy, which provided a robust assessment

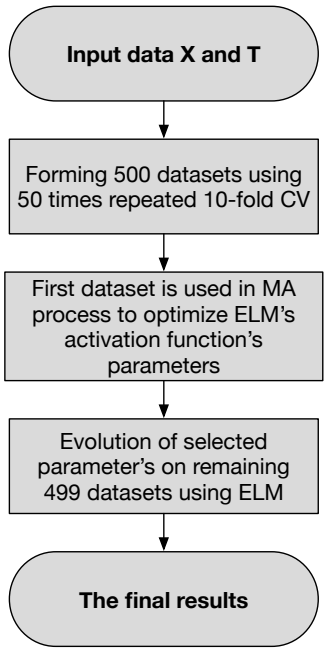


Figure 1: MA-ELM pipeline flowchart for ELM activation function's parameters optimization process.

of performance. The calculations were performed on PC class computer with Ryzen 9 3900X CPU, 64GB 3200 MHz DDR4 RAM, RTX 3090 GPU and Python 3.10 with Tensorflow 2.10.

In the results Tab. 1 the column *act_p* represents the number of parameters for activation function. The parameters used in the experiment are as follows: n denotes the number of neurons, e indicates the number of epochs and p represents the population size. The ACC column shows the mean accuracy obtained from 50 runs of 10-fold cross-validation, where the values of activation function's parameters were estimated in the first fold during MA evaluation. The t column represents the execution time of the MA method in seconds. The *no_method* indicates raw ELM evaluation using default parameters (set to 1) if needed. For each dataset we included 4 top ACC obtained, the *no_method* with highest accuracy as the core ELM reference point as well as the worst obtained performance to represent the impact of activation function and it's parameter selection to the final ACC (if in top 4 ACC *no_method* is present then we select top 5 ACC combined with worst noted result).

The results presented in Tab. 1 highlight the strong dependence on the selection of an optimal activation function for each specific task. For the *Ionosphere* dataset, the highest achieved accuracy (ACC) of 95.69% was generated using the *OriginalHBA* method coupled with the cubic_spline activation function. This combination, with a hidden layer size of $n = 5000$, $e = 1000$ epochs and a population size p of 200, outperformed other activation functions. It demonstrated a slight improvement compared to the core MA model using the Smooth Rectified Linear Unit (srelu) function and achieved over a 30 percentage point (p.p.) increase in accuracy compared to the worst-performing activation function. These results underscore the importance of properly evaluating the model's parameters.

In the case of *Breast Cancer* dataset, the core ELM model with the Soft Shrink (softshrink) activation function achieved the highest ACC, surpassing the MA-ELM models using other activation functions. The difference in ACC between the best and worst-performing activations was 60 p.p. indicating the significant impact of the chosen activation function.

Once applied to the *Australian Credit* dataset, the *OriginalHBA* method with the Asymmetric Quadratic Function (aq) yielded better results than the core ELM with the same activation function. Additionally, the MA-ELM approach achieved higher ACC with a smaller number of hidden layer units compared to the core ELM model. This demonstrates the advantage of using MA in reducing computational complexity, particularly in terms of prediction time.

Interestingly, for the *Musk* dataset, the core ELM model without MA application outperformed the MA-ELM models in terms of ACC for the activation functions Inverse Square Root Unit (isru), softshrink, Exponential Linear Unit (elu) and Parametric ReLU (prelu). Notably, the core ELM model with the smooth sigmoid function achieved an ACC that was 50 p.p. lower than the top-performing MA-ELM model. However, it is important to mention that the MA-ELM models required three times fewer hidden layer units.

In the case of *Banana* dataset, the cubic_spline activation function combined with the MA methods yielded the highest ACC of 95.90%. This result was over 2 p.p. better than the core ELM model with the Hard Shrink activation function, which also required more than twice as many hidden layer neurons.

CONCLUSIONS

In this study, the effectiveness of integrating MA with the ELM model to optimize activation function parameters was investigated. The significant variations in ACC based on the chosen activation function and its parameters were observed. For example, the *OriginalHBA* method combined with the cubic_spline activation function achieved the highest ACC of 95.69% on the *Ionosphere* dataset, outperforming other activation functions. Our findings demonstrate that the MA-ELM approach can be beneficial in applications where it significantly improves or provides comparable results with a lower number of required neurons. However, we also observed cases where the core ELM model without MA outperformed the MA-ELM models, indicating the importance of carefully selecting the activation function based on the dataset characteristics. The core ELM model with the softshrink activation function yielded

the highest ACC on the *Breast Cancer* dataset, surpassing the MA-ELM models. Obtained results emphasize the need to evaluate activation functions thoroughly, as the performance differences between the best and worst-performing functions can be as high as 60 p.p. in terms of ACC. These findings suggest that activation function selection plays a critical role in achieving optimal performance and generalization capabilities in ELM models.
Furthermore, our study demonstrates the potential of MA in enhancing the performance of activation functions and reducing computational complexity. By integrating MA algorithms, we achieved improved results with fewer hidden layer units, leading to computational efficiency benefits. Based on these outcomes, we recommend considering the MA-ELM approach for optimizing activation function parameters in ELM models. However, it is essential to conduct experiments on actual datasets and carefully evaluate the performance before deciding between the core ELM application and the proposed MA-ELM approach.
In conclusion, our research contributes to the understanding of MA integration with activation functions in ELM. These findings highlight the importance of function selection and validate the effectiveness of MA-ELM approach in enhancing model's performance.

REFERENCES

Brooks C.M., 1996. *The nature of life and the nature of death. J UOEH*, 5, no. 2, 133–145.

Chia M.Y.; Huang Y.F.; and Koo C.H., 2021. *Swarm-based optimization as stochastic training strategy for estimation of reference evapotranspiration using Extreme Learning Machine. Agric Water Manag*, 243, 106447. doi:10.1016/j.agwat.2020.106447.

Glorot X.; Bordes A.; and Bengio Y., 2011. *Deep Sparse Rectifier Neural Networks. Proc Int Conf Electron Comput*, 15, 315–323.

Hassoun M., 1995. *Fundamentals of Artificial Neural Networks*. MIT Press, 1st ed. ISBN 9780262082396.

Huang G.B.; Zhou H.; Ding X.T.; and Zhang R., 2015. *Trends in Extreme Learning Machines: A review. Neural Netw*, 61, 32–48. doi:10.1016/j.neunet.2014.09.003.

Huang G.B.; Zhu Q.Y.; and Siew C., 2004. *Extreme Learning Machine: A new learning scheme of feedforward Neural Networks*. In *IEEE Conf. Neural Netw.* vol. 2, 985–990. doi:10.1109/IJCNN.2004.1380068.

Kelly M.; Longjohn R.; and Nottingham K., 2023. *The UCI Machine Learning Repository*. `https://archive.ics.uci.edu`.

Lan G. and DePuy G.W., 2006. *On the effectiveness of incorporating randomness and memory into a multi-start metaheuristic with application to the set covering problem. Comput Ind Eng*, 51, no. 3, 362–374. ISSN 0360-8352. doi:10.1016/j.cie.2006.08.002.

Larrau M. and Potvin J.Y., 2010. *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd ed. ISBN 9781441916655.

Nguyen V.T. and Seyedali M., 2023. *MEALPY: An open-source library for latest meta-heuristic algorithms in Python. J Syst Archit*. doi:10.1016/j.sysarc.2023.102871.

Potdar K.; Pardawala T.S.; and Pai C.D., 2017. *A comparative study of categorical variable encoding techniques for neural network classifiers. Int J Comput Appl*, 175, 7–9. doi:10.5120/ijca2017915495.

Qiu J.; Yin X.; Pan Y.; Wang X.; and Zhang M., 2022. *Prediction of uniaxial compressive strength in rocks based on Extreme Learning Machine improved with Metaheuristic Algorithm. Mathematics*, 10, no. 19. ISSN 2227-7390. doi:10.3390/math10193490.

Rao C. and Mitra S., 1971. *Generalized Inverse of Matrices and its Applications*. John Wiley & Sons. ISBN 9780471708216.

Struniawski K.; Kozera R.; and Konopka A., 2023. *Performance of selected Nature-Inspired Metaheuristic Algorithms used for Extreme Learning Machine*. In *ICCS 2023*. ISBN 9783031360244, 498–512. doi:10.1007/978-3-031-36024-4_38.

Stützle T., 1999. *Local Search Algorithms for Combinatorial Problems - Analysis, Improvements, and New Applications*. DISKI. ISBN 9781586031190.

Venkatraman S. and Yen G., 2005. *A generic framework for constrained optimization using genetic algorithms. IEEE Trans Evol Comput*, 9, no. 4, 424–435. doi:10.1109/TEVC.2005.846817.

Wu L.; Zhou H.; Ma X.; Fan J.; and Zhang F., 2019. *Daily reference evapotranspiration prediction based on hybridized extreme learning machine model with bioinspired optimization algorithms: Application in contrasting climates of China. J Hydrol*, 577, 123960. doi:10.1016/j.jhydrol.2019.123960.

Yang X.S., 2010. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press. ISBN 9781905986286.

3.3.4 Credibility of randomness in Extreme Learning Machine

Publication:

K. Struniawski, A. Konopka, and R. Kozera, *"Credibility of randomness in Extreme Learning Machine"*, in Trust and Artificial Intelligence: Development and Application of AI Technology, J. Paliszkiewicz i J. Gołuchowski, 2025, p. 92–106. doi: 10.4324/9781032627236-10.

Abstract: Extreme Learning Machine (ELM) is a type of neural network introduced in 2004 by Huang et al. with increasing popularity over the last two decades. The core principles of ELM are simplicity of network topology and training process that finds a global minimum using algebraic transformations in a single iteration. This work focuses on testing credibility of randomness in ELM performance. Commonly, in typical applications of ELM, a uniform distribution function is applied as a stochastic number generator. We examine how the choice of randomness source in ELM impacts the final results achieved by this network in practical applications. The credibility of the applied stochastic number generators in ELM is a field of our examination using statistical test suits. The question dealt here is whether other distribution functions applied as a source of randomness in ELM have an impact on the network performance. The research highlights significant variations in model outcomes across diverse distribution functions and demonstrates notable effects associated with various random number generators, including those with top accuracy that fail to meet NIST criteria.

Credibility of randomness in Extreme Learning Machine

Karol Struniawski [1,*] , Aleksandra Konopka [1,]
and Ryszard Kozera [1,2]

[1]Institute of Information Technology, Warsaw University of Life Sciences - SGGW
ul. Nowoursynowska 159, 02-776 Warsaw, Poland
[2]School of Physics, Mathematics and Computing, The University of Western Australia
35 Stirling Highway, Perth, Crawley, WA 6009, Australia

E-mails: {karol_struniawski}, {aleksandra_konopka},{ryszard_kozera}@sggw.edu.pl

ORCIDs: 0000-0002-4574-2986, 0000-0003-1730-5866 and 0000-0002-2907-8632

Extreme Learning Machine (ELM) is a type of neural network introduced in 2004 by Huang et al. with increasing popularity over the last two decades. The core principles of ELM are straightforward network topology and training process that finds a global minimum using algebraic transformations in a single iteration. This paper focuses on testing credibility of randomness in Extreme Learning Machine performance. Commonly, in typical applications of ELM, a uniform distribution function is applied as a stochastic number generator. We examine how the choice of randomness source in ELM impacts the final results achieved by this network in practical applications. The credibility of the applied stochastic number generators in ELM is a field of our examination using statistical test suits. The question dealt here is whether other distribution functions applied as a source of randomness in ELM have an impact on the network performance. The research highlights significant variations in model outcomes across diverse distribution functions and demonstrates notable effects associated with various random number generators, including those with top accuracy that fail to meet NIST criteria.

Introduction

The Extreme Learning Machine (ELM) represents an efficient machine learning technique, distinguished by its single-hidden layer feedforward neural network (SLFN) architecture. Throughout this paper, ELM's weights between input and hidden layer are represented as α accompanied by a bias term b, along with the weights connecting hidden layer to output layer denoted as β (bias is restricted only to the input layer).

The conventional characterization of the learning process often involves an intelligent exploration of the search space that may be achieved by employing methods like stochastic gradient descent in conjunction with the backpropagation algorithm (BA) (Amari 1993). Once we allow the adaptability of parameters α, β and b the SLFNs are referred to as universal approximators. In the context of BA, several techniques for weight initialization play a crucial role in the iterative optimization process. Among these techniques, the most

commonly employed method entails initializing weights with small random values, thereby enhancing the solution's robustness and guarding against network overfitting. Alternatively, other initialization methods take into account specific characteristics of the input data, such as the number of input vectors or features. Noteworthy, examples include the Glorot (Xavier) and He initialization techniques (Nguyen 2021). Research findings consistently underline the sensitivity of BA to initial conditions (Kolen and Pollack 1990).

ELM distinguishes itself from conventional SLFNs in its unique approach to learning process. In ELM, the parameters α and b are randomly initialized and remain constant throughout the training process. Notably, the values of β are determined not through iterative procedures but upon solving an algebraic equations once, aimed at achieving an optimal solution in terms of mean square error.

This paper delves into research focused on two primary areas. *Firstly*, the experiments are conducted to investigate how the choice of the distribution function for random weights of α and b influences the ultimate outcomes. *Secondly*, the various random number generators are examined, evaluating their statistical properties using the U.S. National Institute of Standards and Technology (NIST) tests for randomness and pseudorandomness. This evaluation aims to ascertain their credibility in generating truly random numbers and assesses their impact on the final performance of ELM. The experiments reveal that different distribution functions have a significant impact on the performance of models, while various random number generators exhibit a moderate influence. These findings underscore the research's importance.

Review of Literature

In the field of evaluating the influence of random number generators, along with their quality assessment using statistical test suites and considering the utilization of distribution functions in the initialization process of weights α and bias b, there is a noticeable dearth of existing literature dealing with that issue. Surprisingly, no prior research has concentrated on investigating the ramifications of random number generators within the context of ELM. Dealing with ELM it is essential to determine the values of α, b, the number of hidden units and the activation function along with its associated parameters.

In 2012, R. Wang conducted a study examining the influence of randomly assigned weights between the input and hidden layer within the context of the ELM (Wang, Kwong, and Wang 2012). The fundamental objective of this research is to investigate whether the use of randomly assigned weights could positively impact on the performance of ELM. The experimental findings demonstrated here that, for numerous classification and regression problems, the expansion of feature space dimensions, achieved through the random initialization of weights α and biases, yielded better performance than kernel mappings. Kernel mappings involve techniques for either reducing or expanding the feature space through the application of kernel functions, commonly Mercer kernels (Srivastava, Schumann, and Fischer 2005).

In 2016, G. Dudek conducted an analysis of the ELM approximation capabilities, examining how they depend on the type of activation function and on the ranges from which input weights and biases are randomly generated (Dudek 2016). The finding of this study was experimental evidence confirming that ELM, when applied to function approximation tasks, demonstrates clear benefits when the input weights are randomly selected within a narrow range. We decided to move these findings even further to testify a different distribution functions.

Methodology

This section provides a detailed description of ELM and outlines the methods for generating random numbers based on selected distribution functions concluding with a comprehensive overview of the experimental setup.

Extreme Learning Machine

Extreme Learning Machine is a dense feed-forward neural network introduced in 2004 by Huang et al. (Huang, Zhu, and Siew 2004). ELM stands out for its remarkable speed in training, especially when compared to other machine learning techniques such as Multilayer Perceptron trained with BA (Li et al. 2019). Due to the random selection of weights between the input layer and the hidden layer, the network is highly resistant to overfitting. ELM can be used for classification and regression tasks being applied in various fields such as: signal processing, pattern recognition, market analysis, aviation and aerospace (Ding et al. 2015), big data security, medical diagnosis and predicting protein structure (Chen et al. 2017).

The network consists of input, single hidden and output layer of neurons. The input matrix of features for the training process is denoted as $X_{d\times n}$, where each of n analyzed samples is described by d features. The number of neurons in input layer is equal to d. Number of neurons in a hidden layer L needs to be a priori determined. Number of neurons in output layer M reflects the number of classes to which each sample is being assigned. The matrix of weights between input and hidden layer is denoted as $\alpha_{d\times L}$ and the bias $b = \{b_i\}_{i=1}^{n}$. The coefficients of $\alpha_{d\times L}$ and values of b are randomly generated from a selected probability distribution. Uniform distribution is commonly applied on the interval $(-1,1)$ for evaluation of α and on $(0,1)$ for b.

The weights between hidden and output layer, denoted as β, are computed using algebraic transformations. To compute β a matrix equation $H\beta = T$ needs to be solved. $H_{n\times L}$ is an output from the hidden layer which is computed as $H = f(X^T\alpha + b)$. Activation function, denoted here as f, is usually selected as tangensoidal or sigmoidal function (Huang et al. 2015). The matrix $T_{n\times M}$ is structured such that each row, denoted as $\{t_i\}_{i=1}^{n}$, consists of a vector of length M containing one-hot-encoded values for the i'th of the n classified samples. This encoding results in $t_i \in \{0,1\}^M$, ensuring that each row contains exactly one occurrence of the value 1 among its elements. The latter means that e.g. if $t_2 = \{0,0,1,0\}$ the second sample (of all n samples) is labeled to be assigned to the third out of $M = 4$ possible classes. The matrix H, due to the fact that generically it is not a quadratic matrix, it is non-invertible,

so the value of β cannot be directly computed but it needs to be estimated. The estimation of β, denoted as $\hat{\beta}$, is computed by means of mean square error between $H\beta$ and T (see(1)):

$$\hat{\beta} = \underset{\beta}{argmin}\|H\beta - T\|^2 = H^{\dagger}T. \qquad (1)$$

The value of $\hat{\beta}$ is computed with $\hat{\beta} = H^{\dagger}T$, where $H^{\dagger}$ is Moore-Penrose pseudo-inverse operation (Rao and Mitra 1971). In classification tasks, the output of the network is a matrix $Y_{n\times M}$. Each of the rows in Y denoted as $\{y_i\}_{i=1}^{n}$ corresponds to each of the classified samples. The classification result is inferred transforming the Y matrix to one-hot-encoded $\hat{T}$ applying *argmax* function which sets all values in a given row to 0 except of the maximal number in this row which is set to 1 as it corresponds to the label of class to which the sample is assigned.

Generating random numbers for a given distribution function

The algorithm for generating random numbers from a specified distribution function is necessary for initializing α and b in ELM.

Random number generators

The use of randomness in computing has a historical origin in methods that relied on physical phenomena, such as radioactive decay or electronic noise generation, to obtain random numbers through experiments. Today, a wide range of Pseudorandom Number Generators (PRNGs) exist, which simulate randomness algorithmically.

In practical terms, statistical testing plays a crucial role in establishing evidence that a generator is capable of producing numbers that exhibit the appearance of randomness (Soto and Rukhin 1999). To address this concern, the U.S. National Institute of Standards and Technology (NIST) has developed a comprehensive testing suite designed for assessing the randomness of random number generators. This suite comprises a collection of tests that evaluate various aspects of randomness, aimed at determining whether a sequence of numbers possesses the characteristics expected of truly random data.

Distribution functions

The first step in obtaining random numbers based on the given distribution function is to generate random numbers using the selected algorithm and then to calculate the value of the inverse cumulative distribution function (ICDF) for the generated random numbers. In particular, to obtain random weights $\beta_{d\times L}$ from the uniform distribution function $U(-1,1)$ with parameters $U_a = -1$ and $U_b = 1$ the distribution function (as per (2)) and ICDF (as outlined in (3)) are utilized. To achieve this, a matrix $R_{d\times L} \in (0,1)^{d\times L} \subseteq \mathbb{R}^{d\times L}$ containing random numbers is generated using an algorithm like PCG64 (implemented in Python's *randomgen* package) is initially created. Subsequently, $\beta_{d\times L}$ is obtained by applying the ICDF, denoted as F^{-1}, to the matrix R: $\beta_{d\times L} = F^{-1}_{U(-1,1)}(R)$. Notably, the ICDF is employed because random generators are typically configured to produce real numbers within the 0 to 1 range, which can be interpreted as probabilities p. Therefore, the aim is to determine the corresponding values of x for which the cumulative distribution function F (of the given

probability function with density function f) equals p. This can be expressed as $x = F^{-1}(p)$ or by finding x based on the probability density function (as described in (2), (3) and (4)).

$$f(x; U_a, U_b) = \begin{cases} \frac{1}{U_b - U_a}, & \text{if } U_a \leq x \leq U_b; \\ 0, & \text{otherwise,} \end{cases} \quad (2)$$

$$F^{-1}(p) = U_a + p \cdot (U_b - U_a), \quad (3)$$

$$p = \int_{-\infty}^{x} f(k)dk. \quad (4)$$

Both α and bias weights were generated using the following distribution functions, their ICDFs are well formed in literature:

1. Chi-squared (Chisq).
2. Lognormal (Lognorm).
3. Normal.
4. Pareto.
5. Uniform.
6. Weibull.

The respective parameter values employed in the above distribution functions are discussed in detail in Section 3.3. These parameters can either be explicitly provided or estimated using two well-known initialization methods: the Glorot/Xavier method or the He initialization method. In the case of the Glorot method, it involves utilizing a parameter within the respective ICDF (as outlined in (5)). The Glorot/Xavier method is a widely adopted weight initialization technique in neural networks (Glorot and Bengio 2010). Conversely, the He initialization method, which estimates the parameter's value (based on (6)), is originally designed to complement rectified linear activation functions (ReLU) and their variants. The He initialization method enhances the stability and efficacy of training deep neural networks (He et al. 2015). This method calculates the standard deviation for weight initialization, taking into account the number of input features:

$$\text{std\_glorot} = \frac{1}{\sqrt{\frac{d+L}{2}}}, \quad (5)$$

$$\text{std\_he} = \sqrt{\frac{2}{d}}. \quad (6)$$

Experimental Setup

The experiments in this research were conducted using a blend of the following elements:

1. Dataset.
2. Distribution function.
3. Random number generator.
4. Number of neurons in the hidden layer.

These computations produced a total of 28600 setups.

Dataset

The results are tested on five well-known publicly available UCI repository (Dua and Graff 2017) datasets:

1. **Australian** dataset stands as credit risk assessment comprising a total of 690 entries and 14 features concerning credit card applications, encompassing parameters like income, age and employment history.
2. **Banana** has emerged as a testbed for navigating the complexities of non-linearly of synthetic data with task to detect enigmatic shape reminiscent of a banana with a total of 5300 instances and two classes.
3. **BreastCancer** with 569 instances and 30 features derived from medical images, the dataset encapsulates minute structural nuances. The quest is to distinguish between malignant and benign breast masses.
4. **Ionosphere** with 351 instances and 34 attributes is the binary classification challenge of differentiating between "good" signals that pass through the ionosphere without interference and the "bad" category corresponds to instances that are disrupted or reflect radar signals, making them less useful.
5. **Musk** dataset is a collection of information about various chemical compounds. This dataset contains 6598 samples and describes each of these compounds using 166 features or properties. The goal of this dataset is to help distinguish between compounds that are associated with the scent of musk and those that are not.

Distribution Function

Both α and b weights are generated using the following distribution functions forming 13 different combinations:

1. α: chisq(2) and b: chisq(2),
2. α: chisq(glorot) and b: chisq(2),
3. α: chisq(he) and b: chisq(2),
4. α: lognorm(0,1) and b: lognorm(0,1),
5. α:lognorm(glorot) and b: lognorm(0,1),
6. α: lognorm(he) and b: lognorm(0,1),
7. α: normal(0,1) and b: normal(0,1),
8. α: normal(glorot) and b: normal(0,1),
9. α: normal(he) and b: normal(0,1),
10. α: pareto(1) and b: pareto(1),
11. α: uniform(-1,1) and b: uniform(0,1),
12. α: weibull(1) and b: weibull(1),
13. α: weibull(he) and b: weibull(1).

Random Number Generator

In our research, we employed a total of 22 distinct random number generators from Python's *randomgen* package. This package comprises both cryptographically secure generators designed to produce unpredictable and resilient random numbers, as well as robust PRNGs

with favorable statistical characteristics. These generators yield pseudorandom sequences characterized by strong statistical properties. Their operations typically involve a combination of bitwise manipulations, arithmetic transformations and state updates. The specific generators employed in our research are listed below:

1. *aes*,
2. *chacha*,
3. *dsfmt*,
4. *efiix64*,
5. *hc128*,
6. *jsf*,
7. *lxm*,
8. *mt19937*,
9. *mt64*,
10. *pcg32*,
11. *pcg64*,
12. *philox*,
13. *rdrand*,
14. *romu*,
15. *sfc*,
16. *sfmt*,
17. *speck128*,
18. *threefry*,
19. *xoshiro256*,
20. *xoshiro512*,
21. *xoroshiro128*,
22. *xorshift1024*.

Number of Neurons in the Hidden Layer

In this context, $L \in \{50,100,200, \dots ,900,1000,2000, \dots ,9000,10000\}$ represents the values for the number of neurons in the hidden layer that were subjected in the analysis.

Results

In this paper, the weights α and b for classification task were computed for each of the 28600 different setups using 50 times repeated 10-Fold cross-validation. This technique has been implemented in order to obtain statistically meaningful results. Consequently, this method generated a set of 500 distinct accuracy measurements for each unique configuration. To provide a comprehensive characterization of the performance distribution, key statistical metrics including the mean accuracy, the first quartile, the second quartile (median) and the third quartile were computed for each set of 500 accuracy measurements within the context of each configuration under consideration.

Within the existing body of literature, the uniform distribution function (from (2)) with parameters $U_a = -1$ and $U_b = 1$ or $U_a = 0$ and $U_b = 1$ has conventionally served as the predominant probability function for estimating values of α and b. Nonetheless, our research has revealed a departure from this convention, as the optimal accuracy results vary depending on the specific dataset under consideration (see Table 9.1). Remarkably, the highest accuracy scores were achieved using the Pareto(1) density function, yielding 0.844 for the Australian dataset and 0.869 for the Banana dataset. In stark contrast, when applied to the Musk dataset, this same set of distribution functions produced one of the lowest accuracy results, registering at a mere 0.475. For the BreastCancer dataset, the most favorable results were obtained with the Weibull(he) and Weibull(1) functions achieving a remarkable accuracy of 0.979. Interestingly, these very functions were associated with

suboptimal performance in the Banana dataset, yielding an accuracy score of 0.762. On a different note, the Ionosphere and Musk datasets exhibited their highest accuracy results when employing Normal distribution functions yielding 0.955 and 0.971, respectively. Specifically, the parameters for α were set to (0,1) for the Ionosphere dataset, while the He parameter configuration was employed for the Musk dataset. The difference between maximal and minimal accuracy computed with different sets of distribution functions yields 14.4, 10.7, 10.7, 13.5 and 49.9 for Australian, Banana, BreastCancer, Ionosphere and Musk dataset, respectively. This underscores that the choice of the random number generator has a moderate impact on the model's performance.

In this research, we systematically assessed a collection of 13 sets of distribution functions to determine their effectiveness in achieving optimal accuracy across five distinct datasets. This rigorous analysis involved the identification of the top 100 accuracy results for each dataset, resulting in a total of 500 values for consideration. Remarkably, the distribution function set that emerged as the most frequently employed was Pareto(1), which featured prominently in 157 of the meticulously examined results. This configuration consistently delivered exceptional performance, particularly in the Australian and Banana datasets. The second most prevalent combination, appearing 143 times, consisted of Weibull(1) and Weibull(1). This specific configuration demonstrated impressive accuracy outcomes, particularly in the Australian and BreastCancer datasets. In the third position, the Uniform distribution function with parameters $(-1,1)$ and $(0,1)$ emerged with a total of 70 occurrences among the top-performing results, showcasing its efficacy in delivering remarkable accuracy, particularly in the Ionosphere and Musk datasets.

In this study, we comprehensively investigate 22 distinct random number generators (listed in section 3.3.3). Specifically, within the domain of ELM, our research represents a pioneering endeavor. To the best of our knowledge, prior to this investigation, there has been a conspicuous absence of systematic assessments pertaining to the suitability of various random number generators for generating values of α and b within the context of ELM. The primary objective of our investigation is twofold. *Firstly*, we aim to achieve the highest level of accuracy in our computations. *Secondly*, we endeavor to establish the stability of the selected random number generator. Stability, in this context, entails ensuring that the computed accuracy results exhibit minimal variability between the first and third quartiles, as indicated by the interquartile range.

The attainment of peak accuracy results varies significantly across diverse datasets, as detailed in Table 9.1. Notably, distinct random number generators excel in optimizing accuracy for specific datasets. For the Australian dataset, the *xoshiro256* generator demonstrated its prowess by achieving the highest mean (computed from 50 times repeated 10-Fold cross-validation for various numbers of neurons) accuracy score of 0.844. Meanwhile, the Musk dataset saw the *xoshiro128* generator emerge as the champion, yielding an impressive accuracy of 0.971. In the context of the Banana dataset, it was the *philox* generator that excelled, delivering the highest accuracy of 0.869. Finally, for the BreastCancer dataset, the *pcg64* and *hc128* generators claimed the top spot, recording an exceptional accuracy score of 0.979. The difference between maximal and minimal accuracy computed with different generators yields 7, 2.4, 1.1, 1.4 and 0.5 for Australian, Banana, BreastCancer, Ionosphere and Musk dataset, respectively.

In order to facilitate a comprehensive comparison of the various random number generators, we meticulously analyzed the top 100 accuracy values (each value is a mean generated with 50 repetitions of 10-Fold cross-validation) for each of the five datasets, noting the frequency with which each of the 22 generators appeared. Remarkably, the *pcg64* generator emerged as the most frequently observed choice, securing the top position in a substantial 60 out of the possible 500 highest accuracy results. Following closely, the second and third positions were claimed by *hc128*, *chacha* and *efiix64* generators, all of which delivered noteworthy performance by achieving high accuracy results 30, 26 and 26 times, respectively. Notably, these generators consistently exhibited strong performance across all five datasets.

To assess the stability of the random number generators, we employed a box chart visualization (refer to Figure 9.1) using the Australian dataset, Pareto(1) function and a hidden layer comprising 1000 neurons. Among the generators examined, several demonstrated notable stability characteristics. Specifically, the most stable generators, as evidenced by minimal variability between quantile 1 and quantile 3, include *sfmt*, *xorshift1024* and *xoshiro256*, with differences of merely 0.043, 0.047 and 0.049, respectively. Conversely, a subset of generators exhibited comparatively lower stability. Notably, *xoshiro512*, *philox* and *sfc* yielded high differences of 0.087, 0.087 and 0.085, respectively; indicating greater instability in their generated results.

All 22 random number generators underwent a rigorous evaluation encompassing NIST tests. It is important to note that passing the NIST tests does not guarantee absolute randomness, but it does instill a level of confidence in the quality of the generated random numbers. It is worth highlighting that these tests are structured in such a way that a positive outcome (a test's p-value being less than a specified threshold) indicates that, based on the specific test, the generator did not produce genuinely random results. In cryptographic applications, the p-value threshold is typically set at 0.01; however, in our case, we chose a p-value threshold of 0.05 commonly applied in statistics. Subset of generators faced challenges in this evaluation, falling short on a various number of tests. Specifically, four tests eluded the *theefry* generator, while *pcg64* struggled with three. Meanwhile, *chacha* and *xoshiro256* encountered difficulties in two tests each, whereas *scf*, *speck128*, *dsfmt*, *mt64*, *xoshiro128*, *jsf* and *xorshift1024* experienced a setback on one test each.

Interestingly, two of the three most stable generators encountered challenges in passing either one or two tests. Even the widely successful *pcg64* generator, which consistently yielded top-tier results (60 out of 500 times), fell short on three of the tests. Moreover, the third generator, *chacha*, which achieved high results on 26 occasions out of 500, did not meet the criteria of two of the tests. *This intriguing observation leads to a noteworthy conclusion: in many instances, the generators that deliver the highest accuracy and demonstrate remarkable stability tend to be the very generators that do not successfully pass the prescribed tests.*

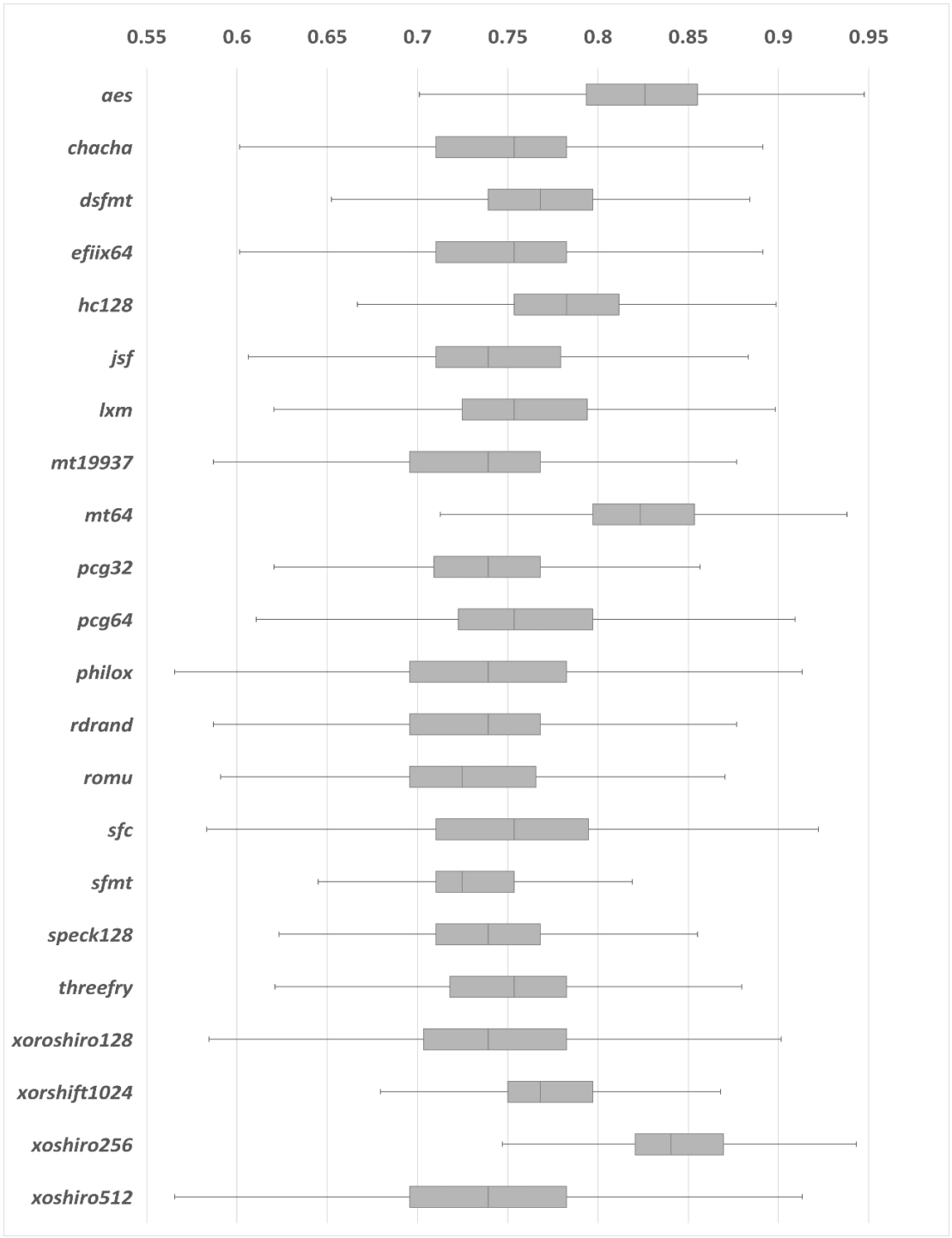

0.55
0.6
0.65
0.7
0.75
0.8
0.85
0.9
0.95
aes
chacha
dsfmt
efiix64
hc128
jsf
lxm
mt19937
mt64
pcg32
pcg64
philox
rdrand
romu
sfc
sfmt
speck128
threefry
xoroshiro128
xorshift1024
xoshiro256
xoshiro512

Figure 9.1. Accuracy results for classification on Australian dataset, with Pareto(1) distribution function and 1000 neurons in the hidden layer, using different random number generators computed applying 50 times a 10-Fold cross-validation. The boxchart presents the quantile 1, quantile 2 and quantile 3 of the 500 computed results for each of the setups.

Alpha	Bias	Australian	Banana	BreastCancer	Ionosphere	Musk
chisquare(2)	chisquare(2)	0.709	0.813	0.920	0.877	0.473
chisquare(glorot)	chisquare(2)	0.717	0.805	0.927	0.941	0.851
chisquare(he)	chisquare(2)	0.716	0.805	0.932	0.941	0.709
lognormal(0,1)	lognormal(0,1)	0.726	0.813	0.916	0.887	*0.472*
lognormal(glorot)	lognormal(0,1)	*0.700*	0.802	*0.874*	*0.820*	0.474
lognormal_he	lognormal(0,1)	0.712	0.813	0.883	*0.820*	0.474
normal(0,1)	normal(0,1)	0.722	0.804	0.933	**0.955**	0.970
normal(glorot)	normal(0,1)	0.718	0.804	0.929	0.933	0.970
normal(he)	normal(0,1)	0.714	0.815	0.929	0.947	**0.971**
pareto(1)	pareto(1)	**0.844**	**0.869**	0.965	0.899	0.475
uniform(-1,1)	uniform(0,1)	0.719	0.804	0.932	0.954	0.970
weibull(1)	weibull(1)	0.720	0.763	0.933	0.867	0.475
weibull_he	weibull(1)	0.800	*0.762*	**0.979**	0.884	0.474
	Generator					
	aes	0.820	0.846	0.977	0.950	0.968
	chacha	0.816	0.848	*0.968*	*0.941*	0.970
	dsfmt	0.836	0.855	0.970	0.946	0.967
	efiix64	0.796	0.865	0.973	0.949	0.968
	hc128	0.843	0.850	**0.979**	0.951	0.968
	jsf	0.826	0.866	0.969	0.944	0.967
	lxm	0.838	*0.845*	0.974	0.949	0.969
	mt19937	0.799	0.852	0.970	0.953	0.968
	mt64	0.820	0.862	0.969	0.951	0.970
	pcg32	0.815	0.852	0.972	0.943	0.967
	pcg64	0.820	0.858	**0.979**	0.946	0.970
	philox	0.795	**0.869**	0.969	0.947	0.969
	rdrand	0.824	0.848	*0.968*	0.946	0.967
	romu	0.830	0.865	0.975	0.954	*0.966*
	sfc	0.794	0.858	0.973	0.944	0.968
	sfmt	*0.774*	0.851	0.972	0.947	0.968
	speck128	*0.774*	0.864	0.972	0.946	0.970
	threefry	0.798	0.851	0.975	0.944	**0.971**
	xoroshiro128	0.825	0.852	0.974	**0.955**	0.967
	xorshift1024	0.775	0.862	*0.968*	0.944	0.970
	xoshiro256	**0.844**	0.855	0.970	0.947	0.968
	xoshiro512	0.791	0.856	0.971	0.946	0.970

Table 9.1. Classification Reports for the best mean accuracy results for five selected datasets depending on the distribution function and random number generator computed (for different numbers of neurons L).

Discussion

Conventionally, the uniform distribution function with parameters equal to $(-1,1)$ and $(0,1)$ has been employed as a source of randomness in ELM, but this research revealed intriguing variations in performance across different datasets. The selection of the randomness source in ELM significantly affected the final accuracy results, challenging established practices. Notably, specific distribution functions consistently emerged as top performers for particular datasets, underscoring the importance of considering dataset-specific characteristics when selecting randomness sources. The Pareto(1) distribution function, in particular, consistently delivered exceptional performance, notably in the Australian and Banana datasets. This diversity in optimal randomness sources emphasizes the need for a more sophisticated approach to randomness within the context of ELM.

The stability of random number generators in ELM was explored, marking a pioneering effort in this field. Surprisingly, the generators that achieved the highest accuracy and also a remarkable stability often did not meet the criteria of the prescribed NIST tests. Conversely, those generators that passed the tests did not consistently exhibit top-tier performance.

Conclusion

In conclusion, this research underlines the critical role of randomness in influencing ELM performance. The selection of the randomness source in ELM was demonstrated to yield a significant impact on accuracy across diverse datasets, challenging conventional practices. The variation in accuracy, ranging from 10 to 50 percentage points, when computed with different sets of distribution functions underscores the critical importance of selecting an appropriate distribution function for accuracy results. In contrast, the variation between maximal and minimal accuracy, which falls within the range of 0.5 to 7 percentage points when using different generators, suggests that the choice of a random generator, while not insignificant, holds relatively less weight compared to the selection of the distribution function. Additionally, a systematic assessment of random number generators in ELM was introduced, revealing that generators achieving the highest accuracy and stability often did not meet the stringent criteria of statistical tests. This observation suggests that the traditional criteria for randomness may not consistently align with optimal ELM performance.

The above findings underscore the necessity for a more subtle approach to randomness within ELM, which takes into account dataset-specific characteristics and explores a broader array of randomness sources. As ELM continues to gain prominence in various applications, comprehending the intricate relationship between randomness and performance forms an essential issue for achieving optimal outcomes and advancing the field.

References

Amari, S.-I. (1993). Backpropagation and Stochastic Gradient Descent Method. Neurocomputing, 5(4), 185–196. https://doi.org/10.1016/0925-2312(93)90006-O.

Chen, C., Li, K., Duan, M., & Li, K. (2017). Chapter 6 - Extreme Learning Machine and Its Applications in Big Data Processing. In Big Data Analytics for Sensor-Network Collected Intelligence (pp. 117–150). Academic Press. https://doi.org/10.1016/B978-0-12-809393-1.00006-4.

Ding, S., Zhao, H., Zhang, Y., Xu, X., & Nie, R. (2015). Extreme Learning Machine: Algorithm, Theory and Applications. Artificial Intelligence Review, 44(1), 103–115. https://doi.org/10.1007/s10462-013-9405-z.

Dua, D., & Graff, C. (2017). UCI Machine Learning Repository. University of California, Irvine, School of Information; Computer Sciences. archive.ics.uci.edu/ml.

Dudek, G. (2016). Extreme Learning Machine as a Function Approximator: Initialization of Input Weights and Biases. In CORES (pp. 59–69). Springer International Publishing.

Glorot, X., & Bengio, Y. (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks. In AISTATS. https://api.semanticscholar.org/CorpusID:5575601.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In ICCV (pp. 1026–1034). https://doi.org/10.1109/ICCV.2015.123.

Huang, G.-B., Zhou, H., Ding, X.-T., & Zhang, R. (2015). Trends in Extreme Learning Machines: A Review. Neural Networks, 61, 32–48.

Huang, G.-B., Zhu, Q.-Y., & Siew, C. (2004). Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. In IEEE Int. Conf. Neural Netw. (Vol. 2, pp. 985–990). https://doi.org/10.1109/IJCNN.2004.1380068.

Kolen, J. F., & Pollack, J. B. (1990). Backpropagation Is Sensitive to Initial Conditions. Complex Systems. https://api.semanticscholar.org/CorpusID:13373157.

Li, H.-T., Chou, C.-Y., Chen, Y.-T., Wang, S.-H., & Wu, A.-Y. (2019). Robust and Lightweight Ensemble Extreme Learning Machine Engine Based on Eigenspace Domain for Compressed Learning. IEEE Transactions on Circuits and Systems I: Regular Papers, 66(12), 4699–4712. https://doi.org/10.1109/TCSI.2019.2940642.

Nguyen, Q. C. (2021). A Fully Rigorous Proof of the Derivation of Xavier and He's Initialization for Deep ReLU Networks. ArXiv. https://doi.org/10.48550/arXiv.2101.12017.

Rao, C. R., & Mitra, S. (1971). Generalized Inverse of Matrices and Its Applications. John Wiley & Sons.

Soto, J., & Rukhin, A. L. (1999). Statistical Testing of Random Number Generators. National Institute of Standards and Technology. https://api.semanticscholar.org/CorpusID:9938076.

Srivastava, A. N., Schumann, J., & Fischer, B. (2005). An Ensemble Approach to Building Mercer Kernels with Prior Information. IEEE Transactions on Systems, Man, and Cybernetics, 3(3), 2352–2359. https://doi.org/10.1109/ICSMC.2005.1571500.

Wang, R., Kwong, S., & Wang, X. (2012). A Study on Random Weights Between Input and Hidden Layers in Extreme Learning Machine. Soft Computing, 16(9), 1465–1475. https://doi.org/10.1007/s00500-012-0829-1.

Biographical Notes

MSc Karol STRUNIAWSKI is the research and teaching assistant at the Institute of Information Technology, Warsaw University of Life Sciences – SGGW. He is Master of Science in Computer Engineering, earned with distinction, was achieved at Warsaw University of Life Sciences – SGGW in 2021. His primary research focus centers around the application of the highly efficient machine learning method known as Extreme Learning Machine (ELM). This innovative approach holds significant potential for various fields, and in the current research, an emphasis is placed on leveraging ELM, metaheuristic algorithms, and image processing. The ELM is utilized in research to precise classify and identify soil bacteria species as well as other classical Machine Learning approaches and Convolutional Neural Networks.

MSc Aleksandra KONOPKA is a research and teaching assistant at the Institute of Information Technology, Warsaw University of Life Sciences – SGGW. In 2021, she earned a Master of Science degree in Computer Engineering with distinction. Her primary research focuses on the theoretical exploration and practical application of classical artificial intelligence methods, advanced neural networks, and digital image processing within biological context. Her research projects involve the analysis of soil bacteria classification using machine learning and image processing techniques, aimed at enhancing data analysis, with potential applications in various fields such as agriculture and horticulture.

Prof. Ryszard KOZERA is currently a Director of The Institute of Information Technology at Warsaw University of Life Sciences - SGGW in Poland. He obtained a Habilitation Degree (Dr

Sc.) from Silesian University of Technology (Gliwice, Poland) in 2006, a PhD from Flinders University of South Australia (Adelaide, Australia) in 1991 and a MSc. Degree from Warsaw University (Warsaw, Poland) in 1985. Selected previous research positions include: Assoc. Prof. at The University of Western Australia, Perth, Australia 1991-2008 (current adjunct Assoc. Prof.) and Prof. at Warsaw University of Technology, Poland 2009-2012. In addition, he was also awarded three times Alexander von Humboldt Research Fellowships (Technical University of Berlin 1996-97; Christian Albrechts University of Kiel, 2000, 2004) in Germany. His research interests are: artificial intelligence, optimization, interpolation and approximation, computer vision and image analysis, numerical analysis, neural computation and networks, partial differential equations and in general applied mathematics and modeling processes in physics, engineering, biology, medical sciences and agriculture. He published over 140 scientific papers in prestigious international journals, research monographs and conference proceedings including 2 single authored papers exceeding 120 pages. He organized many international conferences, workshops and sessions among all in Australia, Germany, Greece, Mexico, Poland and presented over 70 invited talks and conference presentations world-wide. Prof. R. Kozera served also as co-editor of conference proceedings, research monographs, journal, conference and book reviewer as well as he has also won international grants and rewards. He supervised over 70 BSc., Honours or MSc. students and 2 PhD conferred students in Australia and Poland. He delivered lectures on over 30 different topics at the undergraduate, postgraduate and PhD. levels in computer science and applied mathematics.

3.4 Applications of ELM in the identification of microorganisms and in comparison to the Residual Neural Networks

3.4.1 Identification of Soil Bacteria with machine learning and image processing techniques applying single cells' region isolation

Publication:

K. Struniawski, A. Konopka, and R. Kozera, *"Identification of Soil Bacteria with Machine Learning and Image Processing Techniques applying Single Cells' Region Isolation"*, in Modelling and Simulation 2022. The European Simulation and Modelling Conference 2022, 2022, p. 76–81.

Abstract: Soil bacteria have a significant impact on agriculture and horticulture. These bacteria can be distinguished by the microbiologists based on their microscopic images. In our project this approach is automated with the aid of machine learning and image processing techniques. The implemented fully automated recognition system identifies five bacteria genera: Enterobacter, Rhizobium, Pantoea, Bradyrhizobium and Pseudomonas. The computations were performed on handcrafted image descriptors called features on the whole image or automatically selected sub-images including single bacteria instances. The accuracy results were compared for different feature selection and class recognition methods. Applying our new approach improved accuracy of the analyzed classifiers: from 5 to 31 percentage points (pp) for Support Vector Machine, from 2 to 52pp in case of Extreme Learning Machine and from 5 to 44pp using Extreme Learning Machine - Radial Basis Function, depending on the feature selection method. The best accuracy for the analyzed set of bacteria genera reaches 93.49% for Extreme Learning Machine - Radial Basis Function for features calculated on the single bacteria instances retrieved from a given image.

IDENTIFICATION OF SOIL BACTERIA WITH MACHINE LEARNING AND IMAGE PROCESSING TECHNIQUES APPLYING SINGLE CELLS' REGION ISOLATION

Karol Struniawski
Aleksandra Konopka
Ryszard Kozera
Institute of Information Technology
Warsaw University of Life Sciences - SGGW
ul. Nowoursynowska 159, 02-776
Warsaw, Poland
email: {karol_struniawski, aleksandra_konopka
ryszard_kozera}@sggw.edu.pl

14.07.2022

KEYWORDS
Soil bacteria, Microscopic images classification, Machine learning, Image processing, Extreme Learning Machines

ABSTRACT

Soil bacteria have a significant impact on agriculture and horticulture. These bacteria can be distinguished by the microbiologists based on their microscopic images. In our project this approach is automated with the aid of machine learning and image processing techniques. The implemented fully automated recognition system identifies five bacteria genera: *Enterobacter, Rhizobium, Pantoea, Bradyrhizobium* and *Pseudomonas*. The computations were performed on handcrafted image descriptors called features on the whole image or automatically selected sub-images including single bacteria instances. The accuracy results were compared for different feature selection and class recognition methods. Applying our new approach improved accuracy of the analyzed classifiers: from 5 to 31 percentage points (pp) for Support Vector Machine, from 2 to 52pp in case of Extreme Learning Machine and from 5 to 44pp using Extreme Learning Machine - Radial Basis Function, depending on the feature selection method. The best accuracy for the analyzed set of bacteria genera reaches 93.49% for Extreme Learning Machine - Radial Basis Function for features calculated on the single bacteria instances retrieved from a given image.

INTRODUCTION

Due to their characteristics, bacteria have a natural propensity to adapt themselves in most environments on our planet. They inhabit water, air, living organisms and the soil. Soil bacteria have an impact on plant growth, by for example suppressing growth of fungi, fixing significant levels of nitrogen or solubilizing phosphate (Syed-Ab-Rahman et al. 2018). They are also crucial for the food industry and are applied in biotechnological fields (Enez 2019). In addition, they have a significant impact on human health as e.g. they can be used to obtain substances necessary for the production of drugs (Abdul Samad et al. 2020), might cause human infection (Steffan et al. 2020) or show antibacterial activity against human pathogenic bacteria (Prashanthi et al. 2021). Soil bacteria are being identified by microbiologists with methods based on the analysis of DNA, applying indicators that change color in contact with specific bacteria species or by performing microscopic image analysis. The last approach can be automated by means of machine learning and image processing techniques, which consequently saves time and human resources. In our project we distinguish five selected soil bacteria genera: *Enterobacter, Rhizobium, Pantoea, Bradyrhizobium* and *Pseudomonas*. We decided to differentiate them on the genera level because some bacteria species are indistinguishable by sight due to their morphological similarities which ultimately requires usage of alternative methods for identification purposes. Each microscopic image in our dataset contains only one specific bacteria genera. This paper focuses on bacteria that affect crops in agriculture and horticulture. *Rhizobium* and *Bradyrhizobium* are nitrogen-fixing bacteria that live in symbiosis with legumes. Such bacteria are commonly applied forming an ideal solution for the improvement of soil fertility (Zahran 2000). *Enterobacter* are reported as opportunistic plant pathogens (Davin-Regli et al. 2019). Some *Pantoea* produce antimicrobials which are applied in the control of fire blight of pear and apple trees, while others have bioremediation potential (Walterson and Stavrinides 2015). *Pseudomonas* have a positive effect on plant growth and reduce plant diseases (Waghunde and Sabalpara 2021).

The bacteria samples used in this project were prepared by The National Institute of Horticultural Research in Skierniewice. They were grown in specific conditions and the images were prepared according to a strict procedure. More specifically, *Pantoea, Enterobacter* and *Pseudomonas* were grown for 48 hours in 26 Celsius degrees on Plate Count Agar, while *Rhizobium* and *Bradyrhizobium* were grown for 96 hours in 26 Celsius degrees on Yeast Mannitol Agar. The images of the samples are taken with Nikon 80i microscope on the same magnification level. Sample images from the dataset are available under the URL link: https://bit.ly/3qdDuHo. The motivation for this work was the question whether it is possible to select single instances of bacteria in a fully automated way and then classify a given image - from which we extracted the bacteria sub-images - to a given class. In the literature, we have encountered the approach of classification based on single instances of bacteria (Laga et al. 2014), however, the sub-images were manually selected from the image, while our approach is characterized by the full automation of this process.

IMAGE PREPARATION

The dataset consists of 128 images which needed to be prepared for the classification process. In a prior research (Konopka et al. 2022) a set of examined features is either calculated on the whole image or on the region of interest (ROI). In contrast with the latter, in this project a set of features is computed for each selected bacteria on a given image.

First, a segmentation of the ROI is performed to create a binary mask where black area represents the background and white area stands for the region occupied by the bacteria. To construct such a mask we converted a given image from RGB color space to grayscale. More specifically, the conversion is performed by calculating a weighted sum of the R, G and B components: $0.2989R + 0.5870G + 0.1140B$, then an Otsu method is applied (Gonzalez and Woods 2008) yielding an exact value for the threshold level. Finally, both open and close morphological operations (Srisha and Khan 2013) are performed in order to get rid of small objects and fill the black holes inside white areas.

The next step is to select sub-images from an input image (each of them containing only one bacteria instance) based on individual object's characteristics that are specified by the ROI mask. Let $O = \{o_i\}_{i=1}^{k}$ be set of all k sub-images extracted from a given image. First, all the objects which value of solidity is less or equal to 0.91 are filtered out and the remaining set is denoted as $F = \{f_i\}_{i=1}^{n}$, where n is the amount of sub-images after filtration - details on solidity measure are given in subsection: Sub-images geometric features. Following the above filtration process one selects a maximal number p of sub-images that are used to extract local information (in our research $p = 50$). Values of p and solidity threshold are experimentally selected based on accuracy (ACC) of the classifiers applied in this work. Let $S = \{s_i\}_{i=1}^{n}$ be set of filtered sub-images F sorted according to their covered area (which is the amount of white pixels in sub-image's mask). The median value of S (taking into consideration the area value) $s_m \in S$ is selected at index m. The value of p, which stands here for the number of objects surrounding s_m, is a priori specified and is applied in forming set B which is defined for even number of p as $B = \{b_i\}_{i=1}^{p} = \{s_i\}_{i=m-(p/2)}^{m+(p/2)-1}$ and for odd p as $B = \{b_i\}_{i=1}^{p} = \{s_i\}_{i=m-\lfloor p/2 \rfloor}^{m+\lfloor p/2 \rfloor}$. This operation ensures us to disregard either small objects which might be sample contamination or the groups of bacteria classified as a one large object. For the set of selected bacteria sub-images (and for their masks) image padding equal to 5 is applied to each direction (see Fig.1). Padding is used here to avoid omitting edges in calculation of texture features.

Each sub-image $o_i \in O$ has its solidity o_i^s and area value o_i^a. Assume we are provided with a set of $k = 10$ sub-images and $p = 5$. An example image preparation process is presented with the following steps.

1. First step is filtration of the O for solidity greater than 0.91. Let $O^s = \{0.94, 0.96, 0.93, 0.99, 0.95, 0.93, 0.92, 0.96, 0.99, 0.85\}$ be the values of solidity for each of the o_i, respectively. The sub-image o_{10} is eliminated as it corresponds to the value of o_{10}^s lower than given threshold value. As a result $F = \{f_i\}_{i=1}^{9} = \{o_i\}_{i=1}^{9}$ defining $n = \overline{\overline{F}} = 9$.

2. Next sub-images in F are sorted based on their area value. Let $F^a = \{100, 320, 650, 300, 200, 900, 700, 800, 450\}$ be area values of F. After sorting sub-images the new permutation of the set is $S = \{s_i\}_{i=1}^{9} = \{f_1, f_5, f_4, f_2, f_9, f_3, f_7, f_8, f_6\}$.

3. Then median value of the sorted set based on the area value is selected which in our example equals to $s_m = s_5 = f_9 = 450$ and p bacteria sub-images whose area values are surrounding the s_m form $B = \{b_i\}_{i=1}^{5} = \{s_i\}_{i=3}^{7} = \{f_4, f_2, f_9, f_3, f_7\}$.

4. For each $b_i \in B$ (and to their masks) 5 pixels of padding are applied for each of the four possible directions.

FEATURE CALCULATION

In order to accurately identify which bacteria genera appears in a given image a set of features forming an image descriptor is needed. Usage of non-convolutional neural networks requires such set to be handcrafted maximizing ACC of the classifier (Kotwal et al. 2022, Kruk et al. 2016, Rachmad et al. 2021). In previous section it is described how to obtain sub-images B_I and masks of

these sub-images M_{B_I} for the given image I. Note that all of the features mentioned in this section are calculated on combined sub-images C_{B_I} which are results of Hadamard product (Horn and Johnson 1985) of corresponding matrices B_I and M_{B_I}.

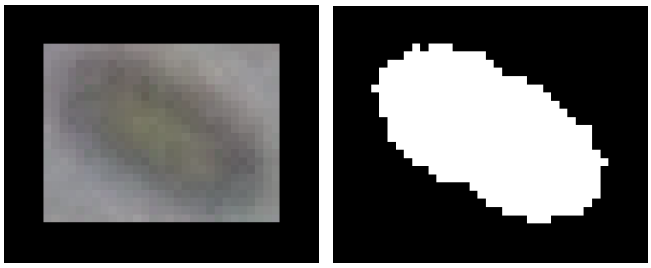


Figure 1: Sub-image of the selected object (*Enterobacter* bacteria instance) with applied padding (left) and its corresponding mask (right).

Color Features

The color features are the most commonly used image descriptors in machine learning (Sande et al. 2008). There is a vast variety of works adopting this concept in microorganisms identification based on microscopic images (Rachmad et al. 2021). In our project statistical measures based on image histogram (Chapelle et al. 1999) are obtained. Pictures taken by the microscope are stored as 8-bit RGB image in $2560 \times 1920px$ resolution, which means that each of 4915200 pixels is stored as a vector of 3 values corresponding to color channels R (red), G (green) and B (blue). Each of the values in this vector is stored on 8 bits, which in decimal system yields values $\bar{l} \in \{0, ..., 255\}$. Combined sub-images C_{B_I} are stored analogously, but with various resolutions that are strictly correlated with the size of the objects separated from the original image. Histogram of the pixel intensity values of the monochromatic image is defined as a measure that counts number of pixels for each different intensity value. Histogram of image I is a vector H of length 256, where $H[\bar{l}]$ is defined as a number of pixels on image I with value equal to $\bar{l}$. Similarly, for the color image one isolates every channel from the C_{B_I} and calculates the respective three histograms $H_R[\bar{l}]$, $H_G[\bar{l}]$ and $H_B[\bar{l}]$. Following the latter one obtains statistical measures based on each histogram such as: Mean, Standard deviation, Kurtosis, Skewness and Entropy creating in total 15 features based on color information. The dataset is extended by color statistical measures - Mean and Standard deviation calculated directly on the image for a given image channel. Combining all color features together yields 21 color traits calculated for every C_{B_I}.

Texture Features

Texture features computed in this research are based on calculation of GLCM (Grey Level Co-occurrence Matrix) and GLRLM (Gray Level Run-length Matrix). The concept of GLCM was introduced by Haralick (Haralick et al. 1973) and the idea to apply run-length concept in texture analysis was proposed by Galloway (Galloway 1975). Both approaches have initially been used to distinguish between different terrain types. Extraction of texture features is applied in many different fields including classification of satellite or aerial photographs, crop identification (Iqbal et al. 2021), analysis of the fingerprints (Bakti et al. 2021) or interpretation of medical images (Novitasari et al. 2019).
For calculation of GLCM and GLRLM the image is transformed from RGB color space to grayscale and then it is quantized to l levels. The idea is to shift a small window through the image without overlays and note occurrences of pixels (for GLCM) or occurrences of run-length of pixels (for GLRLM). GLCM and GLRLM are calculated for four orientations defined by angle α equal to $0°, 45°, 90°$ or $135°$. Both methods require specified size of the window $m \times n$, value of α and maximal distance d. As a result 4 GLCM and 4 GLRLM are obtained for different values of α. Then on each matrix one computes specific statistical measures. More specifically, measures calculated for GLCM are: Contrast, Correlation, Energy, Homogeneity, Autocorrelation, Cluster Prominence, Inverse Difference, Dissimilarity and Entropy. In case of GLRLM other statistical measures are computed: Short Run Emphasis, Long Run Emphasis, Grey Level Non-uniformity, Run Length Non-uniformity, Run Percentage, Low Grey Level Run Emphasis, High Grey Level Run Emphasis, Short Run Low Gray Level Emphasis, Short Run High Gray Level Emphasis, Long Run Low Gray Level Emphasis and Long Run High Gray Level Emphasis (Novitasari et al. 2019). In total one obtains 4 values for each of the measures computed on GLCM or GLRLM. As a final feature value a mean of the measures is calculated for all matrices e.g. $Energy = 0.25(Energy_{GLCM^{0°}} + Energy_{GLCM^{45°}} + Energy_{GLCM^{90°}} + Energy_{GLCM^{135°}})$.
This paragraph is closed with an example of GLCM computation. Assume that the variables are set to $l = 4$, $\alpha = 135°$, $d = 2$ and size of window w is 4×4. The GLCM always has a size of $l \times l$. Suppose that the pixel $w_{1,1}$ is now in question. Its current value is equal to 3 (see Fig.2) so the values in the fourth row of the GLCM are incremented. For a given d and α only the values $w_{2,2} = 0$ and $w_{3,3} = 2$ are taken into account. Consequently, the values are incremented in first and third column (in fourth row). The same operation applies to all other pixels in w.
A similar approach is adopted for GLRLM. This time the size of the matrix is $l \times d$ and one does not iterate over the window but registers occurrences of vectors for

given values of l and length d in the whole window w for the selected angle α. For example if 2 occurrences of the graylevel equal to 0 for the distance $d = 3$ are observed in w one increments the value by two in first row (which stands for graylevel equal to 0) and in third column (which represents the distance) (Konopka et al. 2022).

3	1	3	1
1	**0**	2	3
2	3	**2**	2
2	3	0	1

0	0	0	0
0	0	0	0
0	0	0	0
+1	0	**+1**	0

Figure 2: Increment of the values in GLCM (right) based on the window $w_{l,l}$ (left) pixel $w_{1,1}$ for $l = 4$, $d = 2$ and $\alpha = 135°$.

Sub-images Geometric Features

Bearing in mind that measures could be calculated for an object separated from the image (representing a single bacteria) we decided to introduce extra geometric features measuring the solidity, convex area, eccentricity and Feret diameter maximum value (Yap et al. 2012). For a given separate object its area, corresponding to the number of white pixels in the sub-image's mask, can be calculated. Convex image is an image (binary mask) that represents smallest convex polygon (called convex hull) that can contain object, where pixels within hull are equal to 1 (see Fig.3). Convex area is amount of pixels within a hull of convex image. Solidity is a number of non-zero pixels in C_{B_I} divided by number of non-zero pixels in convex image of C_{B_I}.

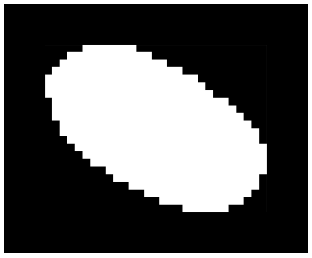


Figure 3: Convex image computed for sub-image presented in Fig.1.

Eccentricity is the ratio of distance between the ellipse foci and major axis length. Eccentricity equal to zero means that object is a circle, whereas eccentricity equal to one means that object is a line. Note here that foci of the ellipse are two points whose sum of distances from any point on the ellipse is always the same. Feret diameters are commonly used as image features (Yap et al. 2012, Pons et al. 2002). We decided to apply maximum Feret diameter, which is defined as the maximum distance between two boundary points on the antipodal vertices of convex hull. In order to compare obtained results with classical approach, where measures are being calculated for a whole image (not the separate objects) a benchmark set of features is calculated. The four additional traits can be only computed for C_{B_I}, because of that a benchmark feature set calculated for I does not include features based on separate objects on the image. Feature calculation process ends with normalization of the trait values between -1 and 1 to avoid differences in ranges of the selected features that may have impact on classificators performance.

FEATURE SELECTION

Feature sets may contain traits that are weakly correlated with affiliation to the class. The other phenomenon that may occur is high correlation between features. Both situations mentioned above may deteriorate performance of the classifiers. To eliminate the latter such features should be removed from the set. In order to obtain the meaningful results two sets of features are used. The first one is called *Normal* and contains 128 samples with 41 calculated features. The second one is called *Separate* and it consists of 5742 sub-images described by 45 features. To both sets feature selection methods: Conditional Mutual Information Maximization (CMIM) (Liang et al. 2019), Fast Correlation Based Filter (FCBF) (Kruk et al. 2016) and Feature Selection based on Genetic Algorithms (Calzolari 2022) are applied. Sets of traits that are created in this way are named for e.g. *Normal_FCBF* or *Separate_Genetic*. The numbers of features that are in these sets are enlisted below (written in brackets after the set name): *Normal*(41), *Normal_CMIM*(3), *Normal_FCBF*(4), *Normal_Genetic*(7), *Separate*(45), *Separate_CMIM*(5), *Separate_FCBF*(2) and *Separate_Genetic*(13).

CLASSIFICATION

The accuracy of the classification process is measured with the cross validation (CV) mechanism. In case of calculation of features for each image CV is performed in a standard way, however when performing this process on sub-images the approach is different. Assume that a set of n images $I = \{I_i\}_{i=1}^{n}$ is given. One selects now p (or less) sub-images $B_I = \{b_i\}_{i=1}^{p}$ for every microscopic image. Each sub-image represents one bacteria on the image. The number of sub-images for the whole set of images is not greater than np. Next a set of features for each sub-image separated from the image is specified. In CV one needs to randomly shuffle the set before selecting training and testing groups. In our setting sub-images should not be shuffled but one should merely shuffle images to ensure that upon such permutation the sub-images attached to a given image are allocated either to the training or to the testing set. More specifically, as-

sume a set of $n = 20$ images I is given and $\Omega = \{B_{I_i}\}_{i=1}^{n}$ represents a full set of sub-images extracted from each I_i. Once 5-Fold cross validation is conducted then a possible outcome from shuffling (for one of 5 iterations) may render a testing set $Ts = \{B_{I_1}, B_{I_5}, B_{I_{10}}, B_{I_3}\}$ and the training set $Tr = \Omega \backslash Ts$. Once the training and testing sets are specified the classification methods are trained on the set Tr. Next, a group of sub-images is selected from Ts (for example B_{I_1}) and it is predicted to which class each sub-image belongs. Then a majority voting is performed (on the predictions of B_{I_1}) to decide to which class a given image should be assigned (in our case image I_1 as its sub-images B_{I_1} are analyzed). Similar operations are performed for other sets of sub-images classifying all images from the testing set (I_1, I_5, I_{10}, I_3). Classification methods used in this work include: Random Forest (RF), Support Vector Machine (SVM) (Kramer 2013), Extreme Learning Machine (ELM) (Huang et al. 2006) and Extreme Learning Machine - Radial Basis Function (ELM-RBF) (Dhini et al. 2021).

Extreme Learning Machine

ELM is dense feedforward neural network with one hidden layer that is capable to solve classification and regression tasks. Activation function in neurons in the hidden layer and their amount needs to be determined empirically by iterative ransack search space comparing accuracy of the model. Currently, there is no known method to optimize number of neurons in hidden layer. The network learning is defined in two steps. First, weights between input and hidden layer and bias are randomly assigned based on continuous uniform distribution $(-1, 1)$. Second, weights between hidden and output layer are computed performing the Moore-Penrose pseudo-inverse operation (Rao and Mitra 1971). The latter assures that the found solution is the best in means of mean-square error. Network's weights are calculated performing mathematical operations only once. Pseudo-inversion cannot be parallelized, however thanks to singular value or QR decomposition it is very well optimized. Learning speed of ELM can be thousands times faster than other methods like Multilayer Perceptron (MLP) (Li et al. 2019). Let H be matrix defined as the neurons output values from the hidden layer and T be our target. In non-binary classification tasks output layer is assumed to contain the same amount of neurons as number of classes in our dataset. Target matrix T may be understood as numerical mapping affiliation of the selected image to the class. Let image I represent class 0, where number of classes is 5. Then T is formatted as a vector $[1, 0, 0, 0, 0]$ assigning 1 to the first neuron that is responsible for affiliation of the classified object to the first class. During network testing process response values are compared on the output layer of the network searching for maximum value. If it is observed on the first neuron then image is assigned to class 0. Weights between hidden and output layer $\hat{\beta}$ are computed by: $\hat{\beta} = H^{\dagger}T$, however to optimize calculation of pseudo-inversion as we mentioned before QR decomposition is more commonly used than orthogonal projection method (Rao and Mitra 1971). QR decomposition of matrix A is defined as $A = QR$, where R is upper triangular and Q is orthogonal ($QQ^T = \mathbb{1}$). Then $H^{\dagger} = H^T(HH^T)^{-1} = R^TQ^T(R^TR)^{-1} = Q^TR^{-1}$.

Extreme Learning Machine - Radial Basis Function

ELM-RBF is a variant of ELM with added one additional layer that maps actual inputs to their distances to centroids by means of Radial Basis Function. This application of ELM is similar to Radial Basis Networks and brings much more robustness to the prediction (Lee et al. 1999). Let $X_{s\times f}$ be an input matrix representing s and f as number of samples and features, respectively. Then k-means clustering algorithm is performed based on our input matrix defining number of clusters k empirically. As a result matrix $C_{k\times f}$ with coordinates of k centroids is obtained. In the next step we calculate distances from each sample s to all k centroids by means of Euclidean distance obtaining matrix $D_{s\times k}$. In the final step matrix $R_{s\times k}$ is computed and that ultimately becomes an input to ELM. For every sample s_i, where $i = 1, \ldots, s$ a maximum distance to the centroid d^{max} and $\sigma_i = \frac{d_i^{max}}{2k}$ are defined. Then distances between s_i and each centroid C_j are computed, where $j = 1, \ldots, k$ filling columns of matrix K in a given i-th row, $K[i, j] = exp\left\{\frac{-D[i,j]}{2\sigma_i^2}\right\}$. In the last step, matrix K serves as an input to regular ELM network. The number of clusters k for k-means method, activation function on neurons in hidden layer and their amount are specified. If $k < f$ we obtain dataset that is reduced in comparison to X, whereas if $k > f$ then we expand our dataset. We iteratively ransack searching space of these 2 parameters (fixing activation function to *tanh*) looking for the best model performance in terms of accuracy.

RESULTS

On each of the formerly prepared sets of features the classification using RF, SVM, ELM and ELM-RBF methods is performed. Computed results presented in Tables 1, 2, 3 and 4 are mean accuracy values of 50 runs of 10-Fold Cross Validation on a given classification method. RF, ELM and ELM-RBF needed to be tuned by selecting parameter values iteratively exploring the searching space by running 50 times 10-Fold CV and comparing mean accuracy. The generated results clearly indicate that SVM, ELM and ELM-RBF classifiers achieved the best performance for datasets containing sub-images separated from the original microscopic image.

Feature Set	ACC
Normal	0.7740
Separate	**0.8233**
Normal_FCBF	0.5822
Separate_FCBF	0.6957
Normal_CMIM	0.3832
Separate_CMIM	0.6932
Normal_Genetic	0.7929
Separate_Genetic	0.7663

Table 1: Support Vector Machine classifier results on the different feature sets.

Feature Set	ACC	Trees Number
Normal	0.9266	420
Separate	0.8943	200
Normal_FCBF	0.8100	280
Separate_FCBF	0.6892	70
Normal_CMIM	0.5994	170
Separate_CMIM	0.8229	10
Normal_Genetic	**0.9334**	220
Separate_Genetic	0.9034	320

Table 2: Random Forest classifier results as the highest obtained mean ACC during tuning the classifier by changing a number of trees (from 10 to 500) on the different feature sets.

Feature Set	ACC	Number of neurons
Normal	0.9058	2500
Separate	**0.9291**	1600
Normal_FCBF	0.2835	1700
Separate_FCBF	0.6268	400
Normal_CMIM	0.2828	100
Separate_CMIM	0.8023	1900
Normal_Genetic	0.7260	1000
Separate_Genetic	0.9204	1400

Table 3: Extreme Learning Machine classifier results as the highest obtained mean ACC during tuning the classifier by changing number of neurons (from 100 to 5000) on the different feature sets.

Feature Set	ACC	Number of neurons	k
Normal	0.8862	4900	95
Separate	**0.9349**	1000	56
Normal_FCBF	0.6100	3000	94
Separate_FCBF	0.6849	500	48
Normal_CMIM	0.3787	4200	93
Separate_CMIM	0.8167	1400	40
Normal_Genetic	0.8579	4300	96
Separate_Genetic	0.9246	1300	45

Table 4: Extreme Learning Machine - Radial Basis Function classifier results as the highest obtained mean ACC during tuning the classifier by changing number of neurons (from 100 to 5000) and clusters - k (from 1 to 100) on the different feature sets.

In case of SVM the best ACC equals 82.33% for a *Separate* dataset (see Tab.1). ACC is 5pp (percentage points) higher than for the *Normal* dataset. Analyzing SVM performance on sets that are obtained applying feature selection methods a substantial increase in ACC is noted on *Separate_CMIM* set in comparison to *Normal_CMIM*. In this case ACC almost doubles lifting from 38.32% to 69.32% rate. However, a 2.6pp decrease of ACC is noted using *Separate_Genetic* comparing to *Normal_Genetic* dataset.

The best result for RF classifier amounting to 93.34% is obtained for 220 trees and *Normal_Genetic* set (see Tab. 2). In case of RF datasets with sub-images perform better only for CMIM, whereas for other feature selection methods yield worse results.

For ELM and ELM-RBF much better ACC results are generated on *Separate* datasets than on *Normal* ones. The highest overall ACC amounts to 93.49% which is also the highest result observed in this paper. It is reached for ELM-RBF with 1000 neurons, 56 centroids and *Separate* dataset (see Tab.4). The latter outperforms ELM-RBF ACC for *Normal* dataset by 5pp. Differences in ACC can be significant recalling ELM on *Separate_CMIM* with 52pp increase of accuracy over *Normal_CMIM* (see Tab.3). In case of ELM and ELM-RBF an increase of ACC using *Separate* over *Normal* sets is transparent. Noticeably, the best ACC for ELM and ELM-RBF is obtained for sets without feature selection method applied.

CONCLUSION

Our aim was to create a system that automatically extracts sub-images from the image and then classifies each image to a bacteria genera by means of machine learning classifiers. The obtained results are satisfactory - for 3 out of 4 analyzed methods we obtained an increase in the accuracy. Experiments render the best ACC reaching 93.49% for ELM-RBF applied to full set of features (with no feature selection method) on *Seperate* dataset with 1000 neurons and 56 centroids. In contrast, the best overall result for dataset with no sub-images amounts to 93.34% correct classifications using RF with 220 trees on set with Genetic feature selection method applied. In addition RF is the only method among tested, which perform worse on *Seperate* sets of features than on *Normal* sets. Applying SVM, ELM and ELM-RBF we observed significant increase of ACC that was mostly noticeable applying feature selection methods such as CMIM and FCBF. Based on our experiments we see that approach presented in this research could be applicable in practical tasks of the soil bacteria identification especially using ELM and ELM-RBF methods that can be engaging due to their fast learning rate and excellent results. In this situation, as we experimentally showed, the segmentation of the single bacteria from the image and proposed method of classification may im-

prove overall ACC rate of the model from 5 up to 31pp for SVM, 2 up to 52pp in case of ELM and similarly from 5 up to 44pp upon using ELM-RBF.

FUTURE RESEARCH

Classification based on sub-images containing single bacteria instances has a great potential for further research. This approach can be applied for classification system with multiple bacteria genera on one image. In upcoming works we intend to enlarge dataset with new images (taken from the samples or artificially created using Data Augmentation Techniques) of the analyzed bacteria genera and introduce new bacteria genera images to verify how increasing number of images or analyzed genera may impact overall results. One should test other classification methods and compare their performance in bacteria identification on whole images and on their sub-images. The new approach may include image preprocessing, alternative image segmentation methods coupled with selecting bacteria on images. Lastly, the new approaches to Extreme Learning Machines like usage of Ricker wavelet as a kernel function in Kernel Extreme Learning Machine or Evolutionary algorithms for ELM learning as well as Convolutional Neural Networks may also be incorporated in future research.

REFERENCES

Abdul Samad K.; Zainol N.; Wan Yussof H.; Ahmad Khushairi Z.; Sharif S.; and Syukri N., 2020. *Isolation, identification and characterization of soil bacteria for the production of ferulic acid through co-culture fermentation using banana stem waste. SN Applied Sciences*, 2, no. 339. doi:10.1007/s42452-020-2151-3.

Bakti L.D.; Imran B.; Wahyudi E.; Arwidiyarti D.; Suryadi E.; Multazam M.; and Maspaeni, 2021. *Data extraction of the gray level Co-occurrence matrix (GLCM) Feature on the fingerprints of parents and children in Lombok Island, Indonesia. Data Br*, 36, 107067. ISSN 2352-3409. doi:10.1016/j.dib.2021.107067.

Calzolari M., 2022. *manuel-calzolari/sklearn-genetic: sklearn-genetic 0.5.1.* doi:10.5281/zenodo.5854662.

Chapelle O.; Haffner P.; and Vapnik V., 1999. *Support vector machines for histogram-based image classification. IEEE trans neural netw*, 10, no. 5, 1055–1064. doi:10.1109/72.788646.

Davin-Regli A.; Lavigne J.P.; and Pagès J.M., 2019. *Enterobacter spp.: Update on Taxonomy, Clinical Aspects, and Emerging Antimicrobial Resistance. Clinical Microbiology Reviews*, 32. doi:10.1128/CMR.00002-19.

Dhini A.; Surjandari I.; Kusumoputro B.; and Kusiak A., 2021. *Extreme learning machine – radial basis function (ELM-RBF) networks for diagnosing faults in a steam turbine. J Ind Prod*, 1–9. doi:10.1080/21681015.2021.1887948.

Enez B., 2019. *Identification of bacteria species isolated from soil and investigation of optimum conditions: application in food industry and biotechnological fields. Progress in Nutrition*, 21, no. 2, 467–472. doi:10.23751/pn.v21i2.7985.

Galloway M.M., 1975. *Texture analysis using gray level run lengths. Comput graph image process*, 4, no. 2, 172–179. ISSN 0146-664X. doi:10.1016/S0146-664X(75)80008-6.

Gonzalez R.C. and Woods R.E., 2008. *Digital image processing.* Prentice Hall. ISBN 978-0133356724.

Haralick R.M.; Shanmugam K.; and Dinstein I., 1973. *Textural Features for Image Classification. IEEE Trans Syst Man Cybern*, SMC-3, no. 6, 610–621. doi:10.1109/TSMC.1973.4309314.

Horn R.A. and Johnson C.R., 1985. *Matrix Analysis.* Cambridge University Press. doi:10.1017/CBO9780511810817.

Huang G.B.; Zhu Q.Y.; and Siew C.K., 2006. *Extreme learning machine: Theory and applications. Neurocomputing*, 70, no. 1, 489–501. ISSN 0925-2312. doi:10.1016/j.neucom.2005.12.126.

Iqbal N.; Mumtaz R.; Shafi U.; and Zaidi S.M.H., 2021. *Gray level co-occurrence matrix (GLCM) texture based crop classification using low altitude remote sensing platforms. PeerJ Comput Sci*, 7, no. e536. doi:10.7717/peerj-cs.536.

Konopka A.; Struniawski K.; Kozera R.; Trzciński P.; Sas-Paszt L.; Lisek A.; Górnik K.; Derkowska E.; Głuszek S.; Sumorok B.; and Frac M., 2022. *Classication of Soil Bacteria Based on Machine Learning and Image Processing.* In *ICCS 2022*. Springer International Publishing, Cham. ISBN 978-3-031-08757-8, 263–277. doi:10.1007/978-3-031-08757-8_23.

Kotwal S.; Rani P.; Arif T.; Manhas J.; and Sharma S., 2022. *Automated Bacterial Classifications Using Machine Learning Based Computational Techniques. Arch Comput Methods Eng*, 29, no. 4, 2469–2490. doi:10.1007/s11831-021-09660-0.

Kramer O., 2013. *Dimensionality Reduction with Unsupervised Nearest Neighbors*, Springer Berlin Heidelberg, chap. K-Nearest Neighbors. ISBN 978-3-642-38652-7, 13–23. doi:10.1007/978-3-642-38652-7_2.

Kruk M.; Kozera R.; Osowski S.; Trzcinski P.; Sas L.; Sumorok B.; and Borkowski B., 2016. *Computerized Classification Systemfor the Identification of Soil Microorganisms. Appl Math Inf Sci*, 10, no. 1, 21–31. doi:10.18576/amis/100103.

Laga H.; Shahinnia F.; and Fleury D., 2014. *Image-based plant stornata phenotyping.* In *2014 13th ICARCV*. 217–222. doi:10.1109/ICARCV.2014.7064307.

Lee C.C.; Chung P.C.; Tsai J.R.; and Chang C.I., 1999. *Robust radial basis function neural networks. IEEE Trans Syst Man Cybern Syst*, 29, no. 6, 674–685. doi:10.1109/3477.809023.

Li H.T.; Chou C.Y.; Chen Y.T.; Wang S.H.; and Wu A.Y., 2019. *Robust and Lightweight Ensemble Extreme Learning Machine Engine Based on Eigenspace Domain for Compressed Learning. IEEE Trans Circuits Syst I: Regul Pap*, 66, no. 12, 4699–4712. doi:10.1109/TCSI.2019.2940642.

Liang J.; Hou L.; Luan Z.; and Huang W., 2019. *Feature Selection with Conditional Mutual Information Considering Feature Interaction. Symmetry*, 11, no. 7. ISSN 2073-8994. doi:10.3390/sym11070858.

Novitasari D.C.R.; Lubab A.; Sawiji A.; and Asyhar A.H., 2019. *Application of Feature Extraction for Breast Cancer using One Order Statistic, GLCM, GLRLM, and GLDM. Adv sci technol eng syst j*, 4, no. 4, 115–120. doi:10.25046/aj040413.

Pons M.N.; Vivier H.; Delcour V.; Authelin j.r.; and Paillères-Hubert L., 2002. *Morphological Analysis of Pharmaceutical Powders. Powder Technology*, 128, 276–286. doi:10.1016/S0032-5910(02)00177-8.

Prashanthi R.; Shreevatsa G.; Krupalini S.; and Manoj L., 2021. *Isolation, characterization, and molecular identification of soil bacteria showing antibacterial activity against human pathogenic bacteria. Journal of Genetic Engineering and Biotechnology*, 19. doi:10.1186/s43141-021-00219-x.

Rachmad A.; Chamidah N.; and Rulaningtyas R., 2021. *Classification of mycobacterium tuberculosis based on color feature extraction using adaptive boosting method. AIP Conference Proceedings*, 2329, no. 1, 050005. doi:10.1063/5.0042283.

Rao C.R. and Mitra S.K., 1971. *Generalized Inverse of Matrices and Its Applications.* John Wiley, New York.

Sande K.; Gevers T.; and Snoek C., 2008. *A comparison of color features for visual concept classification.* In *CIVR 2008*. 141–150. doi:10.1145/1386352.1386376.

Srisha R. and Khan A., 2013. *Morphological Operations for Image Processing : Understanding and its Applications.* In *National Conference on VLSI, Signal processing & Communications.*

Steffan J.; Derby J.; and Brevik E., 2020. *Soil Pathogens that may Potentially Cause Pandemics, Including SARS Coronaviruses. Current Opinion in Environmental Science & Health*, 17, 35–40. doi:10.1016/j.coesh.2020.08.005.

Syed-Ab-Rahman S.F.; Carvalhais L.C.; Chua E.; Xiao Y.; Wass T.J.; and Schenk P.M., 2018. *Identification of Soil Bacterial Isolates Suppressing Different Phytophthora spp. and Promoting Plant Growth. Frontiers in Plant Science*, 9. doi:10.3389/fpls.2018.01502.

Waghunde R. and Sabalpara A., 2021. *Impact of Pseudomonas spp. on Plant Growth, Lytic Enzymes and Secondary Metabolites Production. Frontiers in Agronomy*, 3. doi:10.3389/fagro.2021.752196.

Walterson A.M. and Stavrinides J., 2015. *Pantoea: insights into a highly versatile and diverse genus within the Enterobacteriaceae. FEMS Microbiology Reviews*, 39, no. 6, 968–984. ISSN 0168-6445. doi:10.1093/femsre/fuv027.

Yap F.; Bui J.; Knuttinen M.; Walzer N.; Cotler S.; Owens C.; Berkes J.; and Gaba R., 2012. *Quantitative morphometric analysis of hepatocellular carcinoma: Development of a programmed algorithm and preliminary application. Diagnostic and interventional radiology (Ankara, Turkey)*, 19, 97–105. doi:10.4261/1305-3825.DIR.5973-12.1.

Zahran H., 2000. *Rhizobium-Legume Symbiosis and Nitrogen Fixation under Severe Conditions and in an Arid Climate. Microbiology and molecular biology reviews*, 63, 968–989. doi:10.1128/MMBR.63.4.968-989.1999.

3.4.2 Automated identification of soil Fungi and Chromista through Convolutional Neural Networks

Publication:

K. Struniawski, R. Kozera, P. Trzcinski, A. Lisek, and L. Sas Paszt, *"Automated identification of soil Fungi and Chromista through Convolutional Neural Networks"*, Engineering Applications of Artificial Intelligence, Vol. 127B, p. 1–12, 2024, doi: 10.1016/j.engappai.2023.107333 .

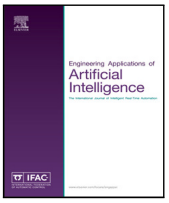
Abstract: The identification of soil microorganisms plays a crucial role in agriculture and horticulture, as it enables the monitoring of beneficial species and early detection of pathogens. In this study, we propose a system that utilizes machine vision and machine learning techniques, specifically Convolutional Neural Networks, to automate the identification of different fungi and Chromista based on microscopic images and morphological traits. Our system aims to provide a cost-effective and efficient method for pathogen detection, improving the overall health and productivity of agricultural systems. We conducted experiments using a dataset of soil microorganisms and the performance of the classifier was evaluated using precision, recall and F1-score measures. Despite challenges such as class imbalance and imperfect subimage retrieval, the classifier achieved promising results, with an overall precision of 82% indicating the high accuracy of correctly predicted positive instances across all classes. Furthermore, the incorporation of a majority voting scheme significantly improved the classifier's performance, addressing the issue of underrepresented classes. The enhanced results demonstrated an average precision and F1-score of 97%. Our work highlights the potential of CNNs in soil microorganism identification and paves the way for future research to expand the dataset and to incorporate a wider range of microorganism genera.

Engineering Applications of Artificial Intelligence 127 (2024) 107333

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Automated identification of soil Fungi and *Chromista* through Convolutional Neural Networks☆

Karol Struniawski [a,*], Ryszard Kozera [a,b], Pawel Trzcinski [c], Anna Lisek [c], Lidia Sas Paszt [c]

[a] Institute of Information Technology, Warsaw University of Life Sciences - SGGW, ul. Nowoursynowska 159, Warsaw, 02-787, Mazowieckie, Poland
[b] School of Physics, Mathematics and Computing, The University of Western, Australia, 35 Stirling Highway, Perth, WA 6009, Western Australia, Australia
[c] Department of Microbiology and Rhizosphere, The National Institute of Horticultural Research, Pomologiczna 18, Skierniewice, 96-100, Lodzkie, Poland

ARTICLE INFO

Keywords:
Soil microorganisms
Agriculture
Horticulture
Machine vision
Machine learning
Convolutional neural networks

ABSTRACT

The identification of soil microorganisms plays a crucial role in agriculture and horticulture, as it enables the monitoring of beneficial species and early detection of pathogens. In this study, we propose a system that utilizes machine vision and machine learning techniques, specifically Convolutional Neural Networks, to automate the identification of different fungi and Chromista based on microscopic images and morphological traits. Our system aims to provide a cost-effective and efficient method for pathogen detection, improving the overall health and productivity of agricultural systems. We conducted experiments using a dataset of soil microorganisms and the performance of the classifier was evaluated using precision, recall and F1-score measures. Despite challenges such as class imbalance and imperfect subimage retrieval, the classifier achieved promising results, with an overall precision of 82% indicating the high accuracy of correctly predicted positive instances across all classes. Furthermore, the incorporation of a majority voting scheme significantly improved the classifier's performance, addressing the issue of underrepresented classes. The enhanced results demonstrated an average precision and F1-score of 97%. Our work highlights the potential of CNNs in soil microorganism identification and paves the way for future research to expand the dataset and to incorporate a wider range of microorganism genera.

1. Introduction

The use of soil fungi as a natural fertilizer is a growing area of interest due to their potential in increasing crop yield and in replacing chemical fertilizers, which are still widely used in agriculture and horticulture (Han et al., 2021; Fan et al., 2023). Certain soil fungi and *Chromista* have a mutualistic relationship with plants, where they exchange nutrients with their host plant. These beneficial fungi, known as mycorrhizal fungi, play a crucial role in promoting the health and growth of many plant species (Smith and Read, 2010). Mycorrhizal fungi can enhance nutrient uptake, improve plant growth and increase resistance to environmental stressors e.g. drought and extreme temperatures. However, some soil fungi can be pathogenic and cause significant crop losses (Ma et al., 2013). Therefore, it is crucial to accurately identify soil microorganisms to monitor beneficial species and to detect pathogens early.

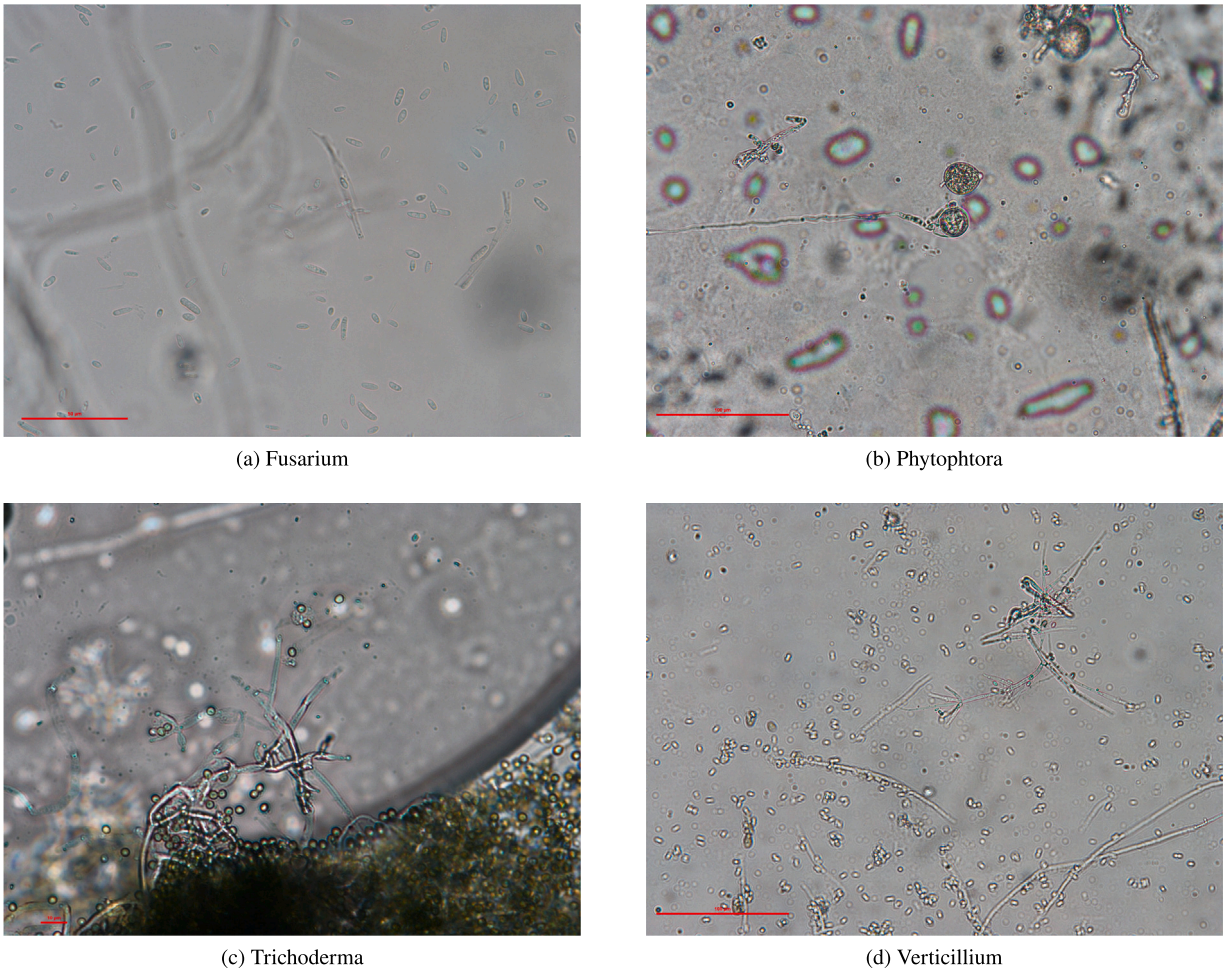
Microbial identification typically involves a combination of morphological, phenotypic and molecular methods. Morphological identification is the most cost-effective approach and involves microscopic observation of the microorganism, where a microbiology specialist identifies the organism based on characteristic fragments (Watanabe, 2010). However, identification based on morphological traits alone is limited to the genus level due to the subtle differences between species. Therefore, microbiologists typically use a combination of different methods for accurate identification.

In this study, we propose the use of machine vision and machine learning techniques, specifically Convolutional Neural Networks (CNN), to identify different fungi and *Chromista* based on microscopic images and their morphological traits. This system aims to automate the identification process, making it faster and more efficient. While the current system is limited to genus-level identification, it can be enhanced in future by incorporating phenotypic and molecular methods.

☆ This research was supported by The National Centre for Research and Development, Poland within the framework of the project BIOSTRATEG, grant number BIOSTRATEG3/344433/16/NCBR/2018.

* Corresponding author.
E-mail addresses: karol.struniawski@sggw.edu.pl (K. Struniawski), ryszard_kozera@sggw.edu.pl (R. Kozera), pawel.trzcinski@inhort.pl (P. Trzcinski), anna.lisek@inhort.pl (A. Lisek), lidia.sas@inhort.pl (L. Sas Paszt).

https://doi.org/10.1016/j.engappai.2023.107333
Received 16 June 2023; Received in revised form 22 August 2023; Accepted 17 October 2023
Available online 22 October 2023



(a) Fusarium (b) Phytophtora

(c) Trichoderma (d) Verticillium

Fig. 1. Microscopic images of cultivated samples obtained from the dataset sourced from *Symbio-Bank* - the microorganism collection at The National Institute of Horticultural Research in Skierniewice, Poland.

Such integration would enable identification at the species level, albeit at a higher cost.

The main objective of our system is to provide in future highly accurate results that can be utilized by farmers to quickly detect potential pathogens and to supply solutions that can be implemented immediately upon the soil's sample delivery to local laboratories. With its low-cost laboratory equipment and full automation process such system becomes highly desirable for the agricultural industry worldwide. System's ability to quickly detect potential threats can minimize the risk of crop losses and can improve the overall productivity of agricultural setups. This aspect is particularly significant taking into consideration recommendations of the United States Environmental Protection Agency that emphasize the importance of implementing methods to enhance crop resistance and to employ intelligent systems addressing the impacts of climate change on agriculture and food supply (U.S. Environmental Protection Agency, 2023). The detection of Mycorrhizal microorganisms enables precise tracking of natural fertilizer application, monitoring their quantity and effects on the soil. This paper describes our initial work in constructing such a system using CNNs and we believe that the system described above represents the ultimate goal for the future.

Identification of soil fungi and *Chromista* using conventional image processing methods (Gonzalez and Woods, 2018) poses significant challenges due to their intricate and diverse structures, which exhibit non-homogeneity unlike soil bacteria (Watanabe, 2010). These structures encompass various components such as hyphae, phialides, micro and macroconidia, conidia cells, zoospores and more. To address the latter our research explores a dataset sourced from *Symbio-Bank*, which houses a collection of microorganisms at The National Institute of Horticultural Research in Skierniewice, Poland. Within this collection, we specifically select, cultivate and capture microscopic images of microorganisms belonging to the genera *Fusarium*, *Trichoderma*, *Verticillium* and *Chromista* of the *Phytophthora* genus. It is important to note that each image in the dataset was assumed to contain only one type of microorganism, obtained through a rectification process that isolates the monoculture from the sample. The dataset comprises 134 images of microorganisms cultivated in Potato Dextrose agar (Merck 1.10130.0500) at a temperature of 26 degrees Celsius. Incubation periods varied, with *Fusarium*, *Verticillium* and *Phytophthora* microorganisms requiring 7–10 days, while the *Trichoderma* strain took approximately 4–5 days to produce conidial spores.

The biomass was collected from the colonies using sterile swabs and spread over the microscope slide. A small amount of distilled water was added and the material was covered with a coverslip. The images were taken using a Nikon 80i microscope and no manual post-processing was performed. Therefore, the images may contain artifacts like light reflections or air bubbles under the coverslip, which could resemble the microorganism's structures. The dataset comprised 57 *Fusarium*, 26 *Trichoderma*, 31 *Verticillium* and 20 *Phytophthora* images, which were obtained with different magnification levels for each of the microorganism genera. Fig. 1 presents sample images from the dataset. A portion of the whole dataset is provided as supplementary material (Struniawski, 2023).

The work starts with the conducting a literature review to identify the current state-of-the-art solutions in the area in question and to determine the direction addressed in our experiments.

Wahid et al. (2018) presented a transfer learning approach where the Inception CNN was retrained using a dataset consisting of five different bacteria species microscopic images known to be harmful to human health. Increasing the total number of samples, multiple bacteria samples were extracted from each image through a manual cropping. However, a major limitation of the system was its reliance on human involvement in the manual input image processing, which rendered the system as a hybrid and not fully automated, thus making the task more laborious.

Zhang et al. (2017) utilized CNNs to cervical cell classification. The proposed approach involves segmenting single cell instances from the original image, which can be challenging due to the location of the nucleus within the cell not always being in the center position. In the case of images containing only cells and background, the task involves pre-segmentation by retrieving subimages based on the observable nucleus within the cell boundaries. These subimage patches are then exploited to train the CNN for cervical cell classification without the need for accurate cell segmentation, as the CNN with a large dataset is recognized as a better generalizer even with inaccurate segmentation. This approach can potentially yield better robustness of the classifier. The limitation of the system is its strict dependency on a coarse nucleus center being provided. To overcome this limitation, the researchers employed transfer learning by the pre-trained CNN on the ImageNet.

Upon examining the research conducted by Ong et al. (2020), it was observed that they employed transfer learning with the ResNet50 CNN model to detect *Fusarium* fungi in laboratory-cultivated samples. The images were captured using a microscope, utilizing three distinct microscopy configurations: Brightfield, Darkfield and Fluorescent, where each configuration provided unique image characteristics. The authors trained three separate CNN models, where each model is specialized in classifying one of the three image modes. To classify the different modes, the authors employed the respective ResNet50 model and a logistic regression classifier was utilized to obtain the final prediction. This study represents a rare example of using CNNs for soil fungi identification, focusing on binary classification for detecting the presence of *Fusarium* in the images. In contrast, our research aims to differentiate various fungi and *Chromista* genera from one another. Noteworthy, the study by Ong et al. employed a manual segmentation method, while our objective is to develop a fully automated system.

2. Methods

The paper provides a comprehensive description of the subimage retrieval process and the application of CNN for the classification task. The subimage retrieval process is elaborated upon, highlighting the methodology employed to extract relevant fragments from the original images. Then, the paper delves into the CNN approach used for classifying the retrieved subimages, discussing the architecture and techniques employed to achieve accurate classification results.

2.1. Subimages retrieval

The microscopic images used in our research contain scale labels in the left-down corner, which are added automatically by the microscope's software. These labels come in three different colors – black, red, or white – selected based on image metrics. However, the presence of these scale labels may introduce distortions during the subimage retrieval process, ultimately impacting performance of the predictor. To mitigate this issue, we masked the label's area using manual thresholding techniques (Gonzalez and Woods, 2018). The threshold values were determined empirically and the mask's area was defined as the set of pixels meeting the following conditions: $r < 30$ and $g < 30$ and $b < 30$ (black label), or $r > 150$ and $g < 125$ and $b < 125$ (red label), or $r > 240$ and $g > 240$ and $b > 240$ (white label), where r, g and b denote the red, green and blue components of the RGB pixel $I = [r, g, b]$, respectively. The resulting binary mask identifies the label area with a value of 1, while the rest of the image is assigned a value of 0. Finally, the label's area is filled with the mean color of the image and merged with the input image to create a new, label-free image for analysis.

In our study, we present an image preprocessing pipeline for enhancing the quality of microscopic images to facilitate subsequent analysis. First, we converted the RGB image to grayscale using the COLOR_BGR2GRAY mode provided by the popular cv2 package in Python. To improve the quality of the image, we applied the widely-used Gaussian blur technique (Gonzalez and Woods, 2018), which effectively smooths out small-scale variations in the image while preserving larger-scale features. A Gaussian kernel with a window size of 5×5 was used in our approach, which has been shown to be an effective choice for reducing image noise while preserving important features (Párraga et al., 1998).

Microscopic images often suffer from low contrast, which can make it difficult to discern the relevant features and structures in the image. To address this, a variety of contrast enhancement techniques have been proposed in the literature, which modify the spatial characteristics of the image to improve its visibility and facilitate image analysis (Cakir et al., 2018). In our work, we employ the Contrast-limited Adaptive Histogram Equalization (CLAHE) method, which has been shown to be effective in enhancing the visibility of structures in low-contrast images. This method has been successfully applied in a variety of applications, such as improving the accuracy of mammographic image classification (Alshamrani et al., 2023) and detecting pneumonia using CNNs (Tjoa et al., 2022). By applying CLAHE to our microscopic images, we aim to improve the performance of our machine learning-based subimage retrieval method by increasing the distinguishability between different fungi structures. On the grayscale image obtained in the previous step the CLAHE method is applied using a clip limit of 0.1 and a tile grid size of 8×8. The clip limit is a threshold value that limits the maximum pixel intensity value in the histogram, while the tile grid size refers to the size of the tile used for local histogram equalization that affects the level of local contrast enhancement.

Segmentation of microscope images is a challenging task due to uneven illumination, lightning variations and noise (Pham et al., 2000). Global thresholding techniques like the Otsu method (Otsu, 1979) are commonly used for segmentation, but they may produce unreliable results when the image is complex and its histogram cannot be segmented by a single threshold value. Our literature review reveals that local adaptive thresholding techniques, such as the one proposed by Wen-Nung Lie and other researchers (Lie, 1995), have been found to be superior to global thresholding techniques for segmenting microscopic images (Khan et al., 2015; Dave and Upla, 2017). Therefore, we applied the Adaptive Image Thresholding algorithm, implemented in Python's cv2 adaptiveThreshold function, with a 9-pixel block size that determines the size of the neighborhood area and a constant of $C = 213$. The threshold value is a Gaussian-weighted sum of the neighborhood values minus the constant C. The adaptive threshold method creates a matrix of thresholds equal to the size of the input image that are based on the local neighborhood of the image pixels. The details of implementation can be found in the original paper (Lie, 1995).

It is important to highlight the significant challenges associated with soil microorganism images, including low contrast, out-of-focus objects, sample contamination and image impurities such as light reflection from the coverslip. It should be noted once the sample is observed under a microscope, the microorganisms exist in a three-dimensional space and the focus of the camera is limited to a specific level. As a result, objects located at different depths may appear out of focus. To tackle these challenges and further enhance the quality of the images, we incorporate morphological operations and filling algorithms into our pipeline. This approach aligns with other studies in the field, which

involve converting microscopic images to grayscale, preprocessing, segmenting and employing morphological operations to eliminate small objects or fill gaps within objects (Mohamad et al., 2014). By applying these techniques, we aim to improve image quality and ensure accurate analysis of the microorganisms in our research.

In our system, we begin by dilating the images using an 11×11 kernel, followed by applying a flood fill algorithm and subsequently performing open and close operations with a 5×5 kernel. We then perform a second flood fill and obtain the final image, from which subimages can be retrieved in subsequent steps. The algorithm used in our study is based on the implementation described in Soille (1999).

In the subsequent stage, the image is labeled by identifying all objects, also referred to as connected components, using the label function from the skimage module. Two pixels are considered connected if they are neighbors and have the same value. It is important to note that we are still operating on the binary image mask, where the pixels can be neighbors in a 1- or 2-connected sense and the value denotes the maximum number of orthogonal hops to consider a pixel as a neighbor (Fiorio and Gustedt, 1996). Consequently, the labeled image is obtained with k objects identified, where the image comprises of pixels with values of either 0, representing the background, or 1 up to k, identified as the objects. Next, utilizing the regionprops function from Python's scikit-image module, we are able to compute the parameters of each object. Here, we used the solidity, bounding box and area measures. Solidity is the ratio of the region's area to the area of its convex hull, which is the smallest convex polygon that can contain the region (Ilonen et al., 2018). The bounding box is the smallest box that contains the entire object, defined by its pixel coordinates on the image and the area is defined as the number of pixels in the region. It is noteworthy that solidity, along with other region properties, is recognized as an accurate descriptor of objects in an image (Vinnett et al., 2022; Acuña et al., 2016; Bailey et al., 2005). The output of the labeling and region properties calculation is a list of all objects in the image with assigned measures.

In the next stage, our objective was to accurately select objects that are fragments of microorganisms, which posed a difficult challenge due to the heterogeneity of soil fungi and *Chromista* organisms, which cannot be identified based on a single cell but instead rely on specific morphological features such as conidiophores, sporodochia, pycnidia, sclerotia, mycelia, etc. Wijayawardene et al. (2020). Retrieving single bacterial cells in microscopic images is straightforward since they are represented by uniformly sized circular cells. Consequently, recent works focus on filtering out objects with solidity lower than a predetermined threshold and retrieving n objects closest to the median area of the objects (Struniawski et al., 2022). However, this approach did not yield satisfactory results in our dataset. After conducting several experiments, we propose a pipeline that specifies a padding parameter p (in our case, $p = 3$) to address the issue of losing crucial information on object edges during processing images phase. To accomplish this goal, we remove all objects that would exceed the image's size after padding and those with solidity lower than 0.8. We also leave out images with standard deviation of pixel luminosity less than 5 to eliminate low contrast objects, which could be mistaken for background or out-of-focus fragments.

The next challenge is encountered when the number of retrieved objects is significantly less than the maximum target number of instances (200 in our case), usually due to uneven lighting conditions or the entire image being out of focus. In such instances, we avoid filtration based on solidity and instead selecting 100 objects with the highest solidity value. Furthermore, we sort objects based on the covered area and select a maximum of 200 objects with the largest area. Once the filtering process is complete, we retrieve single images by cutting the area around the object's bounding box with additional padding. Next we resize the sub-image's mask to match it's subimage size by eroding it with a kernel of size $(2p + 1) \times (2p + 1)$ to mitigate the effect of the enlarged boundaries that may contain background information. This erosion operation helps to remove the background information consistently throughout the boundary of the object. Finally, we filter the object's final image by taking the Hadamard product (Szeliski, 2011) of the sub-image with its corresponding mask, followed by placing such image in the center of 224×224 image with all pixel values set to 0 (black image). Employing the Hadamard product of the object with its corresponding mask alleviates the impact of overlapping objects that may arise following the retrieval process from the original image. This strategic approach guarantees that each subimage exclusively encompasses a single object, a direct outcome of the labeling procedure applied to the input image. The result is a black background image with the color object in the center. It is worth noting that in literature, better classification performance is achieved by using color object images with a black background, which eliminates irrelevant background information and reduces unnecessary distortions (Konopka et al., 2022; Struniawski et al., 2022; Gao et al., 2017). The data flow of the algorithm is presented in Fig. 2.

The generalization abilities of the system forms a significant challenge. Although the aforementioned algorithm demonstrates accurate subimage retrieval for certain genera, its performance may be suboptimal for others. Our studies have revealed that setting the threshold value of solidity filtration to 0.9, instead of 0.8, led to unsatisfactory outcomes of sub-images generation for most images of *Phytophthora* genera. Consequently, the filtration conditions must be as weak as possible to increase the robustness of sub-image computations. However, this approach may result in sub-images that are not actually fragments of microorganisms, but represent in fact noise or background. To address this issue, we have incorporated a majority voting rule after classification, which is further discussed in details (see Section 2.4.2).

After analyzing the generated dataset, we can observe that the system achieves accurate retrieval of subimages for all the examined genera. The filtration process parameters, including image processing kernel sizes and shapes, threshold values, padding and the maximum number of retrieved objects, can be adjusted using a new dataset of soil microorganisms. The stepwise retrieval process is illustrated in Fig. 3, while Fig. 4 showcases exemplary retrieved sub-images for each genus. Additional subimages can be found in the supplementary materials (Struniawski, 2023). The presented system signifies a significant progress towards fully automated analysis of soil microorganisms. It is important to note that the sizes of objects in the images vary due to differences in magnification levels employed during image acquisition. The dataset utilized for training Machine Learning methods for single instance retrieval comprises a total of 10,175 images, including 4278 images of *Fusarium*, 1285 images of *Phytophtora*, 2352 images of *Trichoderma* and 2260 images of *Verticillium*.

2.2. ResNet50 neural network

In our research, we have utilized Residual Network 50 (ResNet50) as a Machine Learning classification model. ResNet50 is a type of CNN that has demonstrated outstanding performance across various domains. Unlike traditional machine learning methods, which rely on manually crafting features in advance (called also as handcrafted features) and feeding them into a chosen classifier such as Support Vector Machines, Random Forests, or Multilayer Perceptron (Hanbal et al., 2020).

In classical ML approaches, the manual determination of features requires domain experts to carefully consider the task's characteristics and evaluate the selected features. This process often leads to suboptimal accuracy, if chosen features are believed to adequately represent the observed phenomenon. It typically involves multiple iterations of feature determination, followed by the application of feature selection methods to form a final set of features. These selection methods aim to choose or filter features that exhibit strong correlation with the target variable and weak correlation with each other. In contrast, features can be automatically calculated from the data using Convolutional

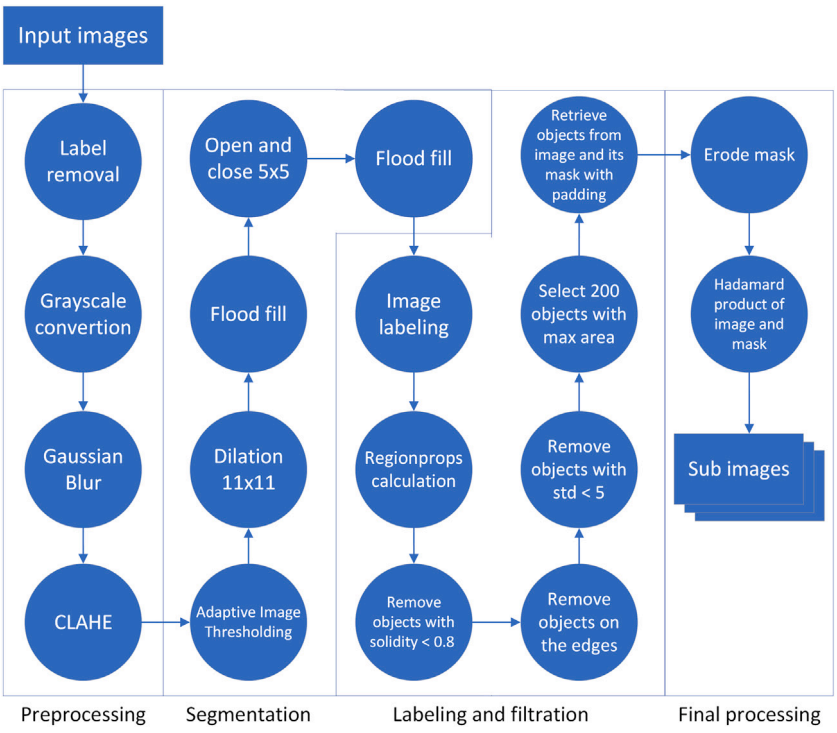


Fig. 2. The data flow diagram illustrating the proposed algorithm for sub-images retrieval.

Neural Networks. CNNs employ convolutional layers with learnable filters to convolve input data, capturing relevant patterns and spatial relationships automatically (Yamashita et al., 2018). For image data, 2D convolutional layers are commonly used, resulting in 2D feature maps. CNNs also incorporate pooling layers, which downsample feature maps and promote translation invariance. This characteristic reduces the network's sensitivity to the precise spatial location of features.

CNNs consist of multiple building blocks, including convolution layers, pooling layers and, in the final block, feature maps are flattened into vectors. This process converts each observation in the dataset into a vector of values that can be recognized as features. Typically, a fully connected (dense) feedforward network with a backpropagation algorithm is employed to train the weights within the dense network, as well as the filter coefficients in the convolutional layers. The training process minimizes a given loss function, enabling the network to learn discriminative representations from the data. The depth of a CNN plays a crucial role in its performance on various tasks and significant advancements have been achieved by employing very deep models (Simonyan and Zisserman, 2015; He et al., 2015). However, increasing the depth of the network by stacking more blocks and expanding the number of layers introduces challenges such as dealing with vanishing or exploding gradient and degradation problems. The vanishing and exploding gradient issues can be addressed through normalized training initialization techniques (LeCun et al., 1998). On the other hand, the degradation problem arises when adding extra layers to a deep architecture leads to higher training error.

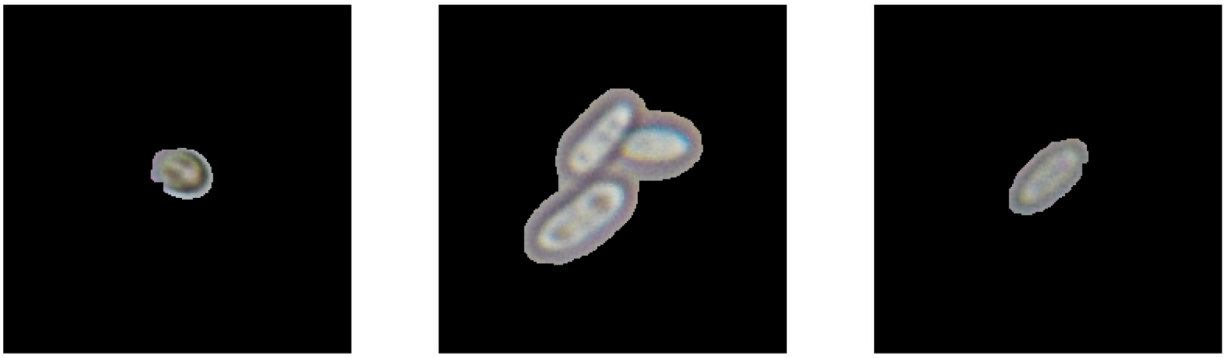
To overcome these challenges ResNets have been introduced (He et al., 2016). They utilize residual connections, also referred to as skip or shortcut connections that allow the network to bypass one or more convolutional layers and directly add the original input to the output of these layers. By incorporating residual connections, ResNets mitigate the degradation problem and enable more effective training.

In this work, we utilize ResNet50, a variant of the ResNet architecture, which has shown remarkable performance in similar tasks (Ong et al., 2020). The ResNet50 implementation employed in this study is provided by the TensorFlow module. ResNet50 is composed of a total of 50 layers (this is the root of 50 in its name), with 48 convolutional layers, one max pool layer and one average pool layer. The architecture consists of five residual blocks, each containing nested convolutional layers, which can be viewed as a combination of convolution, batch normalization and activation operations. While it is not feasible to depict the entire ResNet architecture within the constraints of this paper, we have included it as Supplementary Material (Struniawski, 2023). Fig. 5 showcases a sample of the second residual block with the third convolutional block, where all the aforementioned operations with residual shortcut connection, are visible.

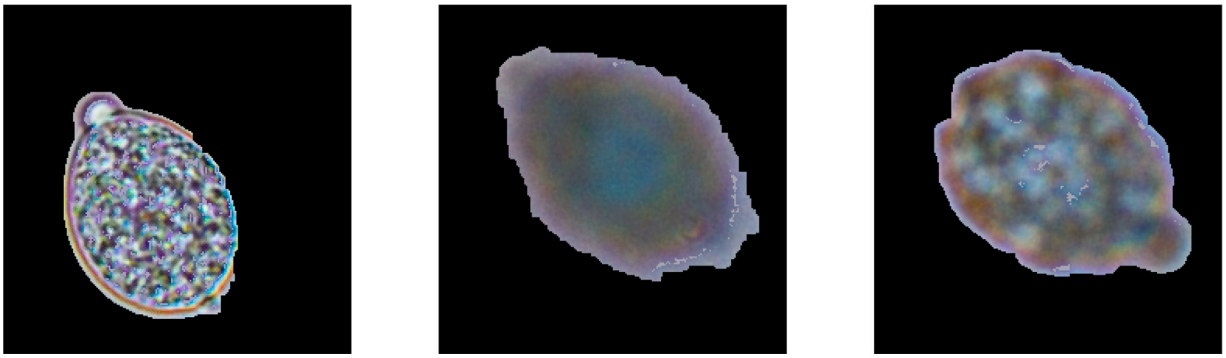
2.3. Transfer learning

Transfer learning is a popular approach in Machine Learning where a model developed for one task is utilized for another, transferring the knowledge gained from the first task to the second one. The idea of transfer learning dates back to 1976 when it was first applied to the perceptron algorithm (Bozinovski and Fulgosi, 1976). CNNs have become increasingly complex, leading to resource-intensive training and transfer learning has become a widely used technique to address this issue. Huge CNNs are typically trained on high-performance computing systems such as clusters, supercomputers, or cloud-based platforms that are optimized for deep learning workloads, with multiple GPUs that parallelize computations and speed up the training process (Huerta et al., 2020). For instance, some of the largest CNN models, including VGG-16, ResNet and Inception, were trained on the ImageNet Large Scale Visual Recognition Challenge dataset, which comprises 1.2 million high-resolution images and took several days to weeks to train. Training a large CNN model from scratch is usually infeasible due to the long training time and high cost. Therefore, we utilized the ResNet50, which had been previously trained on the Imagenet dataset (Deng et al., 2009).

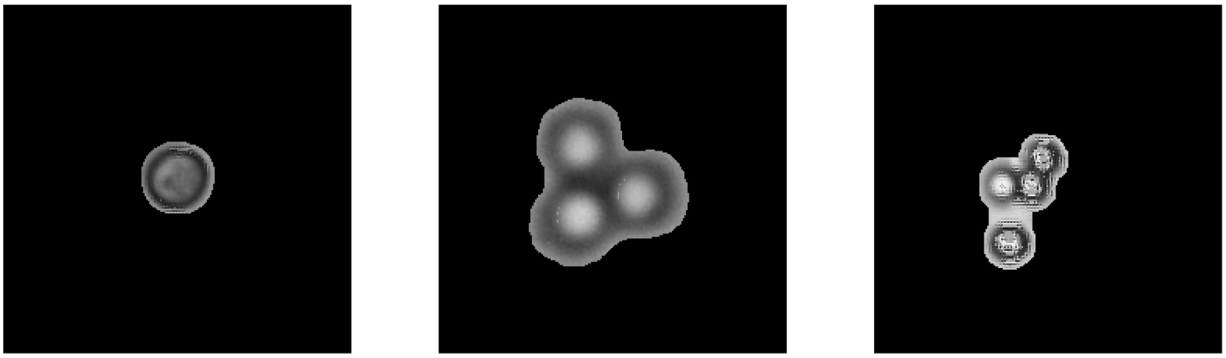
In spite of the fundamental disparity between the ImageNet dataset and microscopic or X-ray images, transfer learning has demonstrated remarkable efficacy in diverse applications, including the classification of COVID-19 from chest X-ray images (Hossain et al., 2022) and the classification of Malaria Cell-Images (Arrabelly and Juliet, 2019). The utilization of the extensive knowledge amassed from the ImageNet



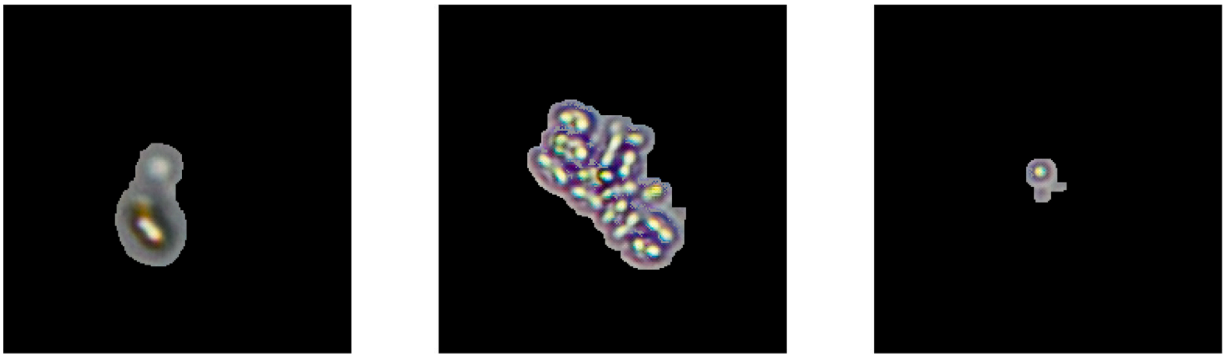
(a) *Fusarium*



(b) *Phytophtora*

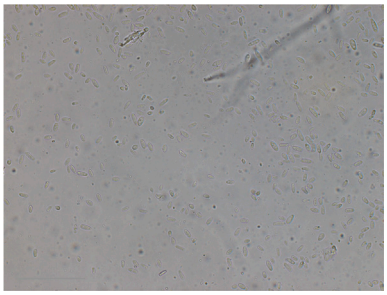


(c) *Trichoderma*

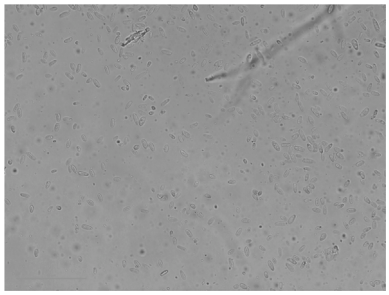


(d) *Verticillium*

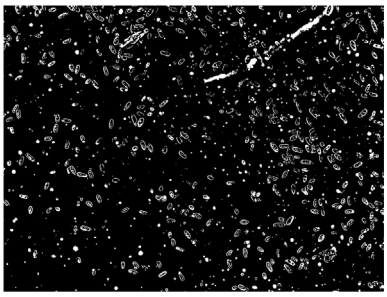
Fig. 3. The sample sub images retrieved from the original dataset from a given genera of soil microorganism.



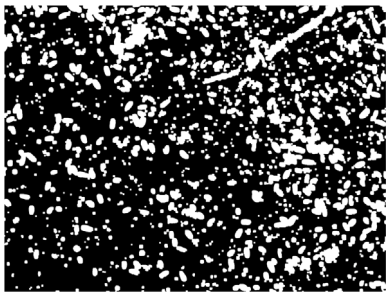
(a) *Fusarium* input image



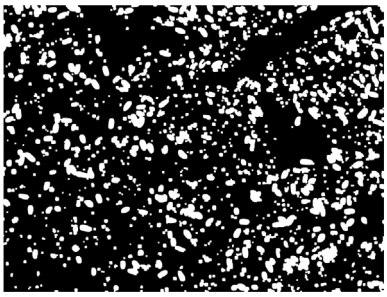
(b) Image after preprocessing of *(a)*.



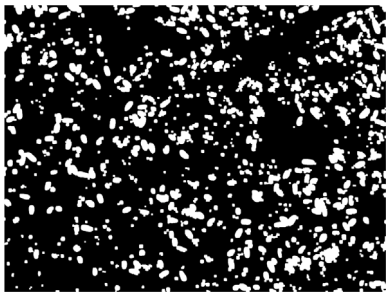
(c) Effect of adaptive thresholding of image *(b)*.



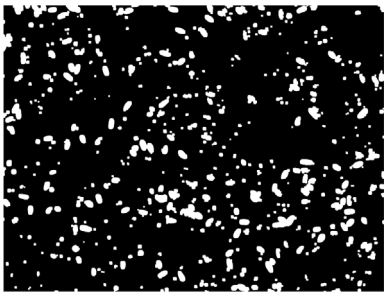
(d) Morphological operations and flood fill of *(c)*.



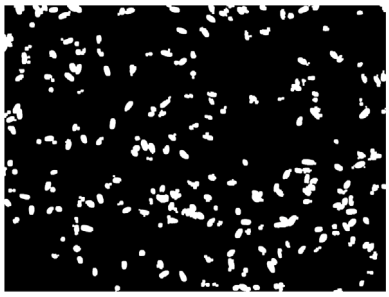
(e) Result of removal objects that are greater that either dimension of image 224 × 224 applied to *(d)*.



(f) Removal of objects with low contrast from *(e)*.



(g) Objects with solidity lower than 0.8 being filtered out from *(f)*.



(h) Maximum 200 objects being selected from *(g)* with greatest area covered.

Fig. 4. Retrieval process of subimages from *Fusarium* microscopic image.

dataset, spanning a wide range of object categories, has proven the remarkable breakthroughs in these respective domains. By harnessing the acquired knowledge from ImageNet, researchers are able to achieve significant advancements in ML area.

In the context of transfer learning, several techniques can be employed. One approach is to initialize the network with pre-trained weights, such as ResNet50 weights established during trained on the ImageNet dataset. Network is then exposed to new images and training is performed in batches. The objective is to fine-tune the network's weights slightly, allowing it to adapt to a new task, such as soil microorganism classification. Another strategy involves freezing specific blocks within the network during training, preventing their weights from being updated through backpropagation.

Determining which blocks to freeze is a crucial decision. As information propagates through the network, the convolutional layers learn to detect higher-level features that become increasingly specific to the given task (Yamashita et al., 2018). Therefore, it is advisable to freeze the initial layers responsible for low-level features that are not task-specific. In our experiments, we chose to freeze all blocks except for the fifth residual block, which includes all convolutional layers named in the ResNet50 topology starting from *conv5_block1_1_conv*. This allows us to fine-tune the high-level feature maps, tailoring them to our specific

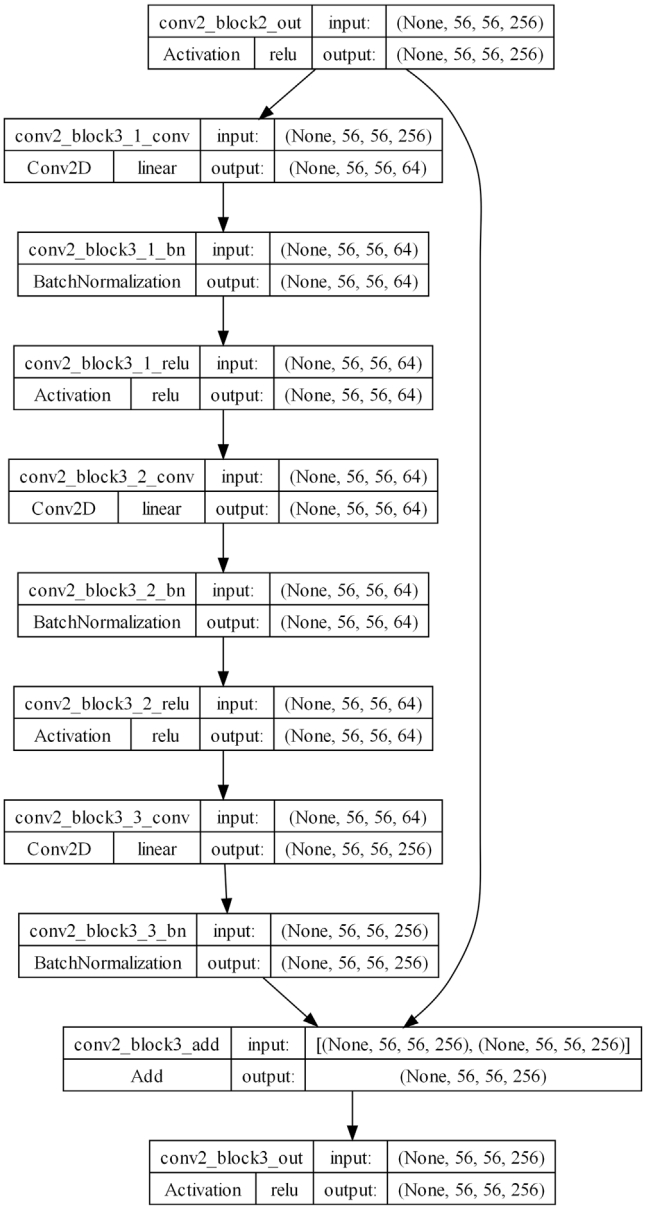


Fig. 5. Fragment of the ResNet50 network topology.

task. By selectively freezing the appropriate blocks and enabling fine-tuning of high-level features, our system can effectively leverage the knowledge gained from pre-training on ImageNet while adapting to the soil microorganism classification task.

2.4. Model training and evaluation process

To validate the statistical significance of our application, we provide a concise overview of the training and evaluation process using transfer learning with ResNet50.

2.4.1. Training

The dataset consisting of 134 microscopic images is partitioned into distinct training, validation and testing sets, with a ratio of 0.6:0.1:0.3 respectively. The training set is employed to feed the network with data during the training process, while the current training loss is evaluated on the validation dataset. The testing set is reserved for assessing the performance of the trained model. Upon partitioning the dataset, subimages are generated, resulting 7121 images in the training set, 1020 in the testing set and 2034 in the validation set. In total, the dataset comprises 10,175 subimages.

For the training process, a configuration is established utilizing the ResNet50 network. The trainable fifth residual block is employed, followed by average pooling and flattening operations. On top of this, a classifier is constructed, consisting of a dense layer with 512 neurons and a dropout rate of 0.3. During training, the dropout mechanism randomly drops 30% of the connections between neurons, thereby preventing overfitting of the classifier. Furthermore, the same elements are added once more, leading to an output dense layer with 4 neurons, corresponding to the number of target classes associated with the soil microorganism genera. In other words topology of the classifier can be described as 512-DropOut-512-DropOut-4. A *softmax* activation function is applied to the output layer to obtain probability distributions over the classes to select affiliation the given class by the highest probability value for a given output neuron. Please refer to Fig. 6 for a visual representation of the classification architecture.

During the training of the network, the categorical cross-entropy loss function is employed, which is a common choice for multi-class classification tasks (LeCun et al., 2015). The utilized optimizer is *Adam*, with a learning rate set to 0.001 (Kingma and Ba, 2014). The training process is conducted with a batch size of 64 images over 1000 epochs. It is worth mentioning that the model was also evaluated for 5000 epochs, but it yielded inferior results. It should be noted that the input dataset was too large to be loaded directly into both the RAM and GPU memory. In this context, the Dataset functionality within the TensorFlow module played a pivotal role, closely followed by the preprocessing step of normalizing the loaded sub-images (ranging from $[0;255]$ to $[0;1]$) for each color channel. The Dataset functionality facilitates the random retrieval of data from memory, allowing for the feeding of batches during the fitting process of the CNN. The whole process is performed on PC computer with Ryzen 9 3900X CPU, 64 GB DDR4 3600MHz RAM and Nvidia RTX 3090 24 GB VRAM GPU using CUDA and cuDNN.

2.4.2. Evaluation

In this study, the trained model from the previous step was subjected to evaluation on the testing set. The classification of individual subimages reflects the effectiveness of the subimage retrieval and CNN training processes. The practical objective is to classify the entire image into the appropriate class, providing an answer to the question of which microorganism is present in the sample. To achieve this, the results obtained from assigning subimages to classes are combined using a majority voting scheme. To illustrate the latter, let us consider the case of the first test image I_t^1 with 150 subimages retrieved. Next, images are fed as input to the pre-trained network in batches, resulting in predictions for each subimage. For instance, the results indicate that 100 subimages are assigned to class C_0, 20 subimages to C_1, 10 subimages to C_2 and 20 subimages to C_3 respectively. By employing the majority voting strategy, the final classification of image I_t^1 is determined based on the class with the highest number of "votes" from the subimages. In this example, I_t^1 is classified as C_0 since it received 100 "votes" for class C_0.

2.5. Preliminary experiments

In the results section, we showcase the outcomes of the most successful setup noted during our experimentation process. Prior to this, we conducted a series of experiments aimed at refining the optimal model configuration. Our initial objective was to determine the most effective method of sub-image retrieval. The comprehensive process, elucidated in Section 2.1 and illustrated in the data flow diagram in Fig. 2, is an outcome of preliminary trials where we assessed various dataset retrieval approaches. In the course of this experimentation, we generated 18 distinct datasets, focusing on factors such as solidity and

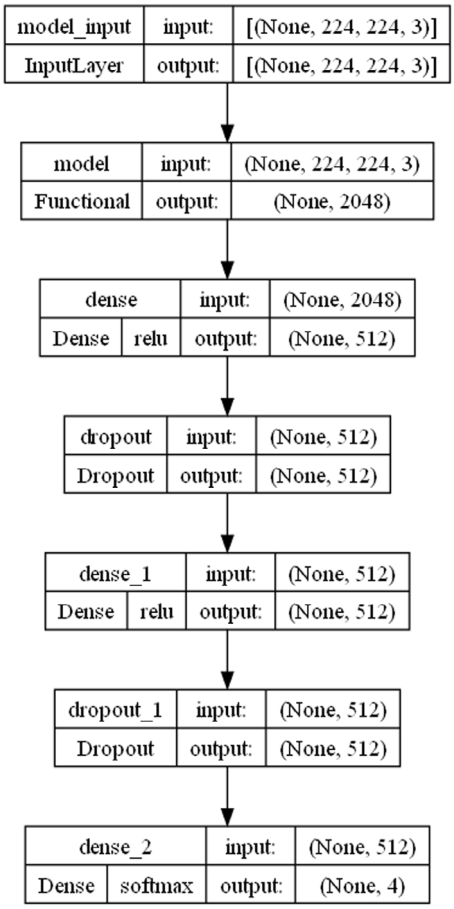


Fig. 6. The ResNet50 classifier, where model_input states as input layer, model as ResNet50 feature extractor and then two classification layers with 512 neurons each and 0.3 dropout rate.

standard deviation filtration thresholds, the number of retrieved sub-images, selection criteria for the designated number of sub-images (including methods like choosing images closest to the median object size or selecting top objects with the highest covered area), and the evaluation of whether a second flood-fill process yielded improved results. Given the impracticality of presenting the detailed results for various sets, including thorough explanations of their nuanced distinctions, we chose to focus on the best sub-image retrieval method.

Subsequently, we proceeded to construct the model and were confronted with the task of fine-tuning hyperparameters for CNN compilation. This encompassed considerations such as the choice of loss function, optimizer along with learning rate, topology of the top layer (classification layers atop convolutional layers), dropout layer rates, batch size, and the selection of the block from which weights would be made trainable for Transfer Learning. Our experimentation encompassed the following configurations:

- v1: trainable from $conv5\_block1\_1\_conv$, top topology: Dense(512, *relu*) x Dropout(0.3) x Dense(512, *relu*) x Dropout(0.3), Categorical Crossentropy loss function, Adam(learning rate 0.001) optimizer, batch size of 32.
- v2: v1 trainable from $conv4\_block1\_1\_conv$.
- v3: v1 trainable from $conv3\_block1\_1\_conv$.
- v4: v3 with batch size of 64.
- v5: v3 with batch size of 256.
- v6: v3 with batch size of 128 and topology of the top layer: 256 x 0.3 x 256 x 0.3.

Table 1
The classifier performance results on different variants of models.

Model version	Precision	Recall	AUC	TOP 1 ACC
v1	**0.710**	**0.698**	**0.875**	**0.706**
v2	0.706	0.705	0.834	0.705
v3	0.684	0.684	0.801	0.684
v4	0.672	0.658	0.845	0.665
v5	0.674	0.659	0.841	0.668
v6	0.688	0.672	0.855	0.682
v7	0.694	0.652	0.858	0.672
v8	0.667	0.658	0.844	0.662
v9	0.458	0.458	0.645	0.458
v10	0.692	0.672	0.857	0.680
v11	0.699	0.696	0.853	0.698
v12	0.686	0.680	0.844	0.683
v13	0.693	0.685	0.834	0.691
v14	0.459	0.314	0.673	0.365
v15	0.694	0.688	0.846	0.692
v16	0.000	0.000	0.649	0.421
v17	0.704	0.700	0.861	0.702
v18	0.692	0.685	0.849	0.690

- v7: v3 with batch size of 128 and topology of the top layer: 256 x 0.3 x 512 x 0.3.
- v8: v3 with batch size of 128 and topology of the top layer: 1024 x 0.3 x 1024 x 0.3.
- v9: v1 trainable from $conv1\_block1\_1\_conv$.
- v10: v3 with topology of the top layer: 512 x 0.2 x 512 x 0.2.
- v11: v3 with topology of the top layer: 512 x 0.4 x 512 x 0.4.
- v12: v1 with topology of the top layer: 512 x 0.4 x 512 x 0.4.
- v13: v1 with topology of the top layer: 512 x 0.2 x 512 x 0.2.
- v14: v1 with batch size of 256.
- v15: v1 with batch size of 128.
- v16: v1 with *Adam*(learning rate 0.1).
- v17: v1 with *Adam*(learning rate $1e-5$).
- v18: v1 with *elu* activation function.

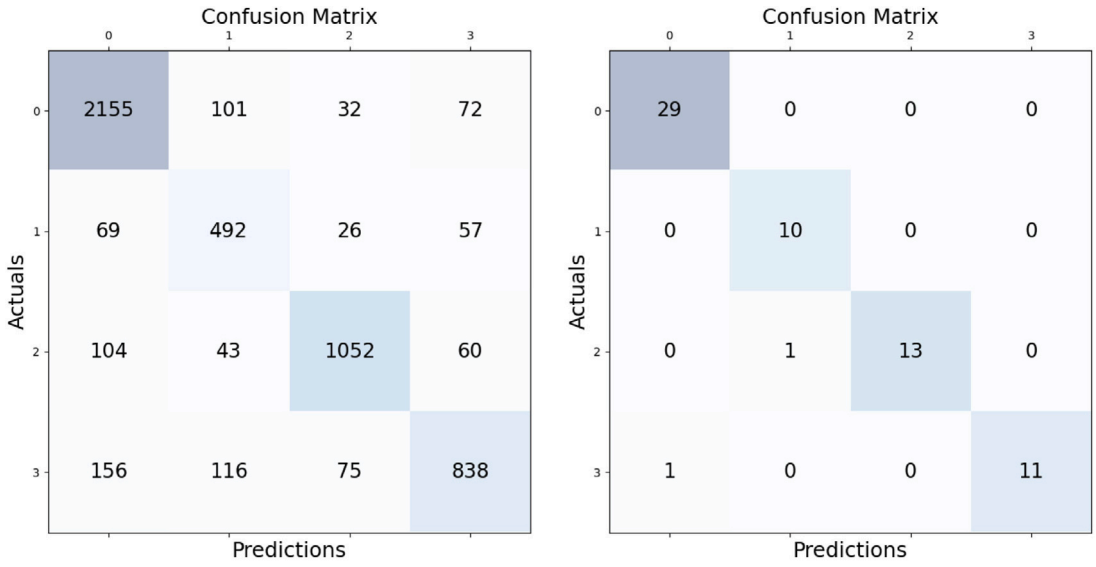
The outcomes are provided in Table 1. Among the variants, the most favorable results in terms of Area Under the Curve (AUC) were obtained for v1 and it is this variant that was chosen for the model detailed in Section 2.4.1.

3. Results

Upon conducting experiments, the performance of the classifier was evaluated using various measures on the testing set, including precision, recall and F1-score for different classes, as well as overall performance. The results are summarized in Table 2. The testing set consisted of a total of 5448 subimages; however, it is important to note that the classes were unbalanced. Specifically, class 1 was represented by 644 images, while class 0 contained 2360 images. As a consequence, the classifier exhibited lower performance in distinguishing class 1, with a precision of 62% compared to 86% for class 0, which is overrepresented in the dataset.

To mitigate the negative impact of class imbalance, we initially attempted to address the issue by undersampling the images. This involved randomly selecting 644 subimages from each class, resulting in a balanced dataset of exactly 644 images per class that yielded poor performance by the classifier, prompting us to retain the original dataset.

The overall precision (that indicates the accuracy of correctly predicted positive instances across all classes) achieved by the classifier was 82%, which is considered promising given the challenges inherent in the experiment. It includes imperfect subimage retrieval, which may include objects not representing actual microorganism fragments. Additionally, the input microscopic images were captured at different magnification levels and contained impurities such as reflections



(a) The whole testing set applied. (b) The majority voting rule incorporated.

Fig. 7. Confusion matrices of the classifier performance results on the whole testing set and upon incorporating majority voting rule on the subimages to merge results obtaining the final classification to the given class.

Table 2
The classifier performance results on applied directly the testing test.

	Precision	Recall	F1-score	Support
0	0.86	0.91	0.88	2360
1	0.62	0.78	0.69	644
2	0.89	0.83	0.86	1259
3	0.81	0.68	0.74	1185
Micro avg	0.82	0.82	0.82	5448
Macro avg	0.80	0.80	0.79	5448
Weighted avg	0.83	0.82	0.82	5448
Samples avg	0.82	0.82	0.82	5448

Table 3
The classifier performance results upon incorporating majority voting rule on the subimages to merge results obtaining the final classification to the given class.

	Precision	Recall	F1-score	Support
0	0.97	1.00	0.98	29
1	0.91	1.00	0.95	10
2	1.00	0.93	0.96	14
3	1.00	0.92	0.96	12
Micro avg	0.97	0.97	0.97	65
Macro avg	0.97	0.96	0.96	65
Weighted avg	0.97	0.97	0.97	65
Samples avg	0.97	0.97	0.97	65

from the coverslip. Despite these impediments, the obtained result is promising.

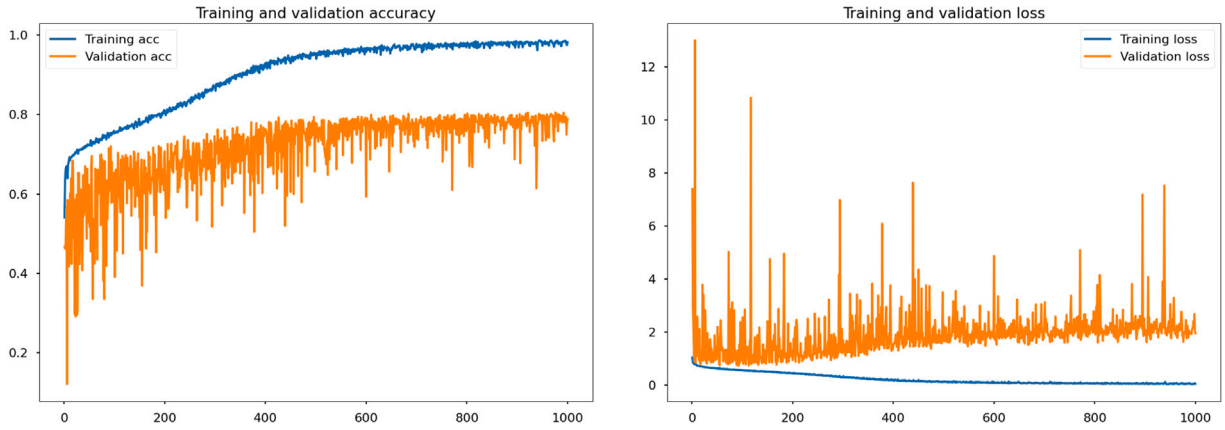
Further insight into the classifier's performance can be gained from the confusion matrix, as shown in Fig. 7. The matrix reveals the classification outcomes for each class, reaffirming the subpar performance observed for class 1.

The incorporation of the majority voting scheme, as detailed in the previous section, has yielded significant improvements in the performance of the classifier. The results on the testing set are depicted in Table 3. Notably, the primary issue observed in the previous model, namely the poor performance of class 1 due to its underrepresentation, has been effectively addressed. The precision for class 1 has now reached 91%, while the average precision and F1-score have significantly improved to 97%, indicating superior performance. These advancements are also evident in the confusion matrix shown in Fig. 7.

The utilization of the majority voting scheme on the retrieved subimages has greatly enhanced the classifier's ability to accurately differentiate input images. This improvement stems from the enhanced generalizability of the classifier, where minor misclassifications in individual subimages do not significantly impact the overall image classification. The classifier exhibits robustness in handling faulty classifications, which aligns with the true objective of our task: accurately identifying the genera of microorganisms present in the image. Given that monocultures are observable in the images, the classification of subimages becomes less relevant, with the focus shifting to assigning the overall image to a specific genera. It should be noted that applying the same scheme directly to polycultures of microorganisms in the sample may require additional methods, such as utilizing graphs with probabilities assigned to the edges.

One aspect of concern in the results is the limited size of the testing set. The preparation of samples for our experiment is a non-trivial and laborious task, involving a purification process, cultivation of samples under specific conditions for several days and image preparation. Consequently, we acknowledge the need to expand the dataset in future work and to incorporate a wider range of microorganism genera. The evaluation results involves assessing the accuracy and loss function values throughout the training process, ensuring that overfitting does not occur. To alleviate the risk of overfitting, we have incorporated dropout in the dense layers of our CNN. As illustrated in Fig. 8, there is no evidence of overfitting. Both the accuracy on the training set and the validation set steadily increase over the course of the epochs, particularly beyond 500 epochs. Additionally, no significant decrease in training accuracy is observed as the training progresses, indicating that our model successfully avoids overfitting.

A significant fact is that computations were also conducted without employing the Transfer Learning. However, the results obtained without Transfer Learning exhibited similarities to those achieved with the latter, even though there was a notably extended computation time. This can be attributed to the requirement of training all blocks of ResNet50 for each epoch, as well as starting with random weights rather than utilizing weights pre-trained on ImageNet. The generated outcomes underscore the superiority of the Transfer Learning.



(a) Accuracy. (b) Loss.

Fig. 8. Accuracy and loss function during training of the examined model on training and validation sets.

4. Conclusions

In conclusion, this study proposes the use of machine vision and machine learning techniques, specifically CNN for the automated identification of different fungi and *Chromista* based on their microscopic images and morphological traits. The system aims to address the limitations of traditional identification methods, such as morphological analysis, which are often restricted to genus-level identification, by automation the process, making it faster and more efficient.

The developed system was evaluated using a dataset of soil microorganisms obtained from the *Symbio-Bank* at The National Institute of Horticultural Research.

The dataset (Struniawski, 2023) comprised images of four different genera: *Fusarium*, *Trichoderma*, *Verticillium* and *Phytophthora*. Despite challenges related to imperfect subimage retrieval, variations in magnification levels and impurities in the images, the classifier achieved a promising overall precision of 82%.

Furthermore, majority voting scheme was incorporated, leading to significant improvements in classifier performance. The precision for the underrepresented class 1, increased to 91% from 62%, while the overall average precision and F1-score improved to 97%. These enhancements demonstrate the system's ability to accurately differentiate between microorganism genera, aligning with the primary objective of identifying the overall image's genera.

The evaluation of the model's accuracy and loss function values throughout the training process revealed no evidence of overfitting. The accuracy on the training and validation sets steadily increased over the epochs, with no significant decrease in training accuracy observed. This indicates that the dropout mechanism implemented in the dense layers of the CNN effectively prevented overfitting.

For future extension of this work, it is important to expand the dataset, to incorporate additional microorganism genera and to explore the integration of phenotypic and molecular methods to achieve species-level identification. The system has the potential to revolutionize this area in the agricultural industry by providing cost-effective and efficient pathogen detection, thereby minimizing the risk of crop losses and improving overall agricultural productivity.

In summary, the combination of machine vision, machine learning techniques and CNNs presented in this study represents a significant step towards developing an automated system for accurate identification of soil microorganisms. With further advancements and integration of complementary methods, the system can serve as a valuable and feasible tool for farmers, laboratories and researchers in the agricultural domain.

CRediT authorship contribution statement

Karol Struniawski: Prepared and programmed the classification tool, Wrote the main manuscript text, Work in the algorithm conceptual phase. **Ryszard Kozera:** Wrote the main manuscript text, Work in the algorithm conceptual phase. **Pawel Trzcinski:** Microscopic images. **Anna Lisek:** Microscopic images. **Lidia Sas Paszt:** Experimental concept.

Declaration of competing interest

The author(s) declare no competing interests.

Data availability

Exemplary data is shared on the Zenodo repository cited in the proposed paper, the full dataset is available to be shared on reasonable request.

Acknowledgments

This research was supported by The National Centre for Research and Development, Poland within the framework of the project BIOSTRATEG, grant number BIOSTRATEG3/344433/16/NCBR/2018.

References

Acuña, C., Vinnett, L., Kuan, S., 2016. Improving image analysis of online bubble size measurements with enhanced algorithms. In: IMPC Proc.. pp. 26–28.

Alshamrani, K., Alshamrani, H.A., Alqahtani, F.F., Almutairi, B.S., 2023. Enhancement of mammographic images using histogram-based techniques for their classification using CNN. Sensors 23 (1), http://dx.doi.org/10.3390/s23010235.

Arrabelly, S.B.R., Juliet, S., 2019. Transfer Learning with ResNet-50 for Malaria cell-image classification. In: ICCSP Proc.. pp. 0945–0949. http://dx.doi.org/10.1109/ICCSP.2019.8697909.

Bailey, M., Gomez, C., Finch, J., 2005. Development and application of an image analysis method for wide bubble size distributions. Miner. Eng. 18 (12), 1214–1221. http://dx.doi.org/10.1016/j.mineng.2005.07.019.

Bozinovski, S., Fulgosi, A., 1976. The influence of pattern similarity and transfer learning upon the training of a base perceptron B2. In: Proceedings of Symposium Informatica. pp. 3–121–5.

Cakir, S., Kahraman, D., Cetin-Atalay, R., Cetin, A., 2018. Contrast enhancement of microscopy images using image phase information. IEEE Access PP, http://dx.doi.org/10.1109/ACCESS.2018.2796646.

Dave, I.R., Upla, K.P., 2017. Computer aided diagnosis of Malaria disease for thin and thick blood smear microscopic images. In: SPIN. pp. 561–565. http://dx.doi.org/10.1109/SPIN.2017.8050013.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. In: CVPR Proc.. pp. 248–255.

Fan, H., Zhang, Y., Li, J., Jiang, J., Waheed, A., Wang, S., Rasheed, S.M., Zhang, L., Zhang, R., 2023. Effects of organic fertilizer supply on soil properties, tomato yield, and fruit quality: A global meta-analysis. Sustainability 15 (3), http://dx.doi.org/10.3390/su15032556.

Fiorio, C., Gustedt, J., 1996. Two linear time Union-Find strategies for image processing. TCS 154 (2), 165–181. http://dx.doi.org/10.1016/0304-3975(94)00262-2.

Gao, Z., Wang, L., Zhou, L., Zhang, J., 2017. HEp-2 cell image classification with Deep Convolutional Neural Networks. IEEE J. Biomed. Health Inform. 21 (2), 416–428. http://dx.doi.org/10.1109/JBHI.2016.2526603.

Gonzalez, R.C., Woods, R.E., 2018. Digital Image Processing, fourth ed. Pearson.

Han, J., Dong, Y., Zhang, M., 2021. Chemical fertilizer reduction with organic fertilizer effectively improve soil fertility and microbial community from newly cultivated land in the Loess Plateau of China. Appl. Soil Ecol. 165, 103966. http://dx.doi.org/10.1016/j.apsoil.2021.103966.

Hanbal, I.F., Ingosan, J.S., Oyam, N.A.A., Hu, Y., 2020. Classifying wastes using Random Forests, Gaussian Naïve Bayes, support vector machine and multilayer perceptron. IOP Conf. Ser.: Mater. Sci. Eng. 803 (1), 012017. http://dx.doi.org/10.1088/1757-899X/803/1/012017.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: ICCV Proc.. pp. 1026–1034. http://dx.doi.org/10.1109/ICCV.2015.123.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: CVPR Proc.. http://dx.doi.org/10.48550/arXiv.1512.03385.

Hossain, M.B., Iqbal, S.H.S., Islam, M.M., Akhtar, M.N., Sarker, I.H., 2022. Transfer learning with fine-tuned deep CNN ResNet50 model for classifying COVID-19 from chest X-ray images. IMU 30, 100916. http://dx.doi.org/10.1016/j.imu.2022.100916.

Huerta, E.A., Khan, A., Davis, E., Bushell, C., Gropp, W.D., Katz, D.S., Kindratenko, V., Koric, S., Kramer, W.T.C., McGinty, B., McHenry, K., Saxton, A., 2020. Convergence of artificial intelligence and high performance computing on NSF-supported cyberinfrastructure. J. Big Data 7 (1), http://dx.doi.org/10.1186/s40537-020-00361-2.

Ilonen, J., Juránek, R., Eerola, T., Lensu, L., Dubská, M., Zemčík, P., Kälviäinen, H., 2018. Comparison of bubble detectors and size distribution estimators. Pattern Recognit. Lett. 101, 60–66. http://dx.doi.org/10.1016/j.patrec.2017.11.014.

Khan, M.B., Nisar, H., Ng, C.A., Lo, P.K., Yap, V.V., 2015. Local adaptive approach toward segmentation of microscopic images of activated sludge flocs. J. Electron. Imaging 24 (6), 061102. http://dx.doi.org/10.1117/1.JEI.24.6.061102.

Kingma, D., Ba, J., 2014. Adam: A method for stochastic optimization. In: ICLR Proc.. http://dx.doi.org/10.48550/arXiv.1412.6980.

Konopka, A., Struniawski, K., Kozera, R., Trzciński, P., Sas-Paszt, L., Lisek, A., Górnik, K., Derkowska, E., Głuszek, S., Sumorok, B., Frąc, M., 2022. Classification of soil bacteria based on machine learning and image processing. In: ICCS Proc.. pp. 263–277. http://dx.doi.org/10.1007/978-3-031-08757-8_23.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444. http://dx.doi.org/10.1038/nature14539.

LeCun, Y., Bottou, L., Orr, G.B., Müller, K.-R., 1998. Efficient BackProp. In: Neural Networks: Tricks of the Trade. pp. 9–50. http://dx.doi.org/10.1007/3-540-49430-8_2.

Lie, W.-N., 1995. Automatic target segmentation by locally adaptive image thresholding. IEEE Trans. Image Process. 4 (7), 1036–1041. http://dx.doi.org/10.1109/83.392347.

Ma, L.-J., Geiser, D.M., Proctor, R.H., Rooney, A.P., O'Donnell, K., Trail, F., Gardiner, D.M., Manners, J.M., Kazan, K., 2013. Fusarium pathogenomics. Annu. Rev. Microbiol. 67 (1), 399–416. http://dx.doi.org/10.1146/annurev-micro-092412-155650.

Mohamad, N., Jusoh, N., Zaw, Z., Win, S., 2014. Bacteria identification from microscopic morphology: A survey. IJSCAI 3, 1–12. http://dx.doi.org/10.5121/ijscai.2014.3201.

Ong, J.D.L., Abigan, E.G.T., Cajucom, L.G., Abu, P.A.R., Estuar, M.R.J.E., 2020. Ensemble convolutional neural networks for the detection of microscopic fusarium oxysporum. In: ISVC Proc. Part I. pp. 321–332. http://dx.doi.org/10.1007/978-3-030-64556-4_25.

Otsu, N., 1979. A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. 9 (1), 62–66.

Párraga, C.A., Brelstaff, G., Troscianko, T., Moorehead, I.R., 1998. Color and luminance information in natural scenes. J. Opt. Soc. Amer. A 15 (3), 563–569. http://dx.doi.org/10.1364/JOSAA.15.000563.

Pham, D., Xu, C., Prince, J., 2000. A survey of current methods in medical image segmentation. Annu. Rev. Biomed. Eng. 2, 315–337. http://dx.doi.org/10.1146/annurev.bioeng.2.1.315.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: ICLR Proc.. pp. 1–14. http://dx.doi.org/10.48550/arXiv.1409.1556.

Smith, S.E., Read, D.J., 2010. Mycorrhizal Symbiosis. Academic Press.

Soille, P., 1999. Morphological Image Analysis: Principles and Applications. Springer-Verlag, pp. 173–174. http://dx.doi.org/10.1007/978-3-662-03939-7.

Struniawski, K., 2023. Sample Microscopic Images of Soil Microorganism. Zenodo, http://dx.doi.org/10.5281/zenodo.7965200.

Struniawski, K., Konopka, A., Kozera, R., 2022. Identification of soil bacteria with machine learning and image processing techniques applying single cells' region isolation. In: ESM 2022. pp. 76–81.

Szeliski, R., 2011. Computer Vision: Algorithms and Applications. Springer Science & Business Media.

Tjoa, E.A., Yowan Nugraha Suparta, I.P., Magdalena, R., Kumalasari CP, N., 2022. The use of CLAHE for improving an accuracy of CNN architecture for detecting pneumonia. SHS Web Conf. 139, 03026. http://dx.doi.org/10.1051/shsconf/202213903026.

U.S. Environmental Protection Agency, 2023. Climate change impacts on agriculture and food supply. https://www.epa.gov/climateimpacts/climate-change-impacts-agriculture-and-food-supply.

Vinnett, L., Cornejo, I., Yianatos, J., Acuña, C., Urriola, B., Guajardo, C., Esteban, A., 2022. The correlation between macroscopic image and object properties with bubble size in flotation. Minerals 12 (12), http://dx.doi.org/10.3390/min12121528.

Wahid, M.F., Ahmed, T., Habib, M.A., 2018. Classification of microscopic images of bacteria using deep convolutional neural network. In: ICECE. pp. 217–220. http://dx.doi.org/10.1109/ICECE.2018.8636750.

Watanabe, T., 2010. Pictorial Atlas of Soil and Seed Fungi, third ed. CRC Press.

Wijayawardene, N.N., et al., 2020. Outline of Fungi and fungus-like taxa. Mycosphere 11 (1), 1060–1456. http://dx.doi.org/10.5943/mycosphere/11/1/8.

Yamashita, R., Nishio, M., Do, R.K.G., Togashi, K., 2018. Convolutional neural networks: an overview and application in radiology. Insights into Imag. 9 (4), 611–629. http://dx.doi.org/10.1007/s13244-018-0639-9.

Zhang, L., Lu, L., Nogues, I., Summers, R.M., Liu, S., Yao, J., 2017. DeepPap: Deep convolutional networks for cervical cell classification. IEEE J. Biomed. Health Inform. 21 (6), 1633–1643. http://dx.doi.org/10.1109/JBHI.2017.2705583.

3.4.3 Extreme Learning Machine for identifying soil-dwelling microorganisms cultivated on agar media

Publication: **K. Struniawski**, R. Kozera, P. Trzciński, A. Marasek-Ciołakowska and L. Sas-Paszt. *"Extreme learning machine for identifying soil-dwelling microorganisms cultivated on agar media"*. Scientific Reports 14, no. 31034, 2024. doi: 10.1038/s41598-024-82174-4.

Abstract: The aim of this research is to create an automated system for identifying soil microorganisms at the genera level based on raw microscopic images of monocultural colonies grown in laboratory environment. The examined genera are: *Fusarium*, *Trichoderma*, *Verticillium*, *Purpureolicillium* and *Phytophthora*. The proposed pipeline deals with unprocessed microscopic images, avoiding additional sample marking or coloration. The methodology includes several stages: image preprocessing, segmenting images to isolate microorganisms from the background, calculating features related to image color and texture for classification. Using an extensive dataset of 2866 images from the National Institute of Horticultural Research in Skierniewice the Extreme Learning Machine model was trained and validated. The model showcases high accuracy and computational efficiency compared to other Machine Learning state-of-the art methods e.g. CatBoost, Random Forest or Convolutional Neural Networks. Statistical techniques, including Multivariate Analysis of Variance were employed to confirm significant differences among the datasets, enhancing the model's robustness. Nevertheless, Shapley Additive Explanations values provided transparency into the model's decision-making process. This approach has the potential to improve early detection and management of soil pathogens, promoting sustainable agriculture and demonstrating machine learning's potential in environmental monitoring, microbial ecology or industrial microbiology.

scientific reports

Check for updates

OPEN

Extreme learning machine for identifying soil-dwelling microorganisms cultivated on agar media

Karol Struniawski[1]✉, Ryszard Kozera[1,2], Paweł Trzciński[3], Agnieszka Marasek-Ciołakowska[3] & Lidia Sas-Paszt[3]

The aim of this research is to create an automated system for identifying soil microorganisms at the genera level based on raw microscopic images of monocultural colonies grown in laboratory environment. The examined genera are: *Fusarium*, *Trichoderma*, *Verticillium*, *Purpureolicillium* and *Phytophthora*. The proposed pipeline deals with unprocessed microscopic images, avoiding additional sample marking or coloration. The methodology includes several stages: image preprocessing, segmenting images to isolate microorganisms from the background, calculating features related to image color and texture for classification. Using an extensive dataset of 2866 images from the National Institute of Horticultural Research in Skierniewice the Extreme Learning Machine model was trained and validated. The model showcases high accuracy and computational efficiency compared to other Machine Learning state-of-the art methods e.g. CatBoost, Random Forest or Convolutional Neural Networks. Statistical techniques, including Multivariate Analysis of Variance were employed to confirm significant differences among the datasets, enhancing the model's robustness. Nevertheless, Shapley Additive Explanations values provided transparency into the model's decision-making process. This approach has the potential to improve early detection and management of soil pathogens, promoting sustainable agriculture and demonstrating machine learning's potential in environmental monitoring, microbial ecology or industrial microbiology.

Sustained growth of yields is needed to fulfill the rise of consumption associated with increasing human population. For this purpose, scientists turn their attention towards the analysis of soil microorganisms that affect plant crop. The influence of pathogenic and other deleterious microorganisms could be very severe and may lead to enormous losses[1]. In contrast, some species of soil microorganisms can effectively control plant pathogens and stimulate plant growth increasing amount and quality of crop[2]. Understandably, the identification of microorganisms is critical in this area of research. Conventional methods are based on isolating the microorganism samples from soil, observing them under a microscope and lastly identifying examined organisms on the basis of their macro and micromorphology combined with phenotypic and molecular biological methods[3]. The goal is to alleviate the latter by automatizing the task of soil microorganisms recognition based only on morphological traits. Noticeably such identification setting is only possible for some soil microorganism species and only on genera level due to the small differences in appearance of microorganisms within a genera[4]. For this reason recognition conducted by microbiologists is a process that involves many different methods combined together that are time and cost consuming[5].

Extensive studies have explored the application of Machine Learning (ML) for detecting pathogenic fungi based on leaf images, achieving remarkable accuracy levels[6]. Convolutional Neural Network (CNN) Models such as AlexNet, GoogleNet, InceptionV3 and ResNets reach AUC scores close to 99% in detecting fungal pathogens through the images of leaves[7,8]. These studies underlie the potential of ML to accurately identify plant diseases and to classify the specific genera of microorganisms responsible for these state. Unfortunately, such approaches identify threats only after symptoms have been manifested. The resulting delay can lead to significant losses, especially with diseases like Verticillium wilt[9]. It is particularly insidious as it spreads rapidly and can persist in

[1]Institute of Information Technology, Warsaw University of Life Sciences - SGGW, ul. Nowoursynowska 159, 02-776 Warsaw, Poland. [2]School of Physics, Mathematics and Computing, The University of Western Australia, 35 Stirling Highway, Crawley, Perth, WA 6009, Australia. [3]The National Institute of Horticultural Research, ul. Pomologiczna 18, 96-100 Skierniewice, Poland. ✉email: karol_struniawski@sggw.edu.pl

the soil for extended periods, making it difficult to eradicate once it becomes established[10]. Thus, early detection is crucial in managing such diseases before they cause widespread damage and agricultural losses.

There are various time-consuming and costly methods for identifying microorganisms in soil samples, including phenotypic and molecular techniques[11]. An alternative approach is to identify the genera of microorganisms with the aid of microscopic imaging. Noteworthy, while identifying soil fungi remains a niche study area, some insights might be gained from studies focusing on infections that are hazardous to human health. Zielinski et al. explored a similar challenge, demonstrating that the use of microscopic images can accelerate the identification process[12]. Their research shows that combining CNNs with bag-of-words techniques can greatly reduce both the time and cost of fungal species identification by removing the necessity for extended biochemical tests. However, the limitation of Zielinski's work is its reliance to the relatively small dataset of 180 images. Additionally, they all are captured under consistent lighting conditions and using a single type of microscope, with expert involvement potentially introducing biases. In contrast, the method presented in this paper not only addresses the need for rapid and cost-effective identification of soil microorganisms but also establishes a robust testing framework. The proposed model is validated across multiple datasets, each prepared under various conditions. This represents also the test scenario under which dataset is generated fully automatically, with the microscope capturing images sequentially and non-overlappingly to cover the sample area.

Recent research has increasingly indicated the transformative potential of ML, particularly deep learning, in microbiology. In particular, studies by Treebupachatsakul[13] and Khasim[14] showcase the application of deep learning for recognizing and classifying microorganisms, with Khasim specifically emphasizing the efficacy of CNNs. Qu[15] and Jiang[16] provide comprehensive overviews of ML applications in microbiology, underscoring its utility in classification tasks and its potential to enhance the organization and application of microbiological knowledge. The surge in research utilizing CNN methods is notable due to their straightforward processing pipeline, which eliminates the need for labor-intensive feature crafting and often bypasses feature selection entirely. Despite this, traditional machine learning methods that involve feature engineering, such as Support Vector Machines (SVM), Random Forests (RF) and k-Nearest Neighbors (KNN), remain prevalent in microbial studies. For instance, Kotwal[17] discusses their use in bacterial classification, demonstrating their continued relevance.

Rani[18], in a comprehensive review covering 100 studies on machine learning applications in microbial recognition from 1995 to 2021, notes that only 12.1% of these papers focus on fungi. This statistic highlights the novel contribution of the current paper in addressing the identification of fungal microorganisms. An example of such focused research is the work by Liu et al.[19], which employs ML techniques to detect and count fungal microorganisms in microscopic images. Similarly, Tahir et al.[20] utilized SVM for detecting fungal spores. Their approach involved extracting image patches, preprocessing them with Gaussian filters and using handcrafted features to achieve a notable accuracy of 88%. These studies collectively emphasize the significant and growing role of ML in advancing the field of microbiology, particularly in the area of fungal identification.

The main research question explored in this study is whether a robust ML model can be developed for the accurate identification of specific soil microorganism genera across various datasets, including those generated through automated image capture. This capability is crucial for advancing systems that provide rapid and precise identification, thereby enabling timely interventions in agriculture to mitigate the devastating impacts of microorganism pathogens on crops or laboratory colonies.

The datasets used in this research is obtained from *Symbio-Bank*[21] - the collection of microorganisms collected by The National Institute of Horticultural Research in Skierniewice. A system developed here is designed to classify microscopic images of the following soil microorganisms - fungi of genera *Fusarium*, *Trichoderma*, *Verticillium*, *Purpureolicillium* and *Chromista* of genera *Phytophthora*. Only a single type of microorganisms is assumed to be present on the image upon applying rectification process to obtain monocultural isolate from the input pictures of the examined material.

Microscopic images of soil-dwelling microorganisms are very complex and difficult to handle by standard image processing methods[22]. Microorganisms exhibit a wide range of shapes and textures, including non-rigid and irregular forms, which complicates segmentation and analysis[23,24]. The respective organisms' structures are not homogeneous as in the case of bacteria[25]. Indeed, they are featured by various irregular structures like hyphae, phialides, micro and macroconidia, conidia cells, zoospores and many more. Watanabe[26] conducted a brief study providing microbiologists a resource for rapid comparison of microorganism morphologies. Taxonomic differences among types of soil microorganisms include different shapes and morphology of microorganism/spore cells, different thickness of microorganism/spore cell walls, differences in color, microorganism/spore cell structure. Consequently, the goal of constructing a classifier using machine vision methods constitutes a non-trivial task.

The research leverages five distinct datasets specifically curated for this study. The initial dataset, comprising 128 images, was meticulously prepared by an expert microbiologist and served as a basis for fine-tuning the image processing pipeline. This foundational set was expanded into a second dataset with 303 images. Subsequently, three additional datasets were created using automated image acquisition techniques, collectively contributing to a total of 2,866 images.

During the literature review, it was observed that this research employs the largest database in publications dedicated to soil microorganism identification. Typically, manual image acquisition is a labor-intensive process, leading most studies to work with datasets containing fewer than 500 images. Furthermore, these images are often captured under a single setup and methodology, which may introduce challenges in generalizing the technology to new environments. To address these limitations, the model was initially optimized using the first dataset, then trained and validated across four additional datasets.

To ensure the findings are both robust and interpretable, the study integrates Explainable AI (XAI) techniques. This approach not only evaluates the model's performance but also elucidates the rationale behind

the classification of individual instances and overall test sets, as elaborated in the Results section, which also includes standard performance metrics. A key aspect of the study is also the comparison between traditional handcrafted feature approaches and CNNs. On the other hand, study aims to advance the research towards practical application stages, where prediction time is critical, especially during large-scale tests. For this purpose, the study employs the Extreme Learning Machine (ELM), a type of Single Layer Feedforward Neural Network (SLFNN), known for its rapid training and exceptionally fast prediction times. ELMs are particularly suited for practical applications where quick predictions are essential[27].

The presented system can distinguish only the tested types of soil microorganisms based on spores obtained from cultures of microorganisms on agar media and visualized on microscopic images. Within the genera of soil microorganisms, there is variability at the level of species and strains[4,28]. At this stage of the development of the presented system, the authors focused on the genera of microorganisms. The authors realize this method will fail flawlessly due to phenotypic variability within species and genera. This paper accurately recognizes the selected soil-dwelling using the ML model that, in this stage, can be directly used in the contamination early-detection of laboratory cultures by the analyzed genera of microorganisms. Detection of microorganisms in soil directly on the soil photo is currently not possible, and up to now, any system capable of performing such a task has not been proposed; this is the scope for future development merging methods proposed in this paper of phenotypic recognition with molecular as the ultimate system. For isolates belonging to *Fusarium*, *Verticillium*, *Phythophthora*, *Trichoderma* and *Purpureocillium*, which do not grow on agar media, it will not be possible to use the presented method. The demonstrated approach is the first stage of work aimed at differentiating among themselves, which is critical for horticultural practice in the five types of soil microorganisms.

Results

In this section, the results from various evaluations are comprehensively presented, focusing on the detailed analysis of the features generated across different datasets. The primary classifier employed in this research is the ELM. To provide a comprehensive evaluation, comparisons with different ML algorithms are also presented in terms of classification metrics and time of execution.

Feature analysis

This subsection explores the distinctions and commonalities in the features across the various datasets. Each dataset comprises a set of 74 features, as detailed in the Methods section. To initiate this analysis, boxplots are utilized to visually represent the value distributions of specific features across the different datasets (see Fig. 1). This method highlights the statistical differences among datasets by selecting representative features from each category for visualization. The boxplots effectively illustrate how the statistical properties of features vary between datasets, underscoring the diversity in the characteristics of images from different strains, microscopes and lightning conditions.

Multivariate Analysis of Variance (MANOVA)[29] was employed to assess the differences between datasets across a reduced feature space (10 dimensions) derived from Principal Component Analysis (PCA)[30]. The MANOVA revealed significant findings across both the intercept and dataset comparisons. In terms of the intercept, the analysis indicated that the combined effect of all PCA components was significantly different from zero (Wilks' lambda = 0.3304, $F(10, 2818) = 571.1673$, $p < 0.0001$). This suggests a robust multivariate separation in the reduced feature space. Furthermore, comparing datasets showed highly significant differences (Wilks' lambda = 0.0119, $F(40, 10687) = 592.4963$, $p < 0.0001$), indicating distinct multivariate distributions among the datasets. Pillai's Trace (2.5217) shows a significant portion of variance in the dependent variables is explained by the grouping variable (Dataset label). Hotelling-Lawley Trace (9.6309) suggests substantial multivariate differences between the datasets. Roy's Greatest Root (5.1398) implies a strong difference in the first canonical dimension underscoring the statistically meaningful differences of feature values across the datasets. Visual confirmation through t-Distributed Stochastic Neighbor Embedding (t-SNE)[31] analysis in Fig. 2 further supported these findings, illustrating clear separation between datasets in the reduced-dimensional space highlight the different nature of the datasets employed in this research yielding the complex task for ML model.

Performance of the model

The performance of the ELM is evaluated both on individual datasets and on combined datasets (marked as All) to assess its robustness and adaptability. The ELM performance is tested on individual datasets, varying the number of neurons in the hidden layer from 100 to 2000. The activation function used is Mish, an unconventional choice for ELM but one that recent studies suggest it can be successfully applied to ELM, borrowing from its use in deep learning techniques[32].

The implementation utilized is TfELM[33], which accelerates computations by leveraging the TensorFlow library to run on a GPU. This setup enables efficient processing and enhances the ELM's computational performance. The evaluation follows a protocol of 50 iterations of 10-Fold Cross-Validation to ensure reliable and stable results for a given dataset.

The ultimate performance metrics, representing the best outcomes for the specified number of neurons, are summarized in Table 1. The ROC/AUC scores reflect the one-vs-rest approach used for multi-class classification.

The results indicate exceptional performance, as evidenced by high F1 scores, precision, recall and near-perfect ROC AUC values across all datasets, affirming the model's robustness and accuracy in classifying various genera of soil microorganisms by the accurate feature generation from various datasets.

Comparison to the other models

The performance of ELM was benchmarked against several well-established classifiers to evaluate its efficiency and accuracy. The chosen classifiers for this comparison included Multi-Layer Perceptron (MLP), Gradient

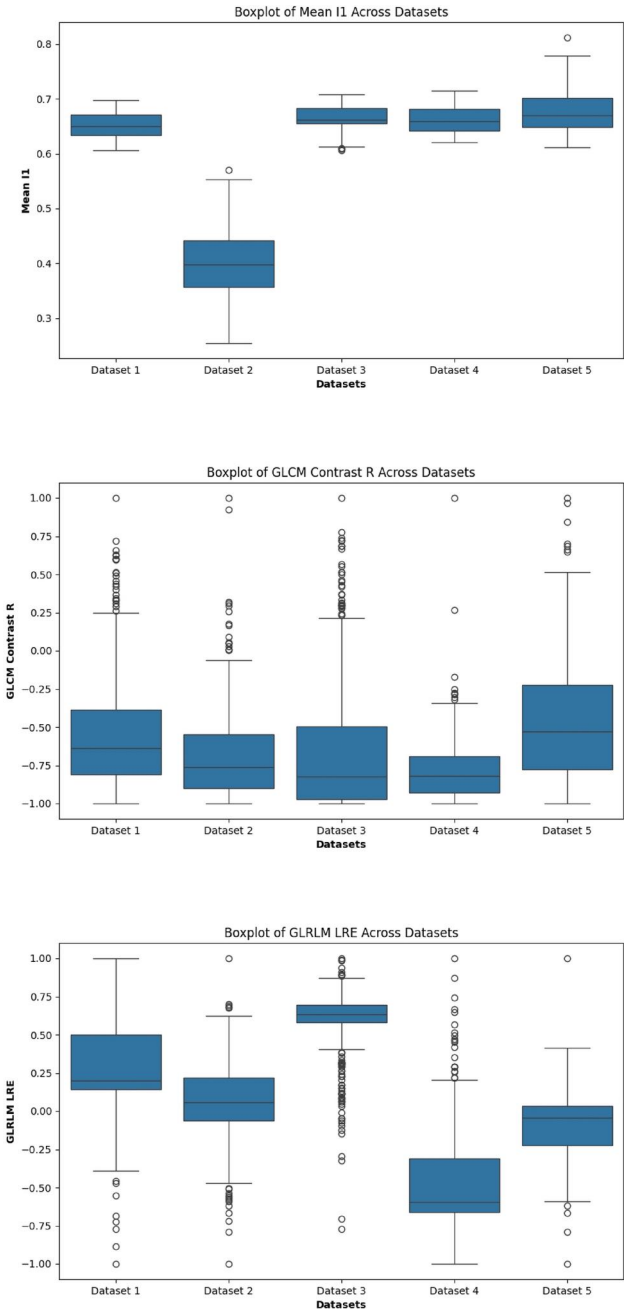


Figure 1. Boxplots of a given feature across different datasets.

Boosting, XGBoost, LightGBM, CatBoost and Random Forest[34], all implemented using the scikit-learn library. ELM demonstrated promising results, particularly comparable to CatBoost, highlighting its capability for precise predictions. Table 2 represents the comparison tests across different datasets and methods.

For reference, experiments were also conducted using Convolutional Neural Networks (CNNs) - ResNet50 and ResNet152V2, fine-tuned on pre-trained ImageNet models. Input images from each dataset were resized to $224 \times 224 \times 3$ and split into training, validation, and testing sets in a 70:20:10 ratio. The models were adapted with a custom top layer consisting of two fully connected layers with 512 neurons, separated by a Dropout layer with a rate of 0.3 to nullify 30% of the weights, enhancing robustness and mitigating overfitting. The Adam optimizer with a learning rate of 0.004 and the Sparse Categorical Crossentropy loss function were used. Each model was trained for a maximum of 1000 epochs, with Early Stopping implemented to halt training if the validation loss did not decrease further. The results are summarized in Table 3.

Time execution

The computational efficiency of various classifiers was assessed to evaluate their suitability for the system. The evaluation involved measuring both the average fit time (training duration) and the average prediction time

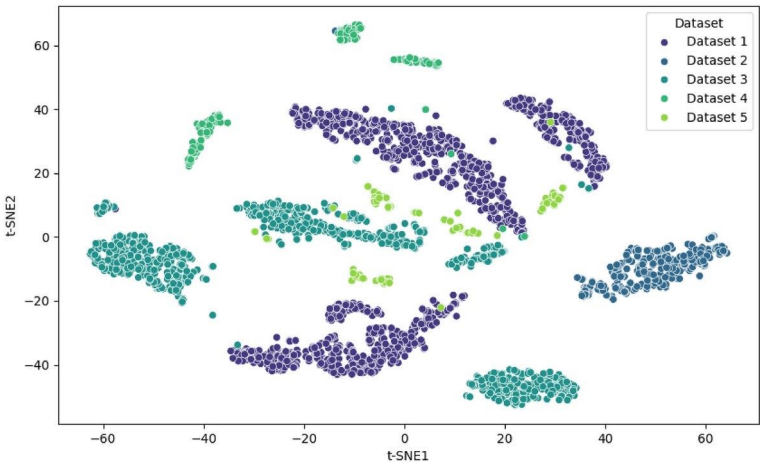


Figure 2. t-SNE visualization of datasets.

Dataset	Neurons	F1	Precision	Recall	AUC (OvR)
All	2000	0.969	0.969	0.969	0.997
`Dataset1`	2000	0.874	0.874	0.874	0.965
`Dataset2`	1000	0.887	0.887	0.887	0.975
`Dataset3`	1000	0.989	0.989	0.989	1.000
`Dataset4`	1000	0.998	0.998	0.998	1.000
`Dataset5`	300	0.984	0.984	0.984	0.996

Table 1. Metrics calculated from 50 runs of 10% cross-validation on a given datasets using Extreme Learning Machine with mish activation function.

during a single fold of 10-Fold cross-validation. For this purpose, the largest dataset was used, a combination of `Dataset1-5`, comprising 2,866 images. The results, summarized in Table 4, clearly illustrate the exceptional performance of the ELM in terms of both fit and prediction times. Compared to other classifiers, ELM stands out, particularly when contrasted with CatBoost that is proved to provide comparable accuracy performance with ELM. The ELM demonstrated a significant advantage, being six times faster in the fit phase and twice as fast in the prediction phase, underscoring its potential for practical industrial applications. Given its accuracy and speed, ELM offers substantial benefits over other classifiers. Noteworthy, CNN time of the prediction has been measured as 0.11 s, twelve times longer than ELM.

Explainable artificial intelligence

This subsection explores the transparency and interpretability of the model's outputs by employing XAI techniques. To elucidate the impact of the handcrafted features, Shapley Additive Explanations (SHAP) are utilized. SHAP values provide insights into the contribution of each feature to the model's predictions for any machine learning method[35]. Figure 3 presents a summary plot for `Dataset1`, where the ELM with 1000 neurons in the hidden layer and the Mish activation function is analyzed. This plot allows us to discern whether specific features have a positive or negative influence on the model's predictions, thereby enhancing the interpretability of the classification decisions.

The beeswarm plot provides a clear visualization of the most influential features in the overall classification process using the test set. As observed, a variety of feature types contribute significantly, indicating that these features serve as robust descriptors of image characteristics. Additionally, individual prediction explanations can be examined using the waterfall plot, which illustrates how specific traits positively or negatively influence the model's prediction for a given sample (see Fig. 4). This dual approach-examining both the overall feature importance and the detailed contribution for individual samples-enhances the understanding of the model's decision-making process. Notably, the analysis of both beeswarm and waterfall SHAP plots reveals that the selected feature sets, including statistical measures over the I1I2I3 and XYZ color spaces, as well as GLCM and GLRLM features, each contribute significantly to the final decisions made by the classifier.

Another approach involved using SHAP to evaluate the overall influence of features on the target variable. The `Dataset1` was cleaned, normalized, and split into training and testing subsets (80/20). An ELM model with 1,000 neurons and a mish activation function was trained. SHAP explainability was applied to the testing data to calculate SHAP values and assess feature importance. The absolute SHAP values were summed across all samples, allowing the ranking of features based on their contribution to the model's predictions. The most influential features were identified and are presented in Table 5.

Dataset	Model	F1	Recall	Accuracy	AUC
All	**CatBoost**	**0.982**	**0.988**	**0.983**	**1.000**
	MLPClassifier	0.980	0.990	0.990	0.999
	LightGBM	0.980	0.987	0.987	1.000
	XGBoost	0.976	0.981	0.985	0.999
	RandomForest	0.974	0.981	0.980	0.999
	GradientBoosting	0.971	0.976	0.976	0.998
	ELM	0.969	0.969	0.969	0.997
	AdaBoost	0.623	0.631	0.631	0.862
Dataset1	**ELM**	**0.874**	**0.874**	**0.874**	**0.965**
	MLPClassifier	0.856	0.861	0.860	0.951
	GradientBoosting	0.853	0.863	0.861	0.966
	XGBoost	0.853	0.856	0.854	0.968
	LightGBM	0.847	0.851	0.851	0.965
	CatBoost	0.841	0.850	0.843	0.968
	RandomForest	0.840	0.844	0.847	0.964
	AdaBoost	0.280	0.281	0.282	0.577
Dataset2	**ELM**	**0.887**	**0.887**	**0.887**	**0.975**
	MLPClassifier	0.852	0.857	0.859	0.966
	CatBoost	0.840	0.846	0.844	0.970
	LightGBM	0.839	0.847	0.843	0.968
	XGBoost	0.829	0.833	0.837	0.967
	GradientBoosting	0.818	0.824	0.828	0.959
	RandomForest	0.791	0.794	0.796	0.957
	AdaBoost	0.446	0.447	0.447	0.743
Dataset3	**RandomForest**	**0.999**	**1.005**	**1.000**	**1.000**
	CatBoost	0.998	1.000	1.006	1.000
	ELM	0.998	0.998	0.998	1.000
	LightGBM	0.997	0.998	0.998	1.000
	MLPClassifier	0.997	1.007	1.001	1.000
	XGBoost	0.995	1.001	1.003	1.000
	GradientBoosting	0.995	1.000	0.996	1.000
	AdaBoost	0.972	0.981	0.976	0.987
Dataset4	**CatBoost**	**0.990**	**0.992**	**0.993**	**1.000**
	ELM	0.989	0.989	0.989	1.000
	LightGBM	0.989	0.998	0.997	1.000
	XGBoost	0.987	0.994	0.996	1.000
	GradientBoosting	0.986	0.996	0.987	1.000
	MLPClassifier	0.985	0.994	0.987	1.000
	RandomForest	0.984	0.985	0.992	0.999
	AdaBoost	0.908	0.911	0.911	0.976
Dataset5	**ELM**	**0.984**	**0.984**	**0.984**	**0.996**
	CatBoost	0.962	0.963	0.964	0.996
	LightGBM	0.955	0.965	0.959	0.997
	XGBoost	0.955	0.964	0.957	0.994
	RandomForest	0.952	0.958	0.956	0.996
	GradientBoosting	0.948	0.957	0.957	0.994
	MLPClassifier	0.946	0.949	0.950	0.996
	AdaBoost	0.817	0.819	0.826	0.963

Table 2. Metrics calculated from 50 runs of 10% cross-validation on a given datasets using various ML models, sorted by descending F1 score per dataset, bold rows are the best results obtained for each dataset.

Discussion

The central objective of this study was to develop a robust ML pipeline capable of accurately classifying soil microorganisms based on their microscopic images. This task, particularly focusing on specific genera of fungi and *Chromista*, presents significant challenges due to the inherent complexity and variability of these organisms'

Dataset	Metric	ResNet50	ResNet152V2
All	Accuracy	0.7438	**0.9000**
	Precision	0.6546	**0.9167**
	Recall	0.5381	**0.9000**
	F1	0.5461	**0.8920**
	AUC	0.9265	**1.0000**
Dataset1	Accuracy	0.6154	**1.0000**
	Precision	0.5000	**1.0000**
	Recall	0.5333	**1.0000**
	F1	0.5079	**1.0000**
	AUC	0.8028	**1.0000**
Dataset2	Accuracy	0.6774	**0.9691**
	Precision	0.4051	**0.9678**
	Recall	0.4485	**0.9678**
	F1	0.4064	**0.9678**
	AUC	0.8915	**0.9992**
Dataset3	Accuracy	**0.9278**	0.8462
	Precision	**0.9240**	0.8750
	Recall	**0.9336**	0.8667
	F1	**0.9277**	0.8389
	AUC	**0.9733**	0.9833
Dataset4	Accuracy	**0.9219**	0.8710
	Precision	**0.9154**	0.9167
	Recall	**0.9156**	0.8263
	F1	**0.9143**	0.8336
	AUC	**0.9844**	0.9854
Dataset5	Accuracy	0.8000	**0.9437**
	Precision	0.8889	**0.9410**
	Recall	0.8000	**0.8742**
	F1	0.7619	**0.8911**
	AUC	**1.0000**	0.9922

Table 3. Metrics calculated for ResNet50 and ResNet152V2 on various datasets. Bold values represent the best result for each metric within the same dataset.

Model	Fit time [s]	Score time [s]
AdaBoost	1.197	0.022
LightGBM	3.564	0.018
ELM	3.610	0.009
XGBoost	4.439	0.018
RandomForest	5.212	0.027
MLPClassifier	5.968	0.008
CatBoost	18.283	0.017
GradientBoosting	30.021	0.007

Table 4. Time of the fit and prediction for a selected ML methods and dataset that incorporates together all images used in research.

structures. The results from the extensive experiments underscore the proposed approach effectiveness highlighting substantial advancements in automated microorganism identification.

Across all datasets, ELM consistently achieved high accuracy, precision, recall and ROC/AUC values, demonstrating its robustness and adaptability to varying imaging conditions and microorganism strains. This consistency is notable whether the model was tested on individual datasets or a combined dataset, underscoring its potential for practical application in diverse environments. When benchmarked against other well-established machine learning models, including MLP, Gradient Boosting, XGBoost, LightGBM, CatBoost and RF, ELM outperformed these methods in terms of both speed and accuracy. Its superior balance of computational efficiency and predictive power makes it particularly suitable for large-scale data processing tasks.

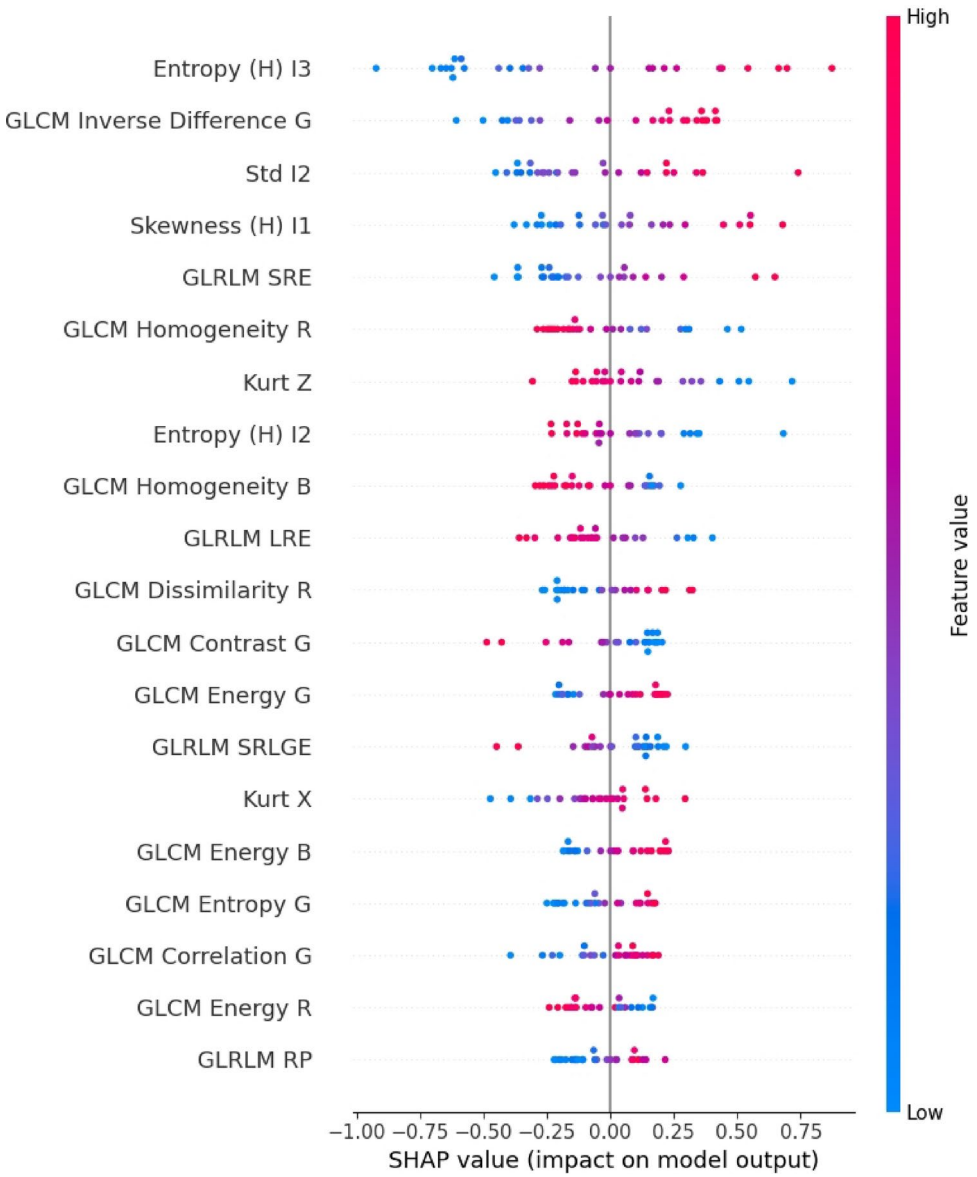


Figure 3. Shapley Additive Explanations summary plot for `Dataset1` and ELM with 1000 neurons in hidden layer and mish activation function.

The comparison of classical feature-based models and CNNs highlights the performance of ELM in terms of execution time and predictive accuracy. While the analyzed CNN models generally achieved higher F1 scores and AUC values across most datasets, their prediction time (0.11s) was significantly slower compared to ELM (0.009s). This makes ELM particularly advantageous in scenarios where quick predictions are critical. Despite its lower performance metrics in some cases, ELM demonstrated competitive results, achieving an F1 score of up to 0.984 in specific datasets, showcasing its balance between computational efficiency and predictive power.

The comprehensive analysis of the datasets using MANOVA revealed significant multivariate differences among them. This finding indicates that each dataset captures unique aspects of the microorganism images, emphasizing the importance of using a diverse set of data to train and validate the model. The variability among datasets is crucial for ensuring that the model is generalizable and robust across different imaging conditions and microorganism characteristics.

Another contribution of this study is the use of SHAP values to understand the impact of individual features on the model's predictions. This analysis provides crucial insights into how the model interprets different morphological traits and makes classification decisions. The ability to interpret these decisions is vital, especially in scientific and industrial applications where understanding the model's reasoning can enhance trust and acceptance of its outputs. The most influential features, as identified by SHAP, span across all categories of the generated features described in the Feature Calculation section. This highlights the effectiveness of the approach, with texture features, in particular, emerging as the most descriptive for the ELM model.

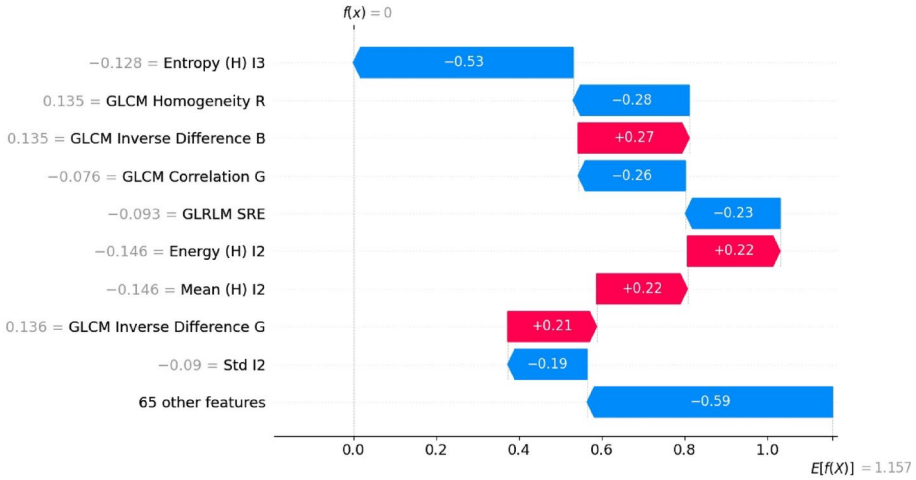


Figure 4. Shapley Additive Explanations waterfall plot for one of the samples from test set of `Dataset1` and ELM with 1000 neurons in hidden layer and mish activation function.

Feature	Importance
Entropy (H) I3	11.13
GLCM Energy B	7.90
GLCM Entropy R	7.72
GLRLM SRLGE	7.04
Skewness (H) I1	6.60
GLCM Correlation R	6.13
GLCM Energy R	5.98
GLRLM SRE	5.83
GLRLM LRE	5.69
Kurthosis X	5.58
Std I2	5.53
GLCM Dissimilarity R	4.99
GLCM Entropy G	4.61
GLCM Homogeneity G	4.58
GLCM Inverse Difference G	4.50
GLCM Homogeneity R	4.49
Std I1	4.32
Kurthosis Z	4.30
GLRLM RLN	4.26
GLCM Cluster Prominence G	4.17

Table 5. Top 20 features ranked by importance based on SHAP values calculated for ELM with 1000 neurons, mish activation function on Dataset1.

The proposed approach also incorporated datasets obtained through automated image acquisition. This automation not only expanded the dataset size significantly but also introduced greater variability in imaging conditions. The model's successful performance on these automated datasets demonstrates its potential for deployment in real-world diagnostic systems, where automated and rapid identification processes are essential. Noteworthy, in particular ELM performed better for datasets automatically acquired than the manual one taken by the expert.

Comparing research with existing literature, it becomes clear that this study sets a new benchmark in the field of soil microorganism identification. The use of a substantially larger and more varied dataset, coupled with automated image acquisition and a robust machine learning model, juxtapose this work from earlier efforts that often relied on smaller, manually curated datasets and more uniform imaging conditions. Future research could focus on scaling this approach for broader use, including integration into automated imaging systems for early detection of pathogenic microorganisms.

Additionally, while this study focused on specific genera of fungi and *Chromista*, the methodology could be expanded to include a broader range of soil microorganisms, such as bacteria, by addressing their unique

morphological characteristics. A key challenge in such an expansion would involve crafting features tailored to the new genera. Adding new genera to the analyzed dataset would require evaluating the existing features to ensure they effectively capture the relevant morphological differences. While the current feature set emphasizes color and texture, new datasets that include genera with distinct structural traits could benefit from the integration of additional geometric or shape descriptors, such as circularity, elongation, or convexity. Moreover, incorporating multi-scale texture features could improve the detection of more complex structures, complementing the existing color and texture-based metrics. If new genera are imaged using alternative methods, such as fluorescence microscopy, the system could be adapted by recalibrating color space features or incorporating intensity-specific metrics tailored to these imaging conditions.

Combining the machine vision-based approach with traditional microbiological methods could yield a hybrid system that leverages the strengths of both. For instance, rapid initial screening using proposed model could be followed by targeted molecular or phenotypic tests for confirmatory diagnosis, thus enhancing both speed and accuracy. Another potential application is the quick detection of cross-contamination during the cultivation process. This is particularly critical when working with soil-dwelling microorganisms, where multiple genera coexist and interact, potentially affecting experimental outcomes and product quality. Contamination detection is especially challenging with certain genera, such as *Trichoderma*, *Fusarium*, *Verticillium*, and *Purpureocillium*. A hybrid system combining rapid machine vision analysis with confirmatory microbiological methods could be invaluable for detecting and mitigating such contamination, ensuring the integrity of experimental cultures.

To ensure the model's adaptability to new and diverse environmental conditions, future studies should explore its performance on datasets obtained from different geographic regions, soil types and agricultural practices. This would further test the model's robustness and ensure its applicability in a wide range of real-world scenarios.

In conclusion, this study successfully addresses the complex task of classifying soil microorganisms based on microscopic images. By developing and validating a robust ML model and integrating diverse datasets including those obtained through automated image acquisition, the advancement in the field is significant. The insights gained from feature analysis and the application of XAI techniques underscore the model's utility and transparency. This work not only advances automated microorganism identification but also lays the groundwork for future innovations in microbial diagnostics and soil health monitoring. The proposed approach holds great promise for transforming agricultural practices through early detection and management of soil-borne pathogens, contributing significantly to sustainable crop production and food security.

Methods

In this study, a comprehensive pipeline is proposed, integrating image preprocessing, segmentation, feature generation and classification using extracted features. Each element of this pipeline is meticulously elaborated upon in this section. Additionally, the five distinct datasets employed in this research are thoroughly explained, along with the detailed processes of image preprocessing, segmentation, feature extraction and the specifics of the ELM method.

Datasets

Microorganisms are grown in Potato Dextrose agar, specifically Merck 1.10130.0500 at 26 degrees Celsius. In the case of *Fusarium*, *Verticillium* and *Phytophthora* the microorganisms are incubated for 7–10 days, while for *Trichoderma* strain it takes about 4-5 days until conidial spores are produced. In total five different datasets were formed that can be obtained from the corresponding author upon the reasonable request.

`Dataset1`

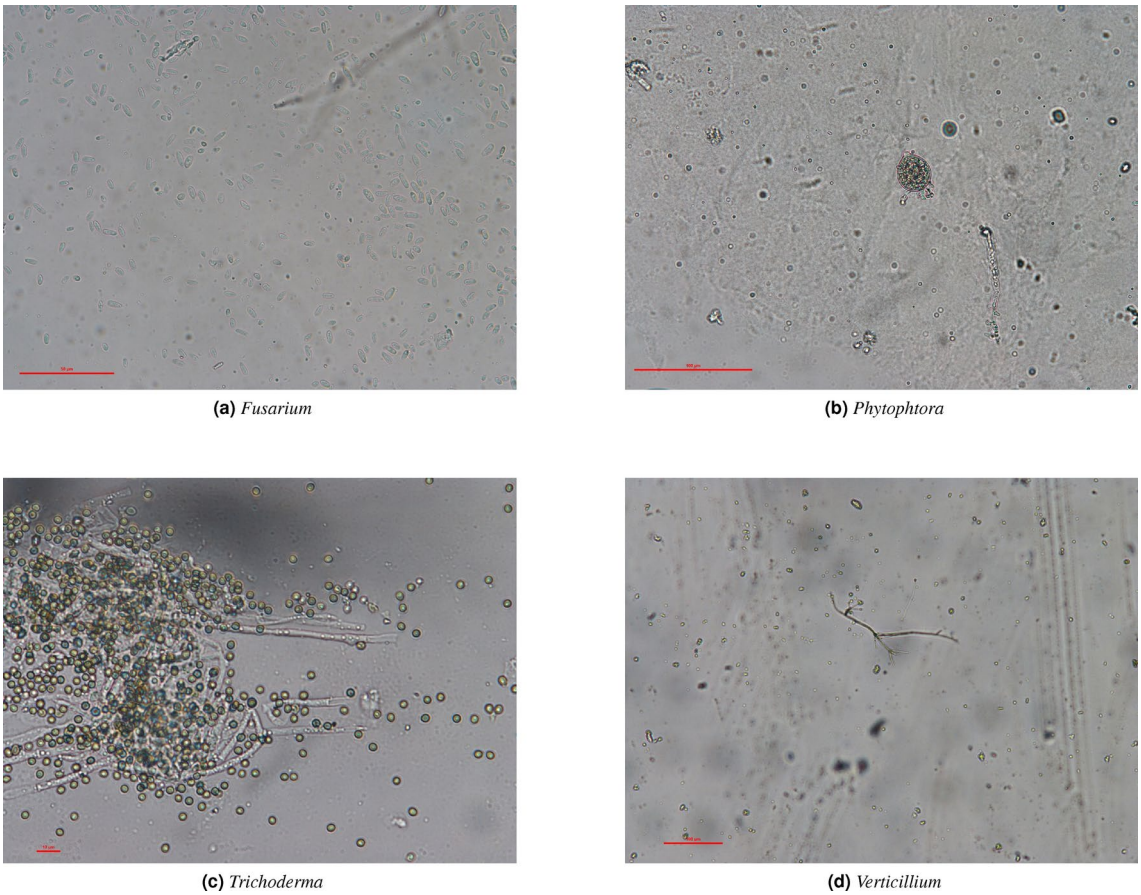
The initial experiments utilized a dataset comprising 128 images of microorganisms from the genera *Verticillium*, *Trichoderma*, *Phytophthora* and *Fusarium*, referred to as `Dataset1` throughout this manuscript with exemplary images presented in Fig. 5. This foundational dataset is based on the work of Kruk et al.[36], which demonstrated the potential for classifying soil-dwelling microorganisms using ML techniques.

The primary objective with `Dataset1` was to develop a robust image preprocessing and segmentation pipeline that could accurately differentiate the microorganism regions from the background as this is recognized as a crucial step for ultimate model's performance. Despite being well-suited for preliminary investigations into image segmentation and feature extraction, `Dataset1` was relatively small, which limited its ability to ensure the robustness of the solution across diverse scenarios.

To overcome these limitations and to provide a comprehensive evaluation of the methodology's performance, the additional, substantially larger datasets are provided. These new datasets were crucial in validating the generalizability and effectiveness of the proposed image preprocessing, masking and classification methods at scale.

`Dataset1` was carefully curated by professional microbiologists who were responsible for both sample preparation and image acquisition through microscopy. The primary goal was to generate high-quality images that clearly showcased the morphological characteristics essential for accurate microorganism identification. To achieve this, each image was meticulously focused to ensure that the unique features of each genus were prominently visible. The process began with the cultivation of the microorganisms, followed by the careful transfer biological material onto microscope slides. To facilitate optimal imaging, a drop of distilled water was added to dilute the biological material and a cover slip was placed on top (see Fig. 8). These prepared slides were then subjected to microscopic examination, allowing the microbiologists to capture detailed images at different magnification levels that would serve as the basis for the ML analysis.

Given that the model in question solely relies on image data for input without the incorporation of additional multimodal features, the quality and clarity of these images were paramount. However, this approach



(a) *Fusarium* **(b)** *Phytophtora* **(c)** *Trichoderma* **(d)** *Verticillium*

Figure 5. Explanatory images from `Dataset1`.

introduces a degree of bias, as image acquisition process rely heavily on the expertise of the microbiologists. This necessitates the involvement of experts throughout the pipeline to ensure that the images accurately represent the microorganisms' traits for effective model training and evaluation.

`Dataset2`

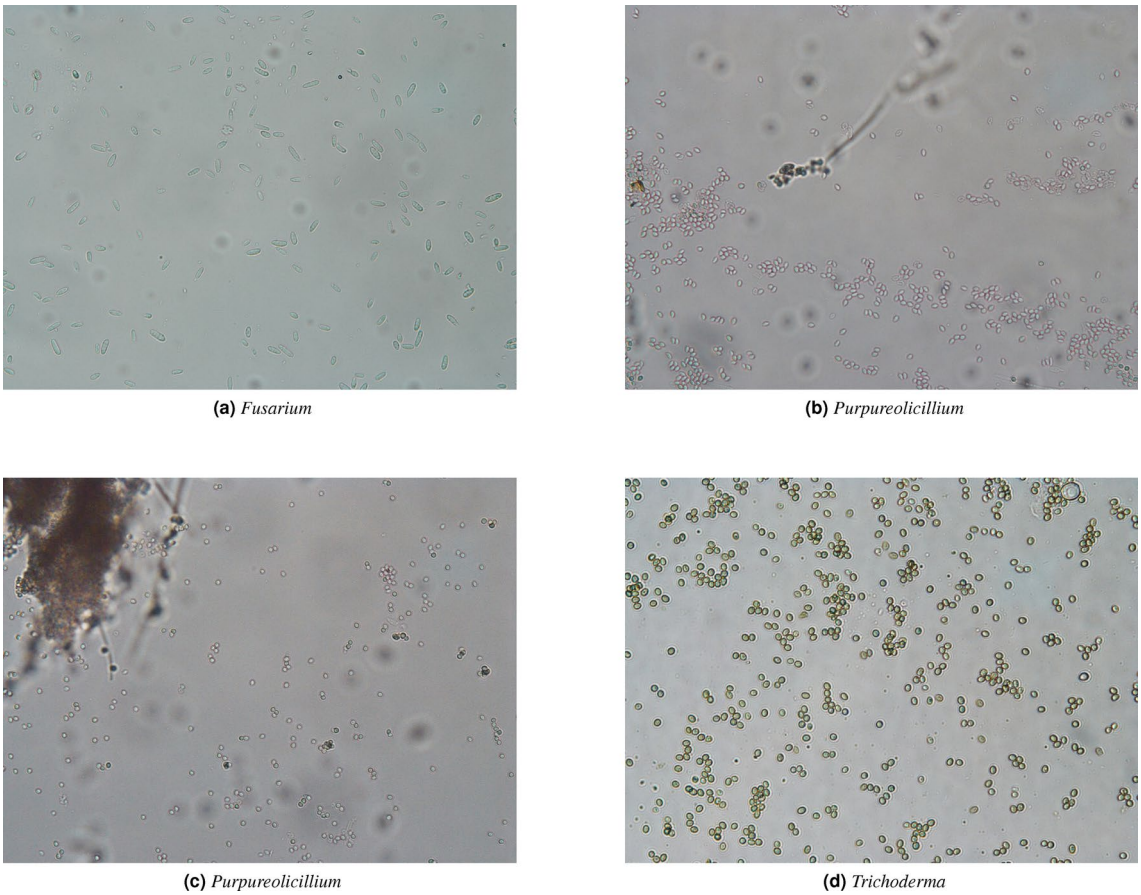
`Dataset2` builds upon the foundation of `Dataset1` by expanding its scope and content. This expanded dataset includes additional images of the original genera (*Trichoderma* and *Fusarium*) and introduces images of a new genus, *Purpureocillium* (see Fig. 6). Prepared using the same procedures as `Dataset1`, `Dataset2` effectively consolidates and enhances the original set, resulting in a comprehensive collection of 304 images.

`Dataset3` and `Dataset4`

Upon evaluating `Dataset1` the accuracy of the generated segmentation masks is assessed for the suitability of the calculated features and the overall performance of the applied ML methods. This comprehensive analysis highlighted the need for a larger, more automated dataset to minimize bias and enhance scalability. Thus, the third dataset, referred to in this paper as `Dataset3` was prepared, aiming to automate the entire image acquisition process and build a significantly larger dataset without the direct involvement of experts.

The critical point in creating `Dataset3` was the elimination of expert bias inherent in manually capturing images, which, while valuable for understanding morphological traits, does not reflect real-world scenarios where a fully automated system is the target. The approach involved using an automated microscope equipped with a movable mechanical stage controlled by built-in servo motors and software drivers. This setup allowed the microscope to systematically capture images across a predefined area without overlap.

During initial trials, the challenges with out-of-focus images due to the varying depths (Z-axis) of microorganisms at high magnification levels and slight imperfections on the slide surfaces were encountered. Although, the autofocus feature helped address this issue, it occasionally failed to achieve perfect focus, leading to some variability in image quality. Despite these challenges, the automated system substantially reduced the reliance on expert involvement and increased the efficiency of image acquisition. To ensure comparability, the samples for `Dataset3` were cultivated similarly to those in `Dataset1`. Two methods of material transfer onto microscope slides were examined. The first method, consistent with `Dataset1`, involved placing the biological material on a slide, adding a drop of water and covering it with a slip. This approach created `Dataset3`. The second method involved smearing the material directly onto the slide without a cover slip, which formed a new dataset, referred to as `Dataset4` (see Fig. 9b).



(a) *Fusarium* **(b)** *Purpureolicillium* **(c)** *Purpureolicillium* **(d)** *Trichoderma*

Figure 6. Explanatory images from `Dataset2` (here represented only new that are not present in `Dataset1`).

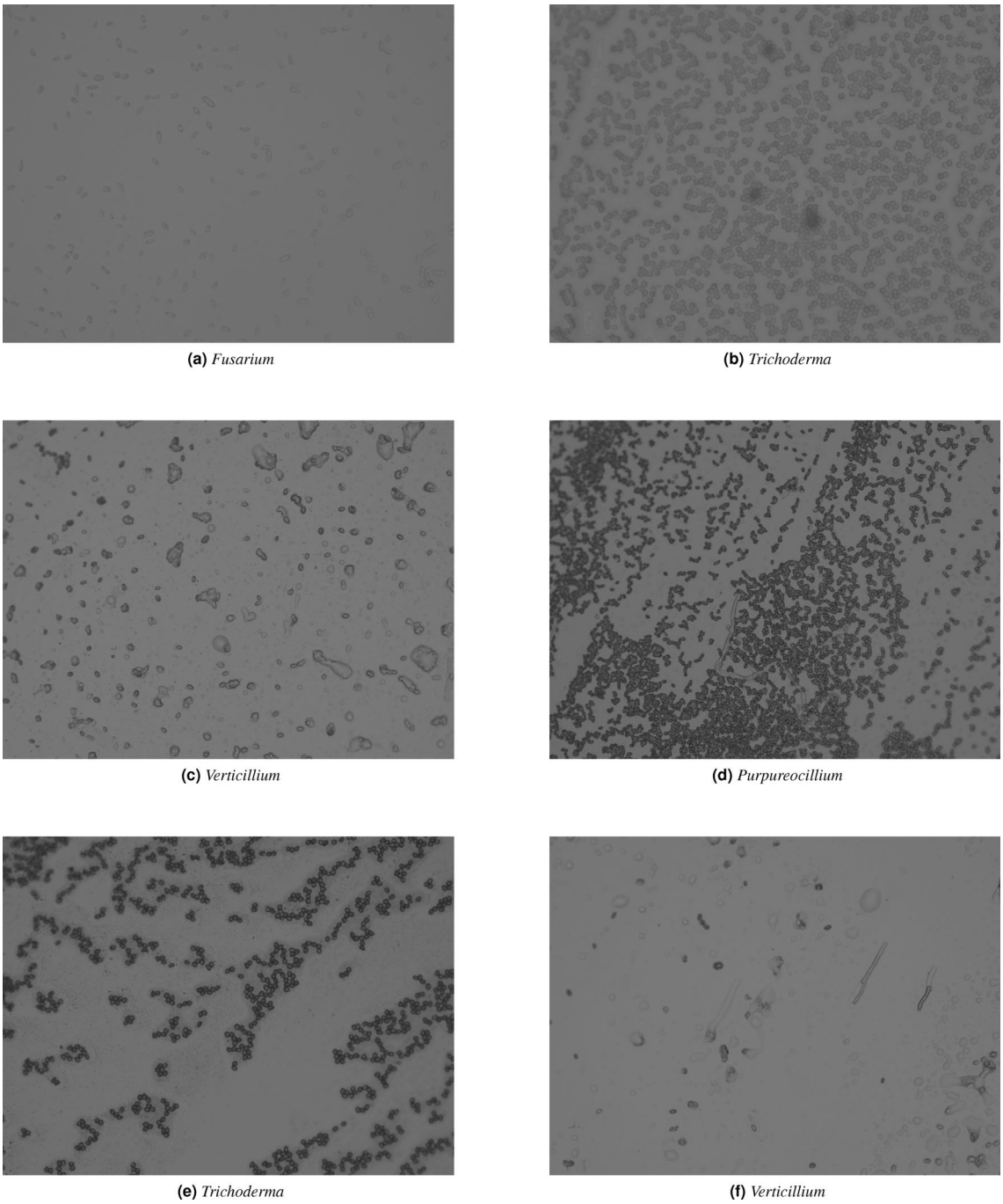
Both `Dataset3` and `Dataset4` were collected without requiring biological expertise for each image. Instead, a trained man handled the sample preparation and setup of the microscope, which included positioning the sample on the mechanical stage, using software to target areas of interest and initiating the automated imaging process with autofocus. This approach simulates a real-world scenario where minimal expert intervention is needed. Sample images from both datasets are presented in Fig. 7. Looking forward, there is potential for further automation, such as using robots for material transfer and advanced software for precise targeting, which would enhance the efficiency and reliability of the image acquisition process. These advancements are crucial for the ongoing development and scalability of the system.

`Dataset5`

The fifth dataset, referred to as `Dataset5`, was generated following a procedure similar to that of `Dataset4` (Fig. 8). For `Dataset4`, images were captured in a non-overlapping, systematic manner to cover the entire sample area, as illustrated in Fig. 9b. Each image was taken using the microscope's autofocus feature, which aimed to optimize focus for each shot. However, while autofocus provided a reasonable solution, it occasionally failed to achieve perfect focus across all images.

To address the limitations of autofocus, `Dataset5` incorporated a depth imaging technique. This method involved capturing a series of images at varying focal distances (the distance between the sample and the camera lens) for each section of the sample. These multiple images, each focused at a different depth, were then automatically stacked to produce a single composite image. This stacking process significantly enhances the overall image quality by ensuring that all areas of the sample are in focus, thus mitigating the problem of focal inconsistencies observed with single-focus images.

Although this depth imaging approach enhances image clarity significantly, it requires a slower data acquisition process due to the need to capture and stack multiple images at different focal depths for each area of the sample. For `Dataset5`, the additional challenges were encountered when attempting to use the depth imaging technique in conjunction with the traditional method of adding a drop of water between the microscope slide and the coverslip that was applied for `Dataset3`. This method, commonly used to stabilize the sample for conventional imaging, proved unsuitable for depth imaging. The water layer caused instability, leading to slight movements of fungal elements within the sample. Consequently, these movements resulted in blurred or misaligned images when stacking the focal series, thereby undermining the intended clarity of the depth imaging process.



(a) *Fusarium* **(b)** *Trichoderma* **(c)** *Verticillium* **(d)** *Purpureocillium* **(e)** *Trichoderma* **(f)** *Verticillium*

Figure 7. Explanatory images from `Dataset3` (**a**–**c**) and `Dataset4` (**d**–**f**).

Overview of the datasets

Initially, `Dataset1` was prepared for preliminary research, focusing on data segmentation and feature calculation. This dataset included 128 images from four genera: *Verticillium*, *Trichoderma*, *Phytophthora* and *Fusarium*.

To test and to enhance the robustness of the solution, this dataset was expanded by creating `Dataset2`. This expanded dataset added more images for *Trichoderma* and *Fusarium* and incorporated a new genus, *Purpureocillium*, increasing the total to 303 images.

Subsequently, with the implementation of an automated image acquisition system, three additional datasets were developed: `Dataset3`, `Dataset4` and `Dataset5`. These datasets were captured using an automated microscope, allowing to gather a significantly larger and more diverse collection of images. Unfortunately, *Phytophthora* could not be included in the automated datasets due to deterioration of all samples held prior to the establishment of the automated imaging system.

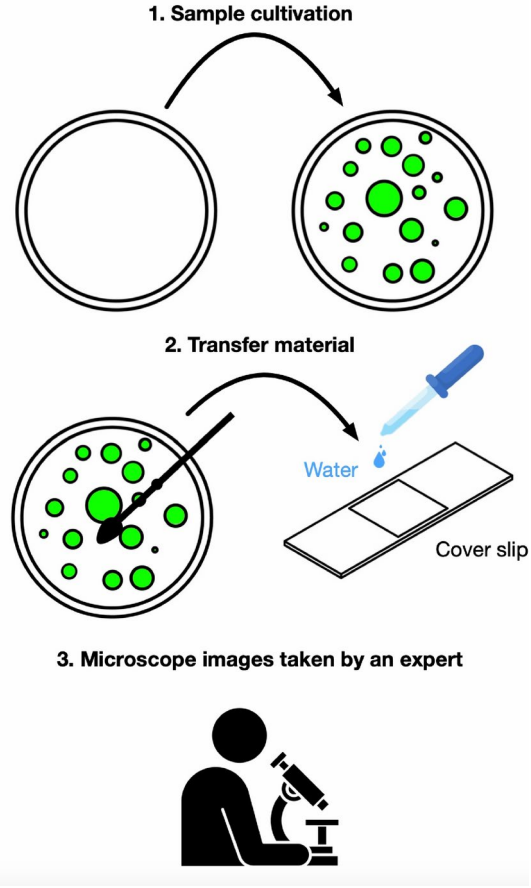


Figure 8. Data acquisition process for `Dataset1` and `Dataset2`.

Collectively, these efforts resulted in the creation of the largest database of soil microorganism images captured through microscopy for the ML purposes, setting a new standard in the literature. Datasets with all details are provided in Table 6.

The findings presented in this research are crucial due to the successful replication of the entire machine vision algorithm and ML methods across a range of datasets. These datasets vary significantly in size and were acquired using both automated and manual methods, as well as through two distinct types of microscopes. Such replication demonstrates that the system retains its performance consistently across different conditions, indicating a strong resistance to overfitting. For statistical proofs of this differences please refer to section *Feature analysis*.

This robustness is evident from the ability to maintain high accuracy and reliability in image analysis and classification, regardless of how the data was collected. The diverse operational settings in which the datasets were obtained, including automated and non-automated means, further validate the adaptability of the system. This versatility ensures that the system may be successfully deployed in a variety of real-world circumstances where consistency and reliability are critical. By proving effective across a broad spectrum of datasets and acquisition techniques, the system showcases its potential for scalability and generalization in the field of microbial image analysis.

Image preprocessing

All input images are automatically preprocessed in order to improve the segmentation phase which in turn has an essential impact on final classification accuracy. Indeed, the primary challenge associated with microscopic images lies in managing noise interference and low contrast impact. Another significant consideration is the presence of scale labels positioned at the bottom-left corner of each image for `Dataset1` and `Dataset2` only. It is important to note that the colors of these labels may vary which potentially brings distortion to the generated results. The latter occurs due to the automatic color selection of the labels by a microscopic software. Image masks that differentiate image to label image foreground and background are prepared by thresholding, whose values are empirically chosen. Consequently, the labels are concealed by replacing their area with the average color value of the complete image. The following stage involves converting the given image I^{RGB} in RGB color space to grayscale forming image I^{gray}, which can be subjected to further processing. To mitigate image noise, this study applies Gaussian and Wiener filters[22]. Finally, to improve the clarity of image features and to aid in distinguishing microorganisms from the background, the Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm[37] is employed.

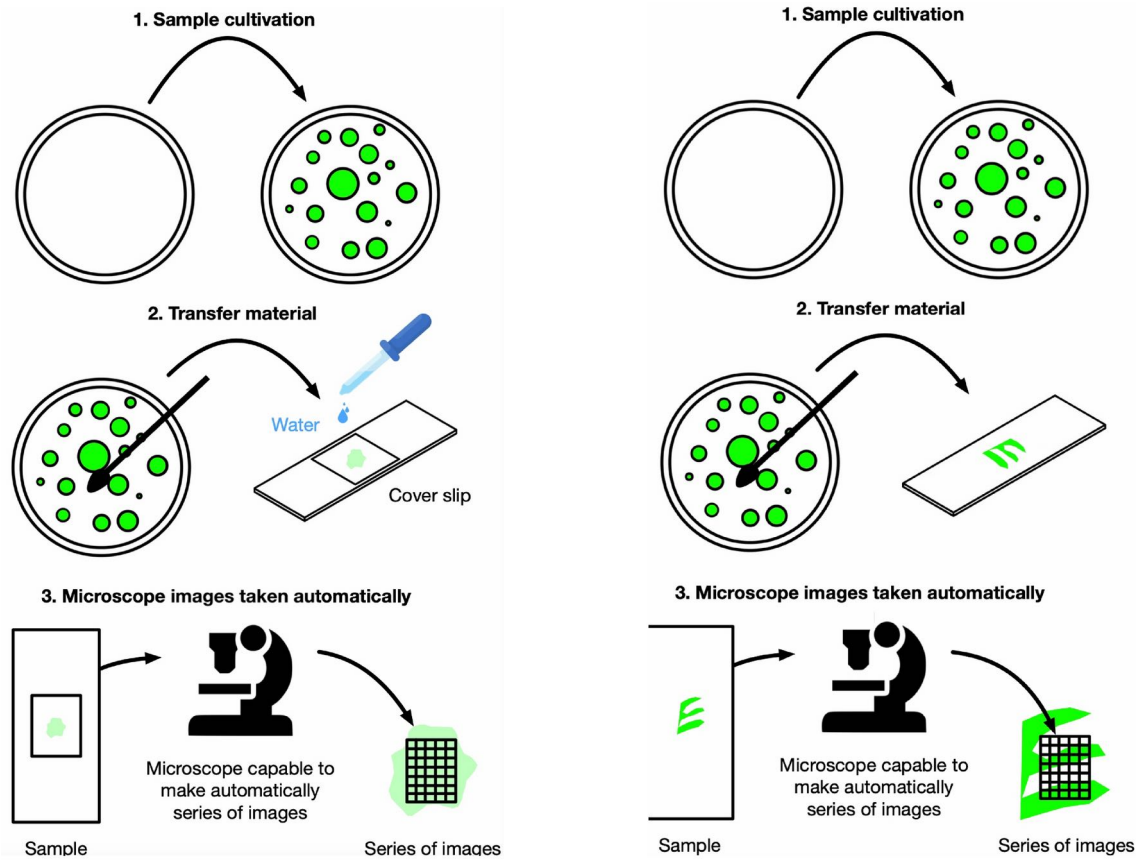


Figure 9. Automatized data acquisition process for given datasets creation.

Dataset	Total Instances	*Verticillium*	*Trichoderma*	*Phytophthora*	*Fusarium*	*Purpureocillium*
Dataset1	128	25	26	20	57	0
Dataset2	303	25	89	20	109	60
Dataset3	960	240	386	0	334	0
Dataset4	1279	260	240	0	240	539
Dataset5	196	49	49	0	49	49

Table 6. Summary of the datasets with specified number of images per class.

Image segmentation

Majority of the microscopic images in the datasets exhibit an imbalanced ratio between the area of the background and the area of the region of interest (ROI) that contains the soil microorganisms. Most of the area on microscopic pictures is covered by background information irrelevant to the classification task e.g. test-tube, medium for the cultivation of bacteria or minor impurities. Therefore, background has to be separated from the ROI. To address this issue, a binary image mask is created, assigning a value of 1 to each pixel that belongs to the ROI and a value of 0 to each pixel that belongs to the background. One of the most commonly used methods to perform image segmentation is an Otsu method[38] that calculates one threshold value globally for the whole image upon minimizing weighted sum of variances within ROI and background pixel values and maximizing weighted sum of variances between them.

According to[39], the Otsu method is known to produce weak results when the area of the ROI is imbalanced in comparison to the background area. Due to the above limitation a method called Adaptive Image Thresholding[22] is used.

The mask obtained using adaptive image threshold needs further image processing using morphological operations. Open operation is used to remove small objects in image and close operation is applied to join together objects that are close to each other. Both operations are applied to remove segmentation imperfections caused by the noise occurrence within original image (the so-called salt and pepper noise removal—see[22]).

Despite incorporating morphological operations, the thresholding results remain unsatisfactory. Classification accuracy of the ELM classifier applied on features computed for the generated mask renders 82.7% for

Dataset1. Unsatisfactory results are caused by the fact that input images are very complex as having different structures and textures. The segmentation accuracy is enhanced up to 87.4%, thus improving the performance of the classifier upon applying *k*-means algorithm[40]. Mask *M* is multiplied using Hadamard product[41] with I_t^{RGB} forming combined image I_c. On that image *k*-means method with $k = 3$ is run. As a result the original image is separated into three classes: i.e. a background and ROI area subsequently divided to two disjoint subclasses. Pixels that are classified to the brightest centroid represent a new, smaller area of ROI. The pixels with the highest contrast are assumed to have the most relevant information to distinguish microorganisms. In contrast, ROI pixels assigned to the centroid with lower values are classified in fact to the background as it may contain data disrupting the final results. In this manner a new mask *M* is generated. Finally, on a given mask, open and close morphological operations are applied. All crucial image segmentation stjpg contributing to the above algorithm are illustrated in Fig. 10 and the results from various datasets in Fig. 11.

Features calculation

Image features are computed as descriptors to measure statistical attributes of an image. The primary objective is to devise features that can effectively discriminate the different genera of microorganisms, enabling the formation of a robust classifier. Thus, color and texture-based handcrafted features are calculated in this study that are explained in the forthcoming subsections.

Color features

Color features are computed on combined images I_c generated by Hadamard product of original image I_t^{RGB} with its mask *M* introduced in the previous section. Conversion of an image from RGB color values to a different color space prior to feature calculation can enhance classification accuracy[42]. Indeed, experiments show such increase once conversion from RGB to $I_1I_2I_3$ color space is applied. First set of color features contains 8 statistical features, each calculated separately for three image color channels of color space $I_1I_2I_3$[43] to which I_c was previously converted. Every image pixel of coordinates *x* and *y* in RGB image is a vector containing 3 values for the respective RGB color space i.e. $I_{RGB}[x, y] = [R[x, y], G[x, y], B[x, y]]$. Conversion from RGB to $I_1I_2I_3$ is made upon applying the following operation:

$$I_{I_1I_2I_3}[x, y] = I_{RGB}[x, y] \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & -1/2 \\ -1/4 & 1/2 & -1/4 \end{bmatrix}.$$

Histograms for images I_t^{RGB} and I_c (denoted by *H* and *h*, respectively, generated upon conversion to $I_1I_2I_3$ color space) are calculated for every color space channel. Image histogram is a vector *h*, where *h*[*i*] is a number of occurrences of image pixels with the corresponding intensity value ranging within $0 \leq i \leq 255$. Having both

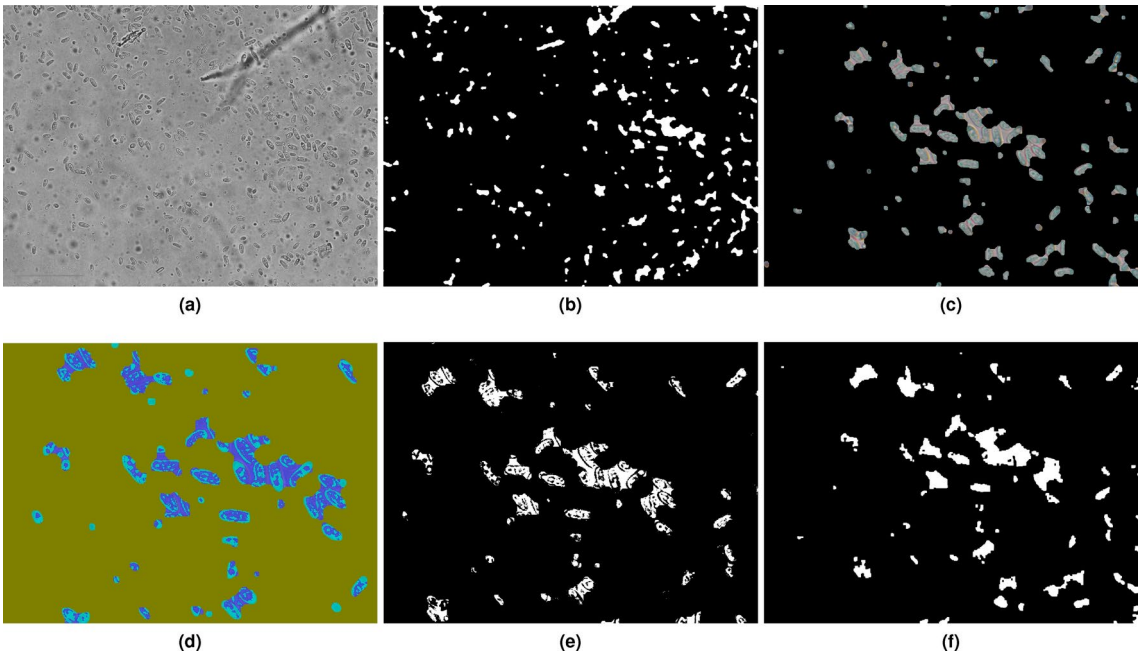
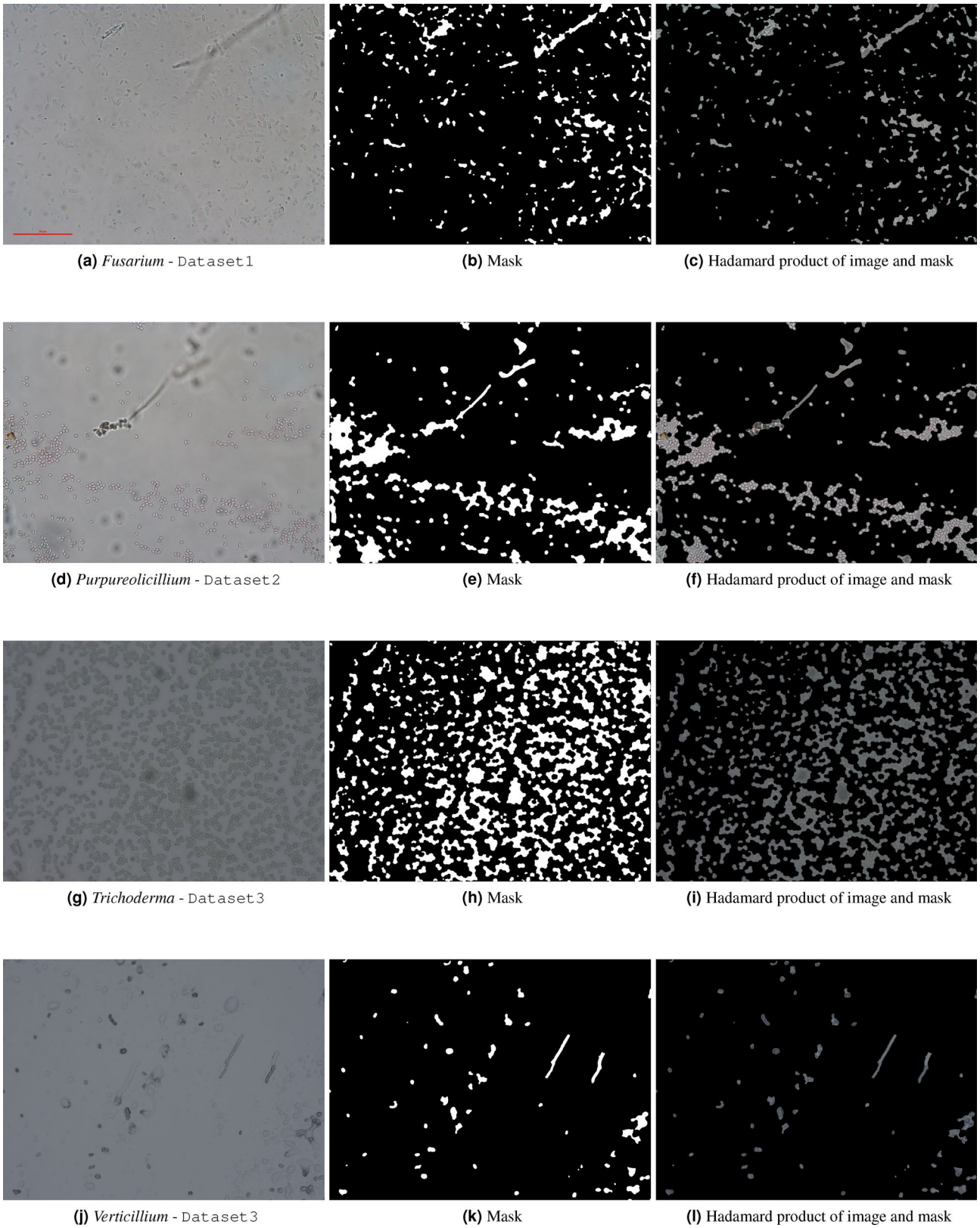


Figure 10. Segmentation process: (**a**)—an input image, (**b**)—fragment of an image mask upon applying Adaptive Image Thresholding and morphological operations on a, (**c**)—Hadamard product of a with b, (**d**)—an image created using *k*-means method on d, (**e**)—an image as a result of choosing pixels that are assigned to the brightest centroid of d and finally (**f**)—an image upon applying morphological operations to the previous step image.



(a) *Fusarium* - `Dataset1` **(b)** Mask **(c)** Hadamard product of image and mask

(d) *Purpureolicillium* - `Dataset2` **(e)** Mask **(f)** Hadamard product of image and mask

(g) *Trichoderma* - `Dataset3` **(h)** Mask **(i)** Hadamard product of image and mask

(j) *Verticillium* - `Dataset3` **(k)** Mask **(l)** Hadamard product of image and mask

Figure 11. Images and corresponding masks generated for various microorganism genera and datasets.

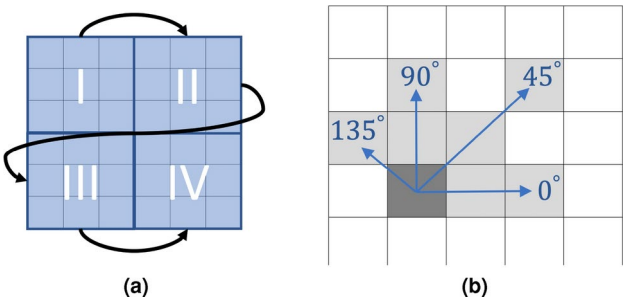


Figure 12. (**a**) Sliding window W over an image I_q, with disjoint regions I, II, III and IV used for calculation of pixel co–occurrences in their respective proximity. (**b**) Analyzed pixels in the neighborhood of pixel of interest $p = w_{42}$ for distance $d = 2$ and directions $\alpha = 0^\circ, 45^\circ, 90^\circ, 135^\circ$.

RGB	HSV	HSL	HSI	XYZ	LUV	LAB	Y'UV	Y'IQ	YCbCr	YDdDr	$I_1I_2I_3$	$C_1C_2C_3$
87.4	84.2	83.0	81.0	77.2	81.2	81.6	74.6	78.6	82.8	80.9	81.0	81.3

Table 7. Mean accuracy calculated from 50 runs of 10% cross-validation and ELM with mish activation function and 1000 neurons on `Dataset1` using different color spaces images representation in GLCM features calculation.

histograms the following features are calculated: mean and standard deviation of H; mean, standard deviation, kurtosis, skewness, entropy and energy of h[36].

The second set of color features is based on image I_c converted into *XYZ* color space. In contrast to the previous case, the calculated measures rely directly on image pixels, not histogram distribution. The latter permits to extract additional information on pixel spatial relationships which in turn is unavailable from image histogram[44]. The *XYZ* color space is selected after comparison of classification accuracy for the different color spaces. For every *X*, *Y* and *Z* color channel four statistical measures are calculated separately i.e. mean, standard deviation, kurtosis and skewness[36] yielding in total 12 features.

Texture features

Image texture defines spatial relations of chrominance or luminance of the image pixels.

Grey-level co-occurrence matrix (GLCM)

GLCM introduced by Haralick[45] is a method to extract spatial relations of pixels for a given angle direction α and maximal distance *d*. To apply the GLCM algorithm, the first step involves quantizing the image into q levels, resulting in the creation of an image I_q. Next, four co-occurrence matrices $M^{\alpha}_{q\times q}$ are defined, each computed for a different angle α (i.e., 0, 45, 90 degrees and 135 degrees). Here all coefficients of $M^{\alpha}_{q\times q}$ are initialized to zero. Examination of pixels relations is performed over their local neighborhood using window W of size $m \times n$ pixels. In sequel, W is iteratively moved throughout the image I assuming that coordinates of I_q are positioned in the top left corner. (see Fig. 12). At the first iteration of algorithm matrix W contains pixels of $I_q[y, x]$ with indices satisfying $0 < y \leq m$ and $0 < x \leq n$. Next W is moved horizontally so that it does not overlap with the previous window - e.g. indices of $I_q[y, x]$ for the second window range over $0 < y \leq m$ and $n < x \leq 2n$. If such left-to-right shift of W is not anymore possible such window is shifted vertically to underneath-left boundary position (e.g. after the first horizontal shifting movement the indices run over $m < y \leq 2m$ and $0 < x \leq n$ as illustrated in Fig. 12 (see window III). GLCM algorithm terminates once none of the above stjpg is further possible (i.e. W is too close to the right bottom corner of the I).

Every obtained W is the subject of further calculations. Co-occurrences of pixel values $q_i = 0, \ldots, q-1$ are noted for a direction $\alpha = 0^\circ, 45^\circ, 90^\circ, 135^\circ$ and maximal distance *d*. Let $w_{11}, \ldots, w_{mn}$ be coefficients of a given matrix W. Iterating through all of the coefficients one analyzes spatial correlations in their neighborhood. Currently analyzed pixel (coefficient of W in certain iteration) is named pixel of interest p. Definition of directions for maximal distance $d = 2$ and $p = w_{42}$ is presented in Fig. 12. Assume p is $w_{42} = 0$, $\alpha = 0^\circ$ and $d = 2$. Rows in M^{0° store the value of p and columns values of co-occurrence pixels being in relation to p. Since $w_{42} = 0$, the only coefficients in the first row of M^{0° can be incremented. Then, based on Fig. 7, w_{43} and w_{44} should be now analyzed. Coefficient $w_{43} = 0$ implies that value in M^{0° in the first column ($M^{0^\circ}[1, 1]$) is incremented. Analogously, for $w_{43} = 3$ $M^{0^\circ}[1, 4]$ is increased. Both stjpg are presented in Fig. 13. In the same manner calculations for every W are performed analyzing all of the w_{ij} for a direction α. Algorithm's result is four matrices: M^{0°, M^{45°, M^{90° and M^{135° whose coefficients are then converted to the probabilities by dividing each of its values by *mn*.

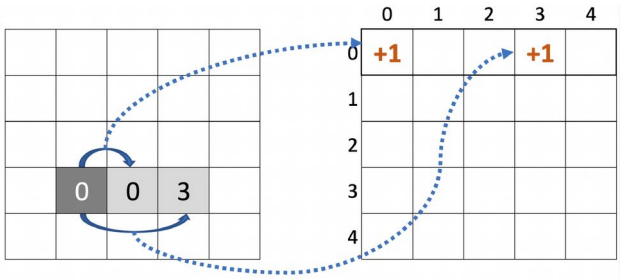


Figure 13. Filling M^{α} based on pixel of interest $p = w_{42}$ for distance $d = 2$ and direction $\alpha = 0^{\circ}$.

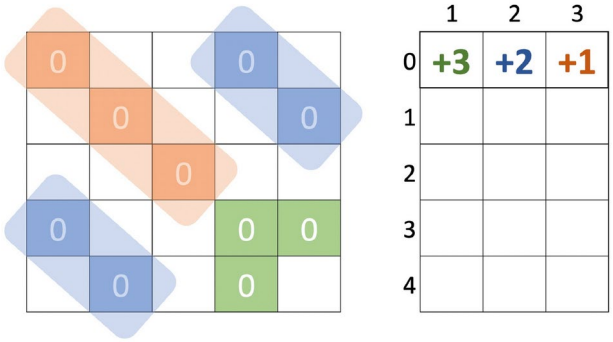


Figure 14. Filling M^{α} (right) for maximum run-length $r = 3$, $\alpha = 45^{\circ}$ and pixel value $q_i = 0$ based on a given W (left).

In the next step, the following statistical measures called GLCM features based on coefficients in matrices M^{α} can be computed: *Contrast, Correlation, Energy, Homogeneity, Autocorrelation, Cluster Prominence, Inverse Difference and Dissimilarity*[46]. Noticeably each of these features is calculable separately for each M^{α}, where $\alpha = 0^{\circ}, 45^{\circ}, 90^{\circ}, 135^{\circ}$. Having computed $Energy^{0^{\circ}}$, $Energy^{45^{\circ}}$, $Energy^{90^{\circ}}$ and $Energy^{135^{\circ}}$ their mean $Energy = 0.25(Energy^{0^{\circ}} + Energy^{45^{\circ}} + Energy^{90^{\circ}} + Energy^{135^{\circ}})$ is next derived. Originally GLCM based features are calculated on the grayscale image i.e. I_t^{gray}. A further improvement is obtained here once GLCM features are computed separately for each of the RGB image I_t^{RGB} channel yielding in total 24 features. In contrast to the work by Duan et al.[43], which proposes an extension that utilizes images in a different color space than RGB for GLCM feature computation, the experiments incorporating different color spaces for GLCM feature computation for microscopic images of microorganisms do not result in an improvement of classification accuracy. Indeed, the best results presented in Table 7 are reached for standard RGB color space, whereas other color channels might decrease classification accuracy from 87.4% to 74.6% using Y'IQ color space and 50 times repeated 10-Fold cross-validation technique and ELM with mish activation function and 1000 neurons on `Dataset1`.

Gray level run length matrix (GLRLM)

In general GLCM is not the only method of texture features extraction. Commonly GLRLM[47] is used together with GLCM. GLRLM creates features that describe spatial relationships between pixels having values of a given length of co-occurrence sequences and follows the concept of a run-length encoding[48] outlined below. Non-overlapping sliding of window W similarly to GLCM is performed on the image quantized to the q levels. GLRLM is defined as four matrices M^{α} each of size $q \times r$, where r is a maximal run-length of co-occurrence pixels. Rows of that matrix are described as gray level values $q_i = 0, \ldots, q-1$ and columns as run-lengths $r_j = 1, \ldots, r$.

For every obtained W run-length of co-occurrence pixel values q for direction α and maximal run-length r is computed. Assume $w_{11}, \ldots, w_{mn}$ form the coefficients of particular W. The directions are similarly defined as in GLCM method (see Fig. 12). In GLRLM the whole window W for each of the gray levels q_i is analyzed. To illustrate the latter for a given W with $\alpha = 45^{\circ}$ a maximal run-length $r = 3$ and take $q_i = 0$ were set. Then number of run-length sequences of length $r_j = 1, \ldots r$ is noted. Based on such choice, the values in $M^{45^{\circ}}$ are incremented e.g. there are 2 sequences of length 2 (means that some row in 2nd column should be enlarged) for gray level equal 0 (first row), so coefficient of $M^{45^{\circ}}[1, 2]$ is increased by 2 (see Fig. 14).

Similarly, all other entries of matrix M^{α} are incremented for every window W based on input image and for every gray level q_i. Consequently all coefficients of M^{α} are divided by qr. As a result matrix M^{α} contains now probabilities of co-occurrence gray levels q_i for run-length r_j and maximum run-length r for a given angle α. Based on computed probabilities the following statistical measures for GLRLM (see Zhou et al.[49]) are calculated: *Short Run Emphasis (SRE), Long Run Emphasis, Grey Level Non-uniformity, Run Length Non-uniformity, Run*

Percentage, Low Grey Level Run Emphasis, High Grey Level Run Emphasis, Short Run Low Gray Level Emphasis, Short Run High Gray Level Emphasis, Long Run Low Gray Level Emphasis and *Long Run High Gray Level Emphasis*. Similarly to GLCM, each of those features is calculated for four different directions α and the final value of them is defined as e.g. $SRE = 0.25(SRE^{0^\circ} + SRE^{45^\circ} + SRE^{90^\circ} + SRE^{135^\circ})$. Experiments to improve classification accuracy based on GLRLM features are conducted not to convert image to grayscale, but to calculate GLRLM on each of the color channels separately. Unfortunately, the same approach in case of GLRLM did not improve the results. A comparison is performed between the use of GLRLM feature generation based on images I_t^{RGB} and I_c. Surprisingly, better results are achieved using image I_t^{RGB}, whereas conducting the same experiment on GLCM do not lead to further improvement in the results.

Feature generation summary
In total for each dataset the same set of features organized into three color spaces: RGB, I1I2I3, and XYZ (as explained in details in section before). The following abbreviations apply:

- RGB: Red (R), Green (G), Blue (B)
- I1I2I3: Intensity channels I1, I2, I3
- XYZ: X, Y, Z color space

Each color channel has the following statistical features:

- I1, I2, I3: Mean, Std, Mean (H), Std (H), Kurtosis (H), Skewness (H), Entropy (H), Energy (H)
- X, Y, Z: Mean, Std, Kurtosis, Skewness
- GLCM (Gray-Level Co-occurrence Matrix) features:
  - Contrast, Correlation, Energy, Entropy, Homogeneity, Autocorrelation, Cluster Prominence, Inverse Difference, Dissimilarity (for R, G, B channels)
- GLRLM (Gray-Level Run Length Matrix) features: SRE, LRE, GLN, RLN, RP, LGRE, HGRE, SRLGE, SRHGE, LRLGE, LRHGENotably the mathematical equations behind these metrics (e.g., Mean, Kurtosis, Skewness, GLCM, GLRLM) can be obtained from corresponding well-known machine vision literature[22,45].

Extreme learning machine
The ELM network is a fully connected feed-forward neural network (see Huang et. al.[50]) designed to solve classification problems. This network consists of one hidden-layer with McCulloch-Pitts neurons[51]. The precise optimal number of hidden-layer neurons is each time evaluated empirically as there is no up-front theoretical method determining such value to maximize the classification accuracy. ELM is characterized by a simple topology structure. In network learning, the utilization of the Moore-Penrose pseudoinverse operation[50] generally involves the application of singular value decomposition (SVD). The latter permits to shorten a time-consuming iterative process of adjusting values of network's weights.

Assume that after image segmentation, feature calculation and feature selection dataset consists of N images. It can be described as $\eta = \{(x_i, t_i)\}_{i=1}^{N}$, where x_i is i-th vector of d features and t_i is i-th label of class to which selected image belongs. Then input layer of ELM has d neurons, defining also input matrix X introduced in (1). The number of output layer neurons coincides with the number of image classes M determining the classification task. Thus the output layer is numerically represented by a vector $y = \{y_i\}_{i=1}^{M}$, where $y_i \in \mathbb{R}$. Note that here the maximal value in vector y at index y_{max} is understood as assigning an input image to the y_{max} class. In order to test the ELM network, the i-th class label t_i needs to be appropriately formatted and compared to y. A possible approach is to format t_i as $\{t_{ij}\}_{j=0}^{m}$ that has all values set to zero except one element at p-th index that is set to one. Then t_i can be written as $t_i = [0, \ldots, 1, \ldots, 0]$ where p-th element indicates that i-th image affiliates to p-th class of images ($t_{i_p} = 1$). Only the greatest value of y_i at index y_i^{max} is chosen. Correct classification is observed if and only if $t_i[y_i^{max}] = 1$ for i-th picture. This concept of formatting is called One-Hot Encoding or $1 - of - K$ scheme[52].

Let now L be a number of units in single hidden layer network that is chosen empirically. Since ELM is a fully connected network, weights between input layer and hidden one are represented by a matrix W from (1), where w_{ij} denotes the weight between i-th neuron in input layer and j-th neuron of hidden layer. The corresponding bias values are assumed to be represented by a vector $b = \{b_i\}_{i=1}^{N}$. In ELM network both W and b are filled with values that are randomly chosen from the interval $[-1, 1]$ using a uniform distribution. Then the outputs of hidden units are calculated as expressed in matrix H (see (1)), where f is any continuous function $f : \mathbb{R} \to \mathbb{R}$.

$$X = \begin{bmatrix} x_{11} & \ldots & x_{1N} \\ \vdots & \ddots & \vdots \\ x_{d1} & \ldots & x_{dN} \end{bmatrix}, \quad W = \begin{bmatrix} w_{11} & \ldots & w_{1L} \\ \vdots & \ddots & \vdots \\ w_{d1} & \ldots & w_{dL} \end{bmatrix}, \quad H = \begin{bmatrix} f\left(\sum_{i=1}^{d} x_{i1} w_{i1} + b_1\right) & \ldots & f\left(\sum_{i=1}^{d} x_{i1} w_{iL} + b_1\right) \\ \vdots & \ddots & \vdots \\ f\left(\sum_{i=1}^{d} x_{iN} w_{i1} + b_N\right) & \ldots & f\left(\sum_{i=1}^{d} x_{iN} w_{iL} + b_N\right) \end{bmatrix}. \quad (1)$$

Weights β between hidden and output layers are still unknown. In order to obtain them, the equitation $Y = H\beta$ has to be evaluated. Indeed since H with probability equal to 1 is irreversible and $\|H\beta - Y\| = 0$ (see Huang et al.[53]), the system $Y = H\beta$ cannot be directly solved. Instead, one finds the optimal parameter $\hat{\beta} = \underset{\beta}{argmin} \ \|H\beta - T\|^2 = H^{\dagger}T$, where $H^{\dagger}$ is the Moore-Penrose generalized inverse of H (see Rao and Mitra[54]) and $T = t_{i\,i=1}^{N}$ describing image appurtenance to the given class formed using concept of one hot

encoding mentioned in the above paragraph. Moore-Penrose pseudoinverse of a real matrix H must satisfy following conditions:

1. $HH^{\dagger}H = H$, 2. $H^{\dagger}HH^{\dagger} = H^{\dagger}$, 3. $(HH^{\dagger}H)^{T} = HH^{\dagger}$, 4. $(H^{\dagger}H)^{T} = H^{\dagger}H$.

Pseudoinverse is known to be unique and pseudoinverse of a non-singular quadratic matrix is identical to the ordinary inverse. For matrices that are singular pseudoinverse solves $Y = H\beta$ in terms of least-squares problem. Calculating $H^{\dagger}$ gives solution $\hat{\beta}$ so that $H\hat{\beta}$ is as close as possible to Y in terms of Euclidean distance.

Note here that H generically corresponds to either overdetermined or underdetermined system. Additionally, there can be many methods to calculate $H^{\dagger}$, one of them recalled here is orthogonal projection method (see Rao and Mitra[54]). If $L > d$ (number of hidden units is greater than number of input units) then system $Y = H\beta$ is underdetermined and $H^{\dagger} = H^{T}(HH^{T})^{-1}$, otherwise system $Y = H\beta$ is overdetermined and $H^{\dagger} = (H^{T}H)^{-1}H^{T}$. In practical applications of ELM $L \gg d$ eventuates. In classification tasks L usually totals a few thousands. Then, operation of inverse HH^{T} could be time-consuming especially taking into account the fact that operation of matrix pseudoinverse cannot be parallelized. To improve learning speed of network QR matrix decomposition is used (see Strang[55]). QR decomposition of matrix A is defined as $A = QR$, where R is upper triangular and Q is orthogonal ($QQ^{T} = \mathbb{I}$). Using QR decomposition, the matrix $H^{\dagger}$ can be computed according to:

$$H^{\dagger} = H^{T}(HH^{T})^{-1} = R^{T}Q^{T}(R^{T}Q^{T}QR)^{-1} = R^{T}Q^{T}(R^{T}R)^{-1} = Q^{T}R^{-1}.$$

The ELM network gives a solution that minimizes mean square error between predicted $\hat{Y}$ and actual Y. Thanks to assigning random values to weights and bias between input and hidden layer, the trained network is very resistant to overtraining. The computed solution is not a local minimum of optimization criterion like in MLP with backpropagation algorithm based on gradient descent[50,56]. The network training process is much faster than other Machine Learning methods like MLP, RF and SVM. The main drawback of this process relies on finding L and f that maximize classification accuracy.

Data availability

The data that support the findings of this study are available from the corresponding author, K.S., upon reasonable request.

Received: 18 July 2024; Accepted: 3 December 2024
Published online: 28 December 2024

References

1. Mendes, R., Garbeva, P. & Raaijmakers, J. M. The rhizosphere microbiome: Significance of plant beneficial, plant pathogenic, and human pathogenic microorganisms. *FEMS Microbiol. Rev.* **37**, 634–663. https://doi.org/10.1111/1574-6976.12028 (2013).
2. Yang, J., Kloepper, J. W. & Ryu, C.-M. Rhizosphere bacteria help plants tolerate abiotic stress. *Trends Plant Sci* **14**, 1–4. https://doi.org/10.1016/j.tplants.2008.10.004 (2009).
3. Schinner, F., Öhlinger, R., Kandeler, E. & Margesin, R. *Methods in Soil Biology* (Springer, Berlin, 2011).
4. Dini-Andreote, F. & Elsas, J. D. v. The soil microbiome-an overview. In *Modern Soil Microbiology* 37–48 (CRC Press, 2019). https://doi.org/10.1201/9780429059186-3.
5. Franco-Duarte, R. et al. Advances in chemical and biological methods to identify microorganisms-from past to present. *Microorganisms* **7**, 130. https://doi.org/10.3390/microorganisms7050130 (2019).
6. Gayathri, S., Wise, D. W., Shamini, P. B. & Muthukumaran, N. Image analysis and detection of tea leaf disease using deep learning. In *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 398–403 (2020). https://doi.org/10.1109/ICESC48915.2020.9155850.
7. Maeda-Gutiérrez, V. et al. Comparison of Convolutional Neural Network architectures for classification of tomato plant diseases. *Appl. Sci.*[SPACE]https://doi.org/10.3390/app10041245 (2020).
8. Howlader, M. R., Habiba, U., Faisal, R. H. & Rahman, M. M. Automatic recognition of guava leaf diseases using Deep Convolution Neural Network. In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)* 1–5 (2019). https://doi.org/10.1109/ECACE.2019.8679421
9. Ayele, A. G., Wheeler, T. A. & Dever, J. K. Impacts of verticillium wilt on photosynthesis rate, lint production, and fiber quality of greenhouse-grown cotton (gossypium hirsutum). *Plants* **9**, 857. https://doi.org/10.3390/plants9070857 (2020).
10. Keykhasaber, M., Thomma, B. & Hiemstra, J. Verticillium wilt caused by *Verticillium dahliae* in woody plants with emphasis on olive and shade trees. *Eur. J. Plant Pathol.* **150**, 1–17. https://doi.org/10.1007/s10658-017-1273-y (2017).
11. Buszewski, B., Rogowska, A., Pomastowski, P., Złoch, M. & Railean-Plugaru, V. Identification of microorganisms by modern analytical techniques. *J. AOAC INT.* **100**, 1607–1623. https://doi.org/10.5740/jaoacint.17-0207 (2017).
12. Zieliński, B., Sroka-Oleksiak, A., Rymarczyk, D., Piekarczyk, A. & Brzychczy-Włoch, M. Deep learning approach to describe and classify fungi microscopic images. *PLOS ONE* **15**, 1–16. https://doi.org/10.1371/journal.pone.0234806 (2020).
13. Treebupachatsakul, T. & Poomrittigul, S. Microorganism image recognition based on deep learning application. In *2020 International Conference on Electronics, Information, and Communication (ICEIC)* 1–5 (2020). https://doi.org/10.1109/ICEIC49074.2020.9051009.
14. Khasim, S., Ghosh, H., Rahat, I., Shaik, K. & Yesubabu, M. Deciphering microorganisms through intelligent image recognition: Machine learning and deep learning approaches, challenges, and advancements. *EAI Endorsed Trans. Internet Things*[SPACE]https://doi.org/10.4108/eetiot.4484 (2023).
15. Qu, K., Guo, F., Liu, X., Lin, Y. & Zou, Q. Application of machine learning in microbiology. *Front. Microbiol.*[SPACE]https://doi.org/10.3389/fmicb.2019.00827 (2019).
16. Jiang, Y., Luo, J., Huang, D., Liu, Y. & Li, D.-D. Machine learning advances in microbiology: A review of methods and applications. *Front. Microbiol.*[SPACE]https://doi.org/10.3389/fmicb.2022.925454 (2022).

17. Kotwal, S. Automated bacterial classifications using machine learning based computational techniques: Architectures, challenges and open research issues. *Arch. Comput. Methods Eng.* **10**, 2469–2490. https://doi.org/10.1007/s11831-021-09660-0 (2022).
18. Rani, P., Kotwal, S., Manhas, J., Sharma, V. & Sharma, S. Machine learning and deep learning based computational approaches in automatic microorganisms image recognition: Methodologies, challenges, and developments. *Arch. Comput. Methods Eng.* **29**, 1801–1837. https://doi.org/10.1007/s11831-021-09639-x (2022).
19. Liu, L. et al. Automatic identification of fungi under complex microscopic fecal images. *J. Biomed. Opt.* **20**, 076004. https://doi.org/10.1117/1.JBO.20.7.076004 (2015).
20. Tahir, M. W. et al. Fungus detection through optical sensor system using two different kinds of feature vectors for the classification. *IEEE Sens. J.* **17**, 5341–5349. https://doi.org/10.1109/JSEN.2017.2723052 (2017).
21. Sas-Paszt, L. et al. Symbio bank—A collection of beneficial soil microorganisms. *Short Commun.*[SPACE]https://doi.org/10.5555/20153115354 (2016).
22. Gonzalez, R. C. & Woods, R. E. *Digital Image Processing* 4th edn. (Pearson, London, 2018).
23. Feng, Z. A local invariant feature extraction and description method for microscopic image of bacteria. *Acta Microscopica* **29**, 1963–1970 (2020).
24. Raghavendra, C., Reddy, K. S. S., Shanmugathai, M. & Devipriya, A. Electron microscopy images for automatic bacterial trichomoniasis diagnostic classification separating and sorting of overlapping microbes. In *11th Annual International Conference (AIC) 2021: On Sciences and Engineering*, vol. 2613, 020089 (AIP Publishing, 2023). https://doi.org/10.1063/5.0110989.
25. Konopka, A., Kozera, R., Sas-Paszt, L., Trzcinski, P. & Lisek, A. Identification of the selected soil bacteria genera based on their geometric and dispersion features. *PLOS ONE* **18**, e0293362. https://doi.org/10.1371/journal.pone.0293362 (2023).
26. Watanabe, T. *Pictorial Atlas of Soil and Seed Fungi* 3rd edn. (CRC Press, Boca Raton, 2010).
27. Zhu, Z.-H., Hong, Q., Wu, L.-F., Wang, Q.-H. & Ma, M.-H. Early identification of male and female embryos based on uv/vis transmission spectroscopy and extreme learning machine. *Spectrosc. Spectral Anal.* **39**, 2780–2787. https://doi.org/10.3964/j.issn.1000-0593(2019)09-2780-08 (2019).
28. Prabhakaran, A. et al. Soil microbiome: Characteristics, impact of climate change and resilience. In *Understanding the Microbiome Interactions in Agriculture and the Environment*, 285–313 (Springer Nature, Singapore, 2022). https://doi.org/10.1007/978-981-19-3696-8_15.
29. Anderson, M. J. A new method for non-parametric multivariate analysis of variance. *Austral Ecol.* **26**, 32–46. https://doi.org/10.1046/j.1442-9993.2001.01070.x (2001).
30. Wold, S., Esbensen, K. & Geladi, P. Principal component analysis. *Chemom. Intell. Lab. Syst.* **2**, 37–52. https://doi.org/10.1016/0169-7439(87)80084-9 (1987).
31. Van Der Maaten, L. & Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**, 2579–2625 (2008).
32. Struniawski, K., Konopka, A. & Kozera, R. Performance evaluation of activation functions in Extreme Learning Machine. In *ESANN 2023 : Proceedings*, 351–356 (2023). https://doi.org/10.14428/esann/2023.es2023-31.
33. Struniawski, K. & Kozera, R. TfELM: Extreme learning machines framework with python and TensorFlow. *SoftwareX* **27**, 101833. https://doi.org/10.1016/j.softx.2024.101833 (2024).
34. Soleymani, S. & Mohammadzadeh, S. Comparative analysis of machine learning algorithms for solar irradiance forecasting in smart grids. https://doi.org/10.48550/arXiv.2310.13791 (2023).
35. Lundberg, S. M. & Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, Vol. 30, 4765–4774 (Curran Associates, Inc., 2017).
36. Kruk, M. et al. Computerized classification system for the identification of soil microorganisms. *AIP Conf. Proc.* **1648**, 6600187. https://doi.org/10.1063/1.4912894 (2015).
37. Pizer, S. M. et al. Adaptive histogram equalization and its variations. *Comput. Gr. Image Process.* **39**, 355–368. https://doi.org/10.1016/S0734-189X(87)80186-X (1987).
38. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man* **9**, 62–66. https://doi.org/10.1109/TSMC.1979.4310076 (1979).
39. Ershov, E., Korchagin, S., Kokhan, V. & Bezmaternykh, P. A generalization of Otsu method for linear separation of two unbalanced classes in document image binarization. *Comput. Opt.* **45**, 66–76. https://doi.org/10.18287/2412-6179-CO-752 (2021).
40. MacQueen, J. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability* 281–297 (1967).
41. Horn, R. A. & Johnson, C. R. *Matrix Analysis* (Cambridge University Press, Cambridge, 1985).
42. Meas-Yedid, V. et al. Automatic color space selection for biological image segmentation. *Proc. IAPR Int. Conf. Pattern Recogn.* **3**, 514–517. https://doi.org/10.1109/ICPR.2004.1334579 (2004).
43. Duan, G., Duan, F., Xu, Y., Gong, H. & Qu, X. Investigation of optimal segmentation color space of Bayer true color images with multi-objective optimization methods. *J. Indian Soc. Remote* **43**, 487–499. https://doi.org/10.1007/s12524-014-0424-2 (2015).
44. Stricker, M. & Orengon, M. Similarity of color images. In *SPIE Conference on Storage and Retrieval for Image and Video Databases*, Vol. 2420, 381–392 (1995). https://doi.org/10.1117/12.205308
45. Haralick, R., Shanmugam, K. & Dinstein, I. Textural features for image classification. *IEEE Trans. Syst. Man Cybern.* **SMC-3**, 610–621 (1973).
46. Lin, W., Hasenstab, K., Cunha, G. M. & Schwartzman, A. Comparison of handcrafted features and Convolutional Neural Networks for liver MR image adequacy assessment. *Sci. Rep.* **10**, 20336. https://doi.org/10.1038/s41598-020-77264-y (2020).
47. Galloway, M. M. Texture analysis using gray level run lengths. *Comput. Graph. Image Process.* **4**, 172–179. https://doi.org/10.1016/S0146-664X(75)80008-6 (1975).
48. Robinson, A. H. & Cherry, C. Results of a prototype television bandwidth compression scheme. *Proc. IEEE Inst. Electr.* **55**, 356–364. https://doi.org/10.1109/PROC.1967.5493 (1967).
49. Zhou, Y. et al. Prediction of overall survival and progression-free survival by the 18F-FDG PET/CT radiomic features in patients with Primary Gastric Diffuse Large B-Cell Lymphoma. *Mol. Imaging* **2019**, 5963607. https://doi.org/10.1155/2019/5963607 (2019).
50. Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K. Extreme learning machine: Theory and applications. *Neurocomputing* **70**, 489–501. https://doi.org/10.1016/j.neucom.2005.12.126 (2006).
51. Palm, G. Warren McCulloch and Walter Pitts: A logical calculus of the ideas immanent in nervous activity. In *Brain Theory* 229–230 (Springer Berlin, Heidelberg, 1986). https://doi.org/10.1007/978-3-642-70911-1_14.
52. Bishop, C. M. *Pattern Recognition and Machine Learning* (Springer, New York, 2006).
53. Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K. Extreme learning machine: A new learning scheme of feedforward neural networks. In *Proc. Int. Jt. Conf. Neural Netw.*, Vol. 2, 985–990 (2004). https://doi.org/10.1109/IJCNN.2004.1380068.
54. Rao, C. R. & Mitra, S. K. *Generalized Inverse of Matrices and Its Applications* (John Wiley & Sons, Hoboken, 1971).
55. Strang, G. *Linear Algebra and Learning from Data* 1st edn. (Wellesley Cambridge Press, Wellesley, 2019).
56. Huang, G.-B. What are extreme learning machines? Filling the gap between Frank Rosenblatt's dream and John von Neumann's Puzzle. *Cogn. Comput.* **7**, 263–278. https://doi.org/10.1007/s12559-015-9333-0 (2015).

Acknowledgements

This research was supported by The National Centre for Research and Development within the framework of the project BIOSTRATEG, grant number BIOSTRATEG3/344433/16/NCBR/2018.

Author contributions

K.S. developed and implemented the classification tool. K.S. and R.K. collaborated on the algorithm's conceptual development and contributed to writing the manuscript. K.S. and A.M-C. handled the microscopic imaging, while P.T. was responsible for sample preparation. L.S.P. contributed to the experimental concept. All authors reviewed and approved the final manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to K.S.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Chapter 4

Conclusions

Following its publication, the *TfELM* software was made publicly available on PyPI, which has significantly expanded its accessibility. Researchers and practitioners can now easily download the tool both from GitHub and PyPI, increasing the reach of the framework. Usage statistics indicate that the *TfELM* tool is actively utilized by end users, even though ELM remains a niche area within the broader field of machine learning. The project has received substantial attention, with data showing significant downloads and engagement. As shown in Figure 4.1, the tool was downloaded consistently, and the project garnered 984 views on GitHub in April 2025 alone, highlighting a growing community of users and contributors interested in the tool's capabilities.

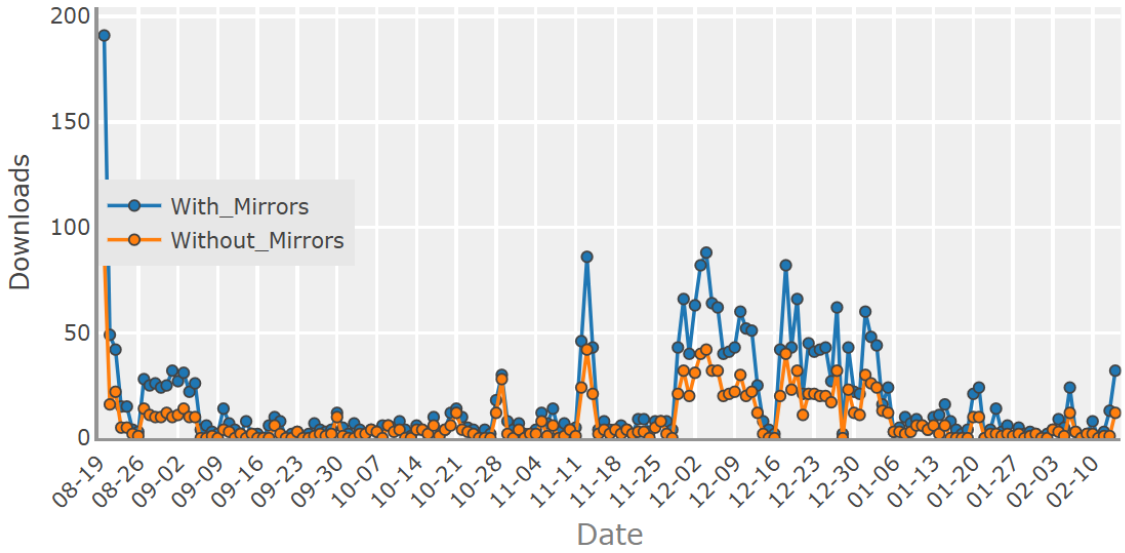


Figure 4.1: Daily download quantity of TfELM package in PyPI from https://pypistats.org/.

4.1 Activation function selection for ELM

A key contribution of this research was the study on activation function selection for Extreme Learning Machines (ELMs), which extended the initial hypothesis that model performance is highly dataset-dependent. The findings demonstrated that certain activation functions, such as Mish, offer substantial performance gains across a variety of

datasets. Specifically, the Mish activation function performed well on 9 out of 10 tested datasets, underscoring its robustness in various machine learning contexts. This is particularly noteworthy because the application of Mish in the context of ELM represents a novel contribution to the field, offering new insights into activation function selection.

The research also uncovered significant variability in ELM classifier performance based on the choice of activation function. In some cases, performance differences reached as much as 80 percentage points, emphasizing the importance of selecting the most appropriate activation function for each specific task. This finding highlights that the activation function choice is not merely a technical detail but a key factor that can drastically affect the efficacy of an ELM model. Consequently, this emphasizes the necessity of evaluating different activation functions on a task-specific basis in order to maximize the performance of ELM-based classifiers.

4.2 Metaheuristic Algorithms for weight fine-tuning

The PhD related study also explored the role of metaheuristic algorithms (MAs) for fine-tuning the weights connecting the input and hidden layers in ELMs. This research revealed that there is no single MA that consistently outperforms others across all datasets. This finding challenges common assumptions that some MAs are universally superior, reinforcing the idea that the effectiveness of a particular algorithm depends on the characteristics of the dataset at hand. Additionally, there was no clear correlation between the computational time required by an MA and the resulting accuracy of the model. The latter indicates that optimizing an ELM model is not a simple trade-off between computational efficiency and accuracy, but rather a complex evaluation of the specific algorithm's performance on a given task. Thus, practitioners should avoid relying on general assumptions and instead should conduct thorough evaluations of MAs in the context of their specific applications.

4.3 Random weight and bias generation stability in ELM

One of the more novel aspects of this research was to investigate the stability of random weight and bias generation in ELM. Traditional Random Number Generators (RNGs) that yielded the highest accuracy and stability in ELM performance often did not pass the stringent standards set by The U.S. National Institute of Standards and Technology (NIST) tests for RNGs. Conversely, RNGs that met the NIST standards did not consistently produce the best performance in terms of accuracy or stability. Such paradoxical relationship between statistical rigor and practical outcomes reveals a key insight: while on one hand adhering to high statistical standards is critical for some applications, on the other hand it may not always translate to optimal performance in machine learning models like ELM. This particular finding suggests that, in the context of ELM, the choice of RNG should prioritize empirical performance over theoretical statistical perfection, a principle that may extend to other areas of machine learning.

4.4 Comparative analysis of ELM versus CNN for soil fungi identification

Comparison of the performance of ELM and Convolutional Neural Networks (CNNs) in the task of soil fungi identification highlighted the remarkable strengths of ELM, particularly in terms of training and prediction speed. The results demonstrated that ELM outperformed CNN in both training time and prediction speed, with ELM being six times faster during training and twice as fast during prediction. This is particularly important in real-world applications where speed and efficiency are crucial, such as in large-scale microbiological studies or clinical diagnostics.

While CatBoost, a gradient boosting algorithm, achieved comparable accuracy to ELM, the latter's speed advantage makes it a more practical choice in scenarios where time constraints are critical. Additionally, CNNs, though often preferred in image-based tasks, took significantly longer to make predictions (0.11 seconds per prediction), which is twelve times longer than ELM's prediction time. The speed advantage of ELM makes it a compelling option for tasks where fast, real-time predictions are required.

Moreover, the comparative analysis revealed that ELM consistently outperformed CNN in terms of not only prediction speed but also accuracy, precision, recall, and ROC/AUC scores across all datasets. This was particularly notable when using automatically retrieved microscopic images, where ELM maintained superior performance. The robustness of ELM was further demonstrated by its ability to handle varying imaging conditions and microorganism strains, highlighting its adaptability to diverse datasets and real-world challenges. This finding positions ELM as a highly versatile model that can be applied across a range of domains, particularly in cases where both speed and accuracy are essential.

The results also illustrated that the performance benefits of ELM were not confined merely to individual datasets but were generalizable across combined datasets. This reinforces the practical applicability of ELM in large-scale tasks, such as environmental monitoring, clinical diagnostics, and industrial applications where multiple data sources are often combined for analysis.

Chapter 5

Future work and extensions

Future work on the *TfELM* framework should ensure the emphasize ensuring ongoing compatibility with the latest releases of *TensorFlow* to fully leverage the advancements and optimizations that each new version provides. As the machine learning eco-system evolves rapidly, continuous integration of novel approaches from recent publications in the ELM domain seems to be crucial. This will not only enhance the performance and capabilities of the *TfELM* framework but also will solidify its position as a versatile tool for both researchers and practitioners. Active engagement with the latest developments in deep learning and ELM methodologies will allow for the framework's continuous improvement and refinement, ensuring it remains relevant and effective tool in addressing emerging challenges within the ML field.

In the realm of hardware, a significant shift in Apple's chip architecture occurred after the introduction of the M2 chip, which, at the time of the original experiments, represented the cutting edge of performance for Apple Silicon devices. However, with the release of the M4 chip, which offers enhanced processing power and efficiency, future experiments will likely focus on evaluating the performance of the *TfELM* framework on M-Series processors with larger RAM capacities. Such shift will permit further for the analysis of their performance on larger datasets, where traditional hardware might struggle to handle the increased data load efficiently. The greater memory bandwidth and processing capabilities of the latest M-Series chips are expected to provide significant improvements in model training times and the overall handling of complex ML tasks.

One notable hardware feature that warrants further investigation is the scalability of Apple's Mac Studio, particularly the M2 Ultra chip, which supports up to an impressive 192GB of unified memory. Here the available amount of memory far exceeds that of conventional budget-friendly GPUs, which typically support up to 24GB of VRAM. This unique advantage positions Apple Silicon devices, especially those equipped with the M2 Ultra and its successors, as highly promising candidates for running large-scale models, including modern Large Language Models (LLMs). Many of these models require immense memory resources that often exceed the memory limits of traditional GPU setups, making the Mac Studio's unified memory architecture a valuable asset for handling such demanding workloads. Therefore, future work will involve leveraging this hardware to perform comparative analyses between traditional GPU-based setups and Apple's M-Series processors to assess their performance on computationally high-demand tasks, such as training LLMs or processing large datasets.

Furthermore, ongoing research efforts should be addressed toward optimizing ELM architectures, with the aim of enhancing both training efficiency and predictive perfor-

mance. One promising direction involves exploring the use of multilayer ELM architectures, which have shown potential for improving the representational power of the model. These architectures will be possibly combined with kernel mapping techniques derived from Mercer's theorem to create more sophisticated decision boundaries, potentially enhancing model performance. In addition, approximations of these kernel-based methods using the Nyström method will be investigated to evaluate their feasibility and computational efficiency in large-scale applications. This comparison enables to identify the most suitable kernel mapping approaches for various ELM architectures, guiding future developments in optimization techniques and enhancing the framework's applicability across different domains.

In the context of microorganism identification, future research extensions might focus on seeking for the expansion of the existing dataset to include a broader range of microorganism genera, with the goal of creating a more diverse and representative training set. Such expansion will facilitate more accurate and generalized identification models capable of recognizing a wider array of microbial species. Moreover, most likely efforts will be made to integrate phenotypic and molecular methods into the identification process, enabling species-level identification with higher precision. By incorporating DNA sequencing data, biochemical testing results, and morphological characteristics into the model training pipeline, the resulting system will not only improve the accuracy of microbial classification but also offer a more comprehensive tool for researchers in microbiology. The improvements above seem to be proposed for practical applications in clinical microbiology, environmental monitoring, and food safety, where on accurate species-level identification is critical.

Finally, future research may explore the potential for real-time microorganism identification systems that can operate in dynamic environments, such as clinical laboratories or field research settings. By integrating the *TfELM* framework with next-generation sequencing technologies and real-time data acquisition tools, this research aims to create an adaptive, high-performance model capable of rapid and accurate microorganism identification in challenging conditions. The mentioned advancements could have significant implications for both basic microbiological research and applied clinical diagnostics.

Chapter 6

Research curriculum vitae

ORCID: 0000-0002-4574-2986.
Google Scholar: Karol Struniawski
Scopus: Karol Struniawski

6.1 Education

- Master of Science in Computer Science, diploma with distinction obtained from the Faculty of Applied Informatics and Mathematics, Warsaw University of Life Sciences in 2021.
- Bachelor's degree in Computer Science and Econometrics from the Faculty of Applied Informatics and Mathematics, Warsaw University of Life Sciences in 2019.

6.2 Professional career

- Research and Teaching Assistant (a primary place of work), Institute of Information Technology, Warsaw University of Life Sciences (SGGW), since 2021.
- Intern, Budgeting and Analysis Department, Provident SA, July-December 2018.

6.3 Academic achievements

6.3.1 List of all PhD candidate publications

1. K. Struniawski and R. Kozera, "Bias or justice? Analyzing LLM sentencing variability in theft indictments across gender, ethnicity, and education factors," in *Computational Science – ICCS 2025. 25th International Conference, Singapore* (in press).
2. J. Sawicka *et al.*, "Design and characterization of antibacterial peptide nanofibrils as components of composites for biomaterial applications," *Current Protein & Peptide Science*, Article 26, 2025, DOI: 10.2174/0113892037353453241219185311.

3. K. Struniawski, A. Konopka, and R. Kozera, "Credibility of Randomness in Extreme Learning Machine," in *Trust and Artificial Intelligence: Development and Application of AI Technology*. Routledge, 2025, pp. 92–106, ISBN 978-1-032-62632-1, DOI: 10.4324/9781032627236-10.

4. K. Struniawski and R. Kozera, "*TfELM*: Extreme Learning Machines framework with Python and *TensorFlow*," *SoftwareX*, no. 27, 2024, Article number: 101833, pp. 1–9, DOI: 10.1016/j.softx.2024.101833.

5. K. Struniawski *et al.*, "Extreme Learning Machine for identifying soil-dwelling microorganisms cultivated on agar media," *Scientific Reports*, vol. 14, no. 31034, pp. 1–23, 2024, DOI: 10.1038/s41598-024-82174-4.

6. K. Struniawski, A. Konopka, and R. Kozera, "Exploring Apple Silicon's potential from simulation and optimization perspective," in *Computational Science – ICCS 2024. 24th International Conference*, Malaga (Spain), July 2–4, 2024, Proceedings, Part V, *Lecture Notes in Computer Science*, vol. 14836, pp. 35–42, 2024, ISBN 978-3-031-63775-9, DOI: 10.1007/978-3-031-63775-9_3.

7. K. Struniawski *et al.*, "Automated identification of soil fungi and Chromista through Convolutional Neural Networks," *Engineering Applications of Artificial Intelligence*, vol. 127B, no. 107333, pp. 1–12, 2024, DOI: 10.1016/j.engappai.2023.107333.

8. R. Wyszyński, K. Struniawski, and A. Konopka, "Augmented single instance -driven identification of fungal pathogens through the Convolutional Neural Networks," in *Modelling and Simulation'2024. The European Simulation and Modelling Conference ESM'2024*, San Sebastian (Spain), pp. 79–83, 2024, ISBN 9789492859334.

9. R. Wyszyński and K. Struniawski, "Residual Neural Networks in single instance-driven identification of fungal pathogens," *Machine Graphics & Vision*, vol. 32, no. 3/4, pp. 45–64, 2023, DOI: 10.22630/MGV.2023.32.3.3.

10. K. Struniawski, R. Kozera, and A. Konopka, "Performance of selected nature-inspired Metaheuristic Algorithms used for Extreme Learning Machine," in *Computational Science – ICCS 2023. 23rd International Conference*, Prague (Czechia), July 3–5, 2023, Proceedings, Part III, *Lecture Notes in Artificial Intelligence*, vol. 10475, pp. 498–512, 2023, ISBN 978-3-031-36023-7, DOI: 10.1007/978-3-031-36024-4_38.

11. A. Konopka, K. Struniawski, and R. Kozera, "Classification performance of Extreme Learning Machine Radial Basis Function with K-means, K-medoids and Mean Shift clustering algorithms," in *Computational Science – ICCS 2023, 23rd International Conference*, Prague (Czechia), July 3–5, 2023, Proceedings, Part IV, *Lecture Notes in Artificial Intelligence*, vol. 10476, pp. 171–186, 2023, ISBN 9783031360268, DOI: 10.1007/978-3-031-36027-5_13.

12. K. Struniawski, A. Konopka, and R. Kozera, "Metaheuristic Algorithms in Extreme Learning Machine for selection of parameters in activation function," in *Modelling and Simulation'2023. The 2023 European Simulation and Modelling Conference*, Toulouse (France), pp. 239–244, 2023, ISBN 978-9-492859-28-0.

13. K. Struniawski, A. Konopka, and R. Kozera, "Performance evaluation of activation functions in Extreme Learning Machine," in *ESANN 2023:Proceedings*, Brugge (Belgium), pp. 351–356, 2023, ISBN 9782875870872, DOI: 10.14428/esann/2023.es2023-31.

14. A. Konopka, K. Struniawski, and R. Kozera, "Performance analysis of Residual Neural Networks in soil bacteria microscopic image classification," in *Modelling and Simulation'2023. The 2023 European Simulation and Modelling Conference*, Toulouse (France), pp. 144–149, 2023, ISBN 978-9-492859-28-0.

15. A. Konopka et al., "Classification of soil bacteria based on Machine Learning and image processing," in *Computational Science – ICCS 2022, 22nd International Conference*, London (UK), Proceedings, Part III, *Lecture Notes in Computer Science*, vol. 13352, pp. 263–277, 2022, ISBN 978-3-031-08756-1, DOI: 10.1007/978-3-031-08757-8_23.

16. K. Struniawski, A. Konopka, and R. Kozera, "Identification of soil bacteria with Machine Learning and image processing techniques applying single cells' region isolation," in *Modelling and Simulation 2022. The European Simulation and Modelling Conference 2022*, Porto(Portugal), pp. 76–81, 2022, ISBN 9789492859242.

Currently, I am the author or co-author of three publications that are under peer review.

6.3.2 Bibliometric data

Publications: 16.
Research data: 1.
h-index (Scopus): 3.
Total IF: 15.6.
Total SNIP: 2.616.
Total Polish ministerial (MNiSW) score: 1,790.

6.3.3 Conference and seminar presentations

1. Exploring Apple Silicon's potential from simulation and optimization perspective, Computational Science – ICCS 2024, 24rd International Conference of Computational Science, July 1–4 2024, Malaga, Spain.

2. Metaheuristic Algorithms in Extreme Learning Machine for selection of parameters in activation function, Modelling and Simulation'2023. The 2023 European Simulation and Modelling Conference, 24-26 October 2023, Toulouse, France.

3. Performance evaluation of activation functions in Extreme Learning Machine, ESANN 2023 - European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 4-6 October 2023, Brugge, Belgium.

4. Performance of selected nature-inspired Metaheuristic Algorithms used for Extreme Learning Machine, Computational Science – ICCS 2023. 23rd International Conference, 3-5 July 2023, Prague, Czechia.

5. Klasyfikator Extreme Learning Machine – zastosowanie i optymalizacja, XI Konferencja Symbioza Techniki i Informatyki, 9-22 June 2023, Kiry, Poland.

6. Identification of soil bacteria with Machine Learning and image processing techniques applying single cells' region isolation. Modelling and Simulation 2022. The European Simulation and Modelling Conference 2022, 26-28 Oct 2022, Porto, Portugal.

7. Identyfikacja grzybów ryzosferowych przy użyciu metod sztucznej inteligencji, Seminar of the Institute of Horticulture in Skierniewice, 2 Nov 2021, Skierniewice, Poland.

8. Analiza obrazów mikroskopowych grzybów ryzosferowych, Seminar of the Department of Information Systems at the Institute of Information Technology of the Warsaw University of Life Sciences, 8 Sep 2021, Warsaw, Poland.

6.3.4 Reviews

- 13 reviews for Engineering Applications of Artificial Intelligence, Elsevier (ISSN: 1873-6769), 2024-2025.
- 2 reviews for Scientific Reports, Springer Nature (ISSN 2045-2322), 2024-2025.
- Plant Methods, Springer (ISSN: 1746-4811), 2025.
- International Journal of Computational Intelligence Systems, Springer (ISSN: 1875-6883), 2025.
- Cluster Computing, Springer (ISSN: 1573-7543), 2025.
- BMC Plant Biology, Springer (ISSN: 1471-2229), 2025.
- IEEE Access, IEEE (ISSN 2169-3536), 2024.

6.3.5 Research collaborations

- Institute of Horticulture, Skierniewice, Poland.
- Institute of Physics, University of Warsaw, Poland.
- Institute of Chemistry, University of Warsaw, Poland.
- Università della Tuscia, Viterbo, Italy.
- University of Pennsylvania, Philadelphia, USA.
- The University of Western Australia, Perth, Australia.

6.3.6 Popularization of science, contact with the media

I participated in the creation of several popular science journalistic texts promoting Institute of Information Technology, WULS-SGGW in the media and highlighting cooperation with the Institute of Physics of the University of Warsaw:

1. Article on sggw.edu.pl.
2. Article on mp.pl.
3. Article on pulsmedycyny.pl.
4. Article on rynekzdrowia.pl.
5. Article on uw.edu.pl.
6. Article on scienceinpoland.pl.
7. Article on gov.pl.
8. Article on forumakademickie.pl.
9. Article on medicalpress.pl.
10. Article on farmacjapraktyczna.pl.
11. Article on portalsamorzadowy.pl.
12. Article on gazetaolsztynska.pl.
13. Article on naszesprawy.eu.
14. Article on biotechnologia.pl.
15. Article on wnp.pl.
16. Article on suski.dlawas.info.
17. Article on dzienniknaukowy.pl.

6.3.7 Organizational and administrative activities

1. Associate Editor of Machine Graphics & Vision (ISSN: 1230-0535), since 2025.
2. Secretary of Machine Graphics & Vision (ISSN: 1230-0535), 2021-2024.
3. Program Committee member of Computational Science and AI for Addressing Complex and Dynamic Societal Challenges Equitably workshop at the 25th International Conference on Computational Science, Singapore, 2025.
4. Program Committee member of IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, Taiwan, 2025.
5. Member of the organizing committee of the conference "28th International Conference on Applications of Computer Algebra (ACA'2023)", July 17 - 21, 2023, Warsaw, Poland .

6. Supervisor of the Student Scientific Club at SGGW - “InIT”.

7. Secretary of the recruitment committee of the SGGW Doctoral School for the Institute of Information Technology, 2024.

6.3.8 Promoting science among students

Joint two scientific publications with SGGW Computer Science student - Rafał Wyszyński that based on publications received Polish National Science Centre Scholarship within the SONATA grant.

6.3.9 Honors and awards

- III degree award of SGGW’s Rector for scientific achievements (2024).
- Award for the best speaker during the XI Conference Symbiosis of Technology and Informatics - Kiry, Poland, 19-22 Jun 2023.