POLSKO-JAPOŃSKA
AKADEMIA TECHNIK
KOMPUTEROWYCH

Polsko-Japońska Akademia Technik
Komputerowych

Praca doktorska

Detekcja obiektów w obrazach cyfrowych z użyciem uczenia głębokiego w warunkach stratnej kompresji obrazu

Autor: mgr inż. Tomasz Gandor

Promotor: dr hab inż. Henryk Josiński

Promotor pomocniczy: dr hab inż. Jakub Nalepa

Bytom, 27 września 2022

Abstract

Thesis title: **Object detection in digital images using deep learning under lossy image compression**

Practical applications of object detection, such as video surveillance, are constrained by the need to transmit and store large image data. This forces the use of lossy compression methods, which allow for a strong reduction in the amount of processed data, but cause a deterioration in image quality.

The subject of this work is the assesment of influence, which this deterioration has on detection efficiency. The efficiency is understood both as sensitivity (detection of the highest possible number of correct objects) and precision (high percentage of correct detections and low percentage of false detections). Next, the goal of the research is to counteract the deterioration of object detection performance through appropriate model training. The results obtained have direct application in the video surveillance industry. The work is both theoretical and experimental.

The theoretical part presents, on the one hand, the basics of object detection and its underlying mechanisms, and, on the other hand, the JPEG lossy compression algorithm and the reasons why it causes a decrease in quality.

The experimental part was divided into two stages. The first stage includes an extensive study of object detection performance based on selected detection models, which were tested on a set of 5,000 test images. The images of the test set were compressed using the JPEG algorithm for different quality, controlled by the Q parameter of the algorithm, and the values of the object detection efficiency measures were determined. The results obtained confirm the research hypotheses. The different impact that compression quality deterioration has on detection sensitivity and precision was

characterised. In stage two, eight models trained by the dissertations's author were tested using four different approaches designed to improve detection performance for images subjected to heavy compression (Q parameter less than 30).

The results obtained confirmed the research hypotheses. The different effects that compression deterioration has on sensitivity and on detection precision and on measures of average precision (AP) were characterized. Performance improvements under strong compression are described, and the source code of the research software and the processed dataset have been made publicly available for download on the Internet.

Conclusions from the obtained results are supplemented with practical guidelines for designers of intelligent monitoring systems and suggestions for interesting directions of further research.

Spis treści

Spis rysunków

Spis tabel

Rozdział 1

Wprowadzenie

Cyfrowe obrazy i sekwencje wideo są obecne w stale rosnącej liczbie dziedzin działalności człowieka. Dotyczy to zastosowań amatorskich, profesjonalnych, artystycznych, administracyjnych oraz badawczych. Przyczynia się do tego zarówno powszechna dostępność urządzeń mobilnych wyposażonych w aparaty cyfrowe, jak i dostępność profesjonalnego sprzętu do akwizycji obrazu. Obrazy cyfrowe są transmitowane za pośrednictwem Internetu: za pomocą poczty elektronicznej, z użyciem przestrzeni składowania w chmurze oraz portali społecznościowych.

Dane obrazu przed kompresją, tzw. surowe, mają bardzo dużą objętość: obrazy pozyskiwane za pomocą standardowych aparatów cyfrowych mają rozmiary od kilku do kilkudziesięciu megapikseli (MPix), a w sekwencjach wideo każda klatka ma rozmiar od 0.3 do kilku megapikseli (dla klatki w rozdzielczości Full HD, tj. 1920×1080, to około dwa megapiksele), przy czym takich klatek na sekundę jest od kilkunastu do kilkudziesięciu, typowo 15–60. Dla obrazów i sekwencji kolorowych każdy piksel jest reprezentowany przez trzy liczby, których znaczenie jest zależne od zastosowanej przestrzeni barw, najczęściej są to składowe czerwona, zielona i niebieska, które tworzą przestrzeń RGB (ang. *red, green, blue*)[1]. Rozmiar każdej składowej zazwyczaj wynosi jeden bajt, kiedy stosowana jest tzw. 8-bitowa głębia koloru (w przetwarzaniu wideo wysokiej jakości stosowane są także głębie 10- i 12-bitowe, a w profesjonalnej fotografii także 16-bitowa, wówczas piksel zajmuje 6 bajtów, albo nawet 8 bajtów przy przestrzeniach barw z czterema składowymi). Przykładowe rozmiary obrazu i jednosekundowej sekwencji wideo są następujące:

- Obraz 12 MPix z aparatu cyfrowego: $12 \cdot 3 \cdot 10^6$ B = 36 MB, 36 takich zdjęć (odpowiednik kliszy analogowej) zajmuje 1.3 GB.

- Sekunda wideo Full HD 30 FPS: $1920 \cdot 1080 \cdot 3 \cdot 30 = 186\,624$ MB, co po przeliczeniu na przepływność w bitach na sekundę wynosi 1.493 Mbps, a zatem prawie półtora przepływności sieci Gigabit Ethernet.

[1] Możliwe są cztery składowe piksela, tzw. RGBA, tj. RGB + alfa. Dodatkowy kanał to *nieprzeźroczystość* (ang. *opacity*). Ma on znaczenie przy nakładaniu się (mieszaniu) obrazów, określanym jako *alpha blending*.

Te przykłady pokazują, że użycie kompresji obrazu jest niezbędne, szczególnie w przypadku wideo.

Postać cyfrowa obrazu wiąże się z możliwością przetwarzania obrazu za pomocą komputera, a nie tylko prezentacji go ludzkiemu odbiorcy. Można wtedy mówić o widzeniu maszynowym (ang. *machine vision*), nazywanym również „wizją maszynową" i „wizją komputerową" (ang. *computer vision*). W istocie chodzi o transformację informacji wizualnej na inną postać: symboliczną lub liczbową, co pozwala przetwarzać ją dalej, umieszczać w bazach danych, a nawet sterować procesami na jej podstawie, podobnie jak to jest możliwe z innymi informacjami w systemach komputerowych. Zakres zadań, którymi zajmuje się dziedzina widzenia maszynowego, obejmuje proste operacje, począwszy od progowania wartości pikseli, detekcji krawędzi i analizy konturów [82], aż do skomplikowanych zagadnień opisu obrazu w języku naturalnym lub rozpoznawania obrazów na podstawie ich opisu [44].

1.1 Detekcja obiektów na obrazach

Istotnym zadaniem widzenia maszynowego jest detekcja obiektów (ang. *object detection*). Polega ona na wskazaniu położenia wszystkich wystąpień określonej klasy (kategorii) obiektów widocznych na obrazie. Kategorii obiektów może być więcej, i wtedy należy wykryć te obiekty, podać ich położenia i wskazać, do której klasy należy każdy z nich. Liczba klas obiektów jest określona z góry i tylko obiekty tych klas powinny być wykrywane.

Detekcja obiektów jest niezwykle przydatna i jest stosowana w wielu dziedzinach:

- inteligentny monitoring wizyjny,
- obrazowanie medyczne,
- wyszukiwanie w zbiorach fotografii cyfrowych,
- przetwarzanie zdjęć satelitarnych,
- autonomiczne kierowanie pojazdami.

Wyliczenie to nie jest kompletne, a każda z tych dziedzin obejmuje rozliczne konkretne zastosowania, np. monitoring może dotyczyć przestrzeni miejskiej, upraw rolniczych, nieba (obiekty latające), infrastruktury transportowej (torowiska, kanały żeglugowe) oraz prywatnych posesji. Oprócz tego detekcja obiektów może być częścią składową bardziej skomplikowanych procesów, takich jak śledzenie obiektów (ang. *object tracking*), liczenie obiektów (np. pomiar intensywności ruchu drogowego lub pieszego), analiza grup obiektów (zbiorowiska ludzkie, bójki).

W połączeniu z komunikacją sieciową i rozproszonymi urządzeniami końcowymi – tj. tzw. Internetem rzeczy (ang. *Internet of Things*, IoT) – detekcja może być stosowana np. do efektywnego automatycznego sterowania oświetleniem, bramami i sygnalizacją świetlną w ruchu drogowym, co jest to korzystną alternatywą dla detektorów innego

rodzaju, np. pętli indukcyjnych, które wymagają ingerencji w infrastrukturę drogową. Gdy niemożliwe lub nieakceptowane jest autonomiczne podejmowanie decyzji przez systemy, istnieje możliwość współpracy z człowiekiem (np. operatorem monitoringu), który dokonuje ostatecznej oceny danej sytuacji. Detekcja obiektów może być użyta np. do wykrycia zatrzymania się osoby w nocy w niebezpiecznej okolicy i zasygnalizowania tego operatorom.

1.2 Uczenie głębokie

Głębokie uczenie maszynowe (ang. *deep learning*) [29] stanowi jedno z podejść w ramach badań nad sztuczną inteligencją (ang. *artificial intelligence*, AI). Zasadniczo uczenie głębokie jest specyficznym podzbiorem uczenia maszynowego (ang. *machine learning*, ML)[2].

Zastosowanie komputerów do rozwiązywania problemów obejmuje z jednej strony samodzielne programowanie algorytmów, a z drugiej wykorzystanie dostępnych algorytmów ogólnych (przykładami zadań, które można w ten sposób rozwiązywać, są optymalizacja i uczenie maszynowe). Przy programowaniu algorytmów to człowiek jest odpowiedzialny za podanie szczegółowej procedury uzyskania wyników na podstawie danych wejściowych. Zastosowanie algorytmów ogólnych wymaga innego rodzaju pracy – dopasowania zadania do metody, która je rozwiąże. W przypadku optymalizacji wystarczy określenie funkcji celu i dziedziny optymalizowanych parametrów oraz ograniczeń, które mogą tę dziedzinę (czasem w skomplikowany sposób) zawężać. W gestii programisty pozostaje także wybór odpowiedniej metody optymalizacji, być może jej konfiguracja, a resztę pracy wykonuje komputer, poszukując wartości parametrów tak, aby maksymalizować lub minimalizować funkcję celu. Uczenie maszynowe natomiast polega na rozwiązywaniu problemów bez tworzenia algorytmu, z użyciem modeli, które zostają wytrenowane w oparciu o dane przykładowe (tzw. zbiór uczący). Dla różnych wartości danych wejściowych i odpowiadających im oczekiwanych wyników uzyskujemy model, który będzie rozwiązywał ten sam problem również dla nowych danych, spoza zbioru uczącego.

Uczenie maszynowe opiera się na gotowych schematach modeli, np.: drzewo decyzyjne, maszyny wektorów nośnych (ang. *support vector machines*, SVM), sztuczne sieci neuronowe (ang. *artificial neural networks*, ANN). Modele te mają możliwość parametryzacji – wybór zmiennych i wartości progowych w drzewie, parametry wielomianu w SVM lub regresji liniowej, wagi połączeń w sieci neuronowej. Dopasowanie (nauczenie, wytrenowanie) modelu polega na optymalizacji tych parametrów, tak aby wyniki zwracane przez model odpowiadały tym w zbiorze uczącym.

Uczenie głębokie wyróżnia się wśród pozostałych metod uczenia maszynowego tym, że nie wymaga ręcznej transformacji danych wejściowych. Jest to określane mianem samodzielnego uczenia reprezentacji [29]. Dla skomplikowanych zadań, takich jak detekcja obiektów, potrzebne były w tradycyjnych podejściach ML dodatkowe przekształcenia:

[2]Dosłownie to pojęcie oznacza „uczenie się maszyn".

wyliczanie deskryptorów cech, selekcja tych cech pod kątem przydatności i ostatecznie uczenie modelu na tych ręcznie zbudowanych reprezentacjach. Natomiast uczenie głębokie otrzymuje na wejściu dane nieprzetworzone (np. wartości pikseli obrazu), a następnie automatycznie dobiera sposób ich reprezentacji (w procesie optymalizacji parametrów modelu), bez konieczności ręcznej inżynierii cech.

Do realizacji praktycznej uczenia głębokiego stosowane są sztuczne sieci neuronowe, w szczególności sieci konwolucyjne albo inaczej splotowe (ang. *convolutional neural networks*, CNN). Sieci te składają się ze sztucznych neuronów, które są zorganizowane w warstwy (ang. *layers*). Pierwsza warstwa, która przyjmuje dane wejściowe, to warstwa wejściowa, a ostatnia, która zwraca wyniki, to warstwa wyjściowa; warstwy, które znajdują się pomiędzy nimi, to tzw. warstwy ukryte (ang. *hidden layers*). Liczba tych warstw to głębokość sieci, i stąd potoczne rozumienie uczenia głębokiego jako uczenia maszynowego z sieciami neuronowymi o dużej liczbie warstw. Bardziej istotne jest jednak to, że wyjścia niższych warstw stanowią reprezentację, na której operują warstwy wyższe. I tak dla obrazów, wyjścia neuronów pierwszej warstwy mogą opisywać gradient jasności w otoczeniu piksela, następne warstwy budują z tej informacji krawędzie, układy krawędzi i coraz bardziej skomplikowane pojęcia, które ostatecznie reprezentują obecność lub brak obecności wykrywanej klasy obiektów, np. ludzi.

Uczenie głębokie przyczyniło się do wielu przełomów w dziedzinie widzenia maszynowego, dotyczących: klasyfikacji obrazów [49], detekcji obiektów [72–74], generacji i rekonstrukcji obrazów [63, 107], klasyfikacji sekwencji wideo [45], detekcji anomalii w sekwencjach wideo [85]. W niniejszej pracy szczególną uwagę poświęcono detekcji obiektów, która dzięki uczeniu głębokiemu uzyskała wysoką skuteczność oraz jednolite podejście do tworzenia modeli i wykrywania różnych klas obiektów, nawet bardzo wielu jednocześnie [73].

Głębokie modele detekcji obiektów składają się z sieci CNN, która realizuje ekstrakcję cech w obrazie (taka sieć jest określana jako kręgosłup detektora), a następnie ze struktury sztucznych neuronów (podsieci) określanej jako głowica detektora. Głowica może również być siecią konwolucyjną, albo może składać się z warstw w pełni połączonych (ang. *fully-connected*).

Można wyróżnić dwa podejścia do detekcji obiektów w modelach głębokich: jednoetapowe, które jest również nazywane gęstym, oraz dwuetapowe, które jest określane jako rzadkie. Podejście jednoetapowe polega na detekcji w jednym przebiegu, korzystając z informacji zawartej w każdym punkcie (tj. w sposób gęsty) dwuwymiarowego wyniku przekształcenia obrazu za pomocą sieci CNN. Do tej grupy zaliczamy takie modele jak YOLO [7, 72–74], SSD [61] i RetinaNet [58]. W podejściu dwuetapowym najpierw wykonywany jest etap pierwszy – generowanie propozycji (kandydatów) obiektów. Aktualnie ten etap jest oparty na głębokich sieciach CNN, ale we wczesnych modelach dwuetapowych wykorzystywane w nim były proste metody widzenia maszynowego. W etapie drugim propozycje obiektów są, z użyciem sieci CNN, klasyfikowane jako odrzucone lub przyjęte. W drugim przypadku będzie dla nich wyznaczone dokładne położenie i przypisanie do klasy, co składa się na pojedynczy wynik detekcji obiektów. Ponieważ drugi etap dotyczy tylko miejsc wskazanych przez etap pierwszy, jest to detekcja rzadka (tj. selektywna). Przykłady modeli dwuetapowych to R-CNN [27], Fast

R-CNN [26], Faster R-CNN [76], Mask R-CNN [34] i RetinaMask [24].

Istotną wartością dla każdego wykrytego obiektu jest ufność detekcji (ang. *confidence/confidence score*). Jest to teoretyczna (tj. wskazywana przez model) miara wyrażająca prawdopodobieństwo poprawności pojedynczego obiektu wynikowego. Angielska nazwa oznacza „pewność" lub „przekonanie" i tłumaczy w ten sposób znaczenie ufności jako liczby wyrażającej, jak silnie model jest „przekonany" o obecności obiektu w danym miejscu obrazu. Standardowo ufność jest liczbą z przedziału $[0, 1]$.

1.3 Miary skuteczności detekcji obiektów

Jakość wyników uzyskanych z użyciem modelu detekcji obiektów można ogólnie określać jako *skuteczność* detekcji. W idealnym przypadku algorytm zwróci wszystkie widoczne obiekty i nie zwróci żadnych nieprawidłowych (fałszywe detekcje), obiekty będą przypisane do właściwej klasy, a współrzędne prostokątów zawierających (ang. *bounding box*) będą poprawne. Do pomiaru skuteczności detekcji obiektów potrzebny jest zbiór testowy, na który składają się obrazy i metadane dotyczące położenia obiektów. Wówczas można powiedzieć, że dany detektor uzyskuje określone wartości miar skuteczności na konkretnym zbiorze testowym, takim jak np. COCO `val2017` [59].

Podstawowymi miarami skuteczności, na których bazują wszystkie bardziej złożone miary, są liczby bezwzględne: poprawnie wykrytych obiektów (ang. *true positive*, TP), fałszywych detekcji (ang. *false positive*, FP) oraz pominiętych (niewykrytych) obiektów zbioru testowego (ang. *false negative*, FN). W odróżnieniu od oceny klasyfikacji binarnej, nie da się określić wartości *true negative* – liczba możliwych błędnych obiektów jest olbrzymia i nie ma sensu rozpatrywanie problemu, ile spośród nich zostało „poprawnie" odrzucone.

Na podstawie TP, FP i FN można określić dwie kluczowe miary skuteczności, tj. precyzję (ang. *precision/positive predictive value*, PPV) i czułość (ang. *sensitivity/recall/true positive rate*, TPR). Precyzja jest to odsetek poprawnych detekcji wśród wszystkich wyników modelu:

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \tag{1.1}$$

a czułość to odsetek obiektów, które wykrył detektor wśród wszystkich obiektów obecnych w zbiorze testowym:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{1.2}$$

Oficjalna miara stosowana w konkursie detekcji obiektów opartym na zbiorze COCO Detection Challenge nazywa się AP (ang. *average precision*), czyli miara średniej precyzji. Opiera się ona na uśrednieniu wartości precyzji, którą zachowuje model przy osiągnięciu określonego progu czułości. W tym celu należy rozpatrywać obiekty w kolejności malejącej ufności detekcji, która jest określana przez model dla każdego wykrytego obiektu. Dołączając kolejne obiekty wyznacza się TP i FP, a kiedy wartość TPR (tj. iloraz TP przez sumę TP + FN, czyli liczbę obiektów do wykrycia) przekroczy kolejny próg czułości (dla AP są to kolejne procenty liczby obiektów do wykrycia

w zbiorze), wtedy wylicza się dla tego progu wartość PPV i agreguje ją do średniej arytmetycznej (stąd „średnia precyzja" – średnia PPV). Wartości PPV dla tych progów TPR, które nie zostały przekroczone, zostaną przy wyliczaniu średniej zastąpione zerami.

Jest to analogiczna zasada do tej stosowanej w mierze mAP (ang. *mean average precision*), którą zdefiniowano dla zbioru PASCAL VOC [20]. Różnice polegają na większej liczbie progów TPR (100 dla miary AP i 10 dla miary mAP), oraz na tym, że wyliczając wartość AP uśrednia się dodatkowo wynik dla różnej dokładności wskazania lokalizacji obiektu.

Implementacja algorytmu wyliczania miary AP w języku Python została przez twórców zbioru COCO udostępniona w bibliotece COCO API i opublikowana w Internecie[3]. Pozwala to wszystkim na samodzielne wyznaczenie wartości tej miary na podstawie pliku w formacie JSON (ang. *JavaScript Object Notation*), opisującego obiekty zwrócone przez model.

1.4 Kompresja obrazu

Duży rozmiar danych obrazu i wideo spowodował konieczność użycia kompresji do transferu i zapisu fotografii cyfrowych. Kompresja pozwala reprezentować obraz z użyciem mniejszej ilości danych, a iloraz rozmiaru oryginalnego i skompresowanego to *współczynnik kompresji*. Metody kompresji można podzielić na bezstratne (ang. *lossless*) i stratne (ang. *lossy*).

Kompresja bezstratna to odwracalne przekształcenie danych do postaci, w której zajmują one mniej pamięci od oryginalnych. Można do niej wykorzystywać algorytmy kompresji ogólnego przeznaczenia, takie jak LZ-77 [110] (algorytm DEFLATE obecny w formatach zip, gzip), LZMA, Zstd, Brotli. Wadą używania generycznych algorytmów jest brak standardowego wsparcia ze strony oprogramowania: obrazy zapisane w pliku archiwum przed dalszym przetwarzaniem muszą zostać wyodrębnione. Istnieją też formaty plików graficznych, które umożliwiają bezstratną kompresję, są to PNG (wyłącznie bezstratny) i JPEG-2000 (posiada zarówno tryb bezstratny, jak i stratny). Wadą kompresji bezstratnej jest duży rozmiar danych wyjściowych, a często także wysoka złożoność obliczeniowa (długi czas działania). Za to jej zaletą jest zachowanie całej informacji obecnej w obrazie i dlatego jest ona wymagana w dziedzinach, w których nie można sobie pozwolić na utratę informacji: obrazowanie medyczne, astronomia, digitalizacja ważnych dokumentów i dzieł sztuki.

Kompresja stratna jest przekształceniem jednokierunkowym, które jest odwracalne tylko w przybliżeniu. Istnieje wiele operacji, które mają taką własność: w pewnym sensie skalowanie obrazu w dół (ang. *downsampling*) z towarzyszącą operacją odwrotną skalowania w górę (ang. *upsampling*) z interpolacją można traktować jako przykład kompresji stratnej. Podobnie, zmniejszenie palety kolorów albo liczby poziomów jasności pikseli (kwantyzacja) również prowadzi do mniejszego rozmiaru danych. Takie

[3] `https://github.com/cocodataset/cocoapi` (dostęp: 27 września 2022)

operacje jednak są zauważalne przez odbiorcę zdjęcia, co ogranicza ich przydatność[4].

Rozwiązaniem, które znacząco poprawia współczynnik kompresji obrazu przy zachowaniu jego dobrej jakości, jest poddanie danych obrazu transformacji przed usunięciem informacji. W algorytmie kompresji JPEG [98] stosuje się w tym celu dyskretną transformację cosinusową (ang. *discrete cosine transform*, DCT) [1]. Obraz jest dzielony na bloki 8×8 pikseli, każdy blok jest poddawany DCT, co daje w wyniku bloki 8×8 współczynników DCT (ang. *DCT coefficients*). Współczynniki są dzielone z zaokrągleniem do liczby całkowitej przez odpowiadające im dzielniki z tablicy kwantyzacji (ang. *quantization table*, QT), która również ma rozmiar 8×8. Małe współczynniki DCT zostają w ten sposób wyzerowane. Zaokrąglone wyniki dzielenia są następnie kodowane i zapisywane jako rezultat kompresji. W procesie dekompresji, zdekodowane liczby są mnożone przez te same elementy tablicy QT dając w wyniku przybliżone współczynniki DCT. Następnie odwrotna transformacja tych współczynników zwraca przybliżenie oryginalnego obrazu.

Wartości dzielników w tablicy QT są ustalane m.in. przez wybór wartości parametru Q, który jest wartością całkowitą z zakresu $[1, 100]$. Bardziej stratna kompresja jest uzyskiwana przy niższych wartościach parametru Q, które wiążą się z większymi dzielnikami w tablicy QT. Wówczas pojawiają się tzw. artefakty kompresji (ang. *compression artifacts*), które przejawiają się widocznymi blokami (utrata ciągłości przejść tonalnych, „kafelki") oraz zakłóceniami barw i drobnych szczegółów na obrazie.

Pogorszenie jakości obrazu może być mierzone w sposób ilościowy, z użyciem miar jakości obrazu. Miary takie nazywamy referencyjnymi, kiedy do ich obliczenia potrzebny jest obraz oryginalny (referencyjny), a bezreferencyjnymi, kiedy jedyną daną wejściową jest badany obraz, bez potrzeby dostępu do oryginalnego obrazu.

Dobór tablic kwantyzacji w celu poprawy percepcji jakości obrazu przez ludzkiego odbiorcę był tematem prac badawczych [4,18,19]. Wpływ kompresji na modele uczenia głębokiego pozostaje problemem, który zasługuje na bardziej szczegółowe zbadanie.

1.5 Cel, wkład i teza rozprawy

Celem niniejszej pracy jest zbadanie wpływu stratnej kompresji na skuteczność detekcji obiektów na obrazach, realizowanej za pomocą modeli opartych o uczenie maszynowe głębokie. Wpływ ten zostanie opisany w aspektach jakościowym oraz ilościowym. Opracowanie teoretyczne i eksperymentalne będą nakierowane na potwierdzenie następujących tez:

Teza 1 Kompresja JPEG nie powoduje spadku precyzji detekcji dla parametru Q z przedziału $[30, 100]$. Przyczyną spadku skuteczności detekcji jest w tym przypadku pogorszenie czułości detekcji.

[4]Za wyjątkiem zmiany rozmiaru obrazów, które będą prezentowane na urządzeniach o niskiej rozdzielczości, jak np. monitory komputerowe z kilkudziesięcioma punktami na cal (ang. *dots per inch*, DPI). Aczkolwiek, nowsze urządzenia mobilne mają nawet dwukrotnie większe rozdzielczości i wymagają grafiki o większym rozmiarze w celu uzyskania dobrej jakości.

Teza 2 Spadek skuteczności detekcji obiektów dla głębokich modeli, mierzony za pomocą miar AP, jest podobny zarówno w modelach jednoetapowych, jak i dwuetapowych, dla parametru Q z przedziału $[30, 100]$.

Teza 3 Trening detektorów obiektów z użyciem obrazów silnie (Q=20) lub średnio (Q=40) skompresowanych poprawia skuteczność detekcji na obrazach poddanych silnej kompresji, dla parametru Q z przedziału $[5, 30)$. Źródłem poprawy jest zwiększenie czułości detekcji dla tych obrazów.

Wkład własny autora pracy polega na zaproponowaniu procedury badania wpływu kompresji na skuteczność detekcji obiektów oraz na wykonaniu szeroko zakrojonych badań na reprezentatywnej grupie detektorów obiektów i obszernym zbiorze danych testowych, w celu dokładnego scharakteryzowania tego wpływu, jak również na wskazaniu metod jego ograniczania. Oprócz powszechnie używanych miar skuteczności (rodzina miar AP), zostały zastosowane inne miary, co pozwala zaobserwować interesujące różnice w zachowaniu różnych modeli.

Celowość podjęcia tematu wynika z dwóch przesłanek: praktycznej i teoretycznej. Nowa wiedza zdobyta w toku badań ma istotne zastosowanie praktyczne – specjaliści pracujący z systemami, w których przetwarzane są duże liczby obrazów i konieczna jest ich stratna kompresja, otrzymują dzięki uzyskanym wynikom wskazówki, które pozwolą dobrać parametry kompresji w sposób optymalny z punktu widzenia skuteczności działania systemu, oszczędności przestrzeni dyskowej i efektywnego wykorzystania energii.

Przykładem takiego systemu jest CityEye firmy KP Labs sp. z o.o., przy którego opracowaniu zaangażowany był autor tej pracy. Jest to system monitoringu miejskiego, w którym detekcja obiektów wykonywana jest na poszczególnych klatkach, które są skompresowane algorytmem JPEG. Do detekcji można użyć zarówno publicznie dostępnych modeli pre-trenowanych, jak i dedykowanych modeli wytrenowanych przez pracowników KP Labs, a także, po odpowiedniej konfiguracji, modeli wytrenowanych przez użytkownika końcowego. System CityEye jest współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Rozwoju Regionalnego w ramach Regionalnego Programu Operacyjnego Województwa Śląskiego na lata 2014-2020, grant UDA-RPSL.01.02.00-24-0660/16-00.

W aspekcie teoretycznym wartościowe jest poszerzenie wiedzy na temat zachowania głębokich modeli detekcji obiektów w szczególnych warunkach, jakie stwarza stratna kompresja obrazu. Następnie, celowe jest zbadanie możliwości przeciwdziałania pogorszeniu skuteczności detekcji – w przypadku tej pracy proponowaną metodą jest modyfikacja zbioru obrazów wykorzystywanego przy treningu modelu. Wytworzona wiedza będzie przydatna dla twórców i użytkowników końcowych systemu CityEye, a potencjalnie także w wielu innych zastosowaniach detekcji obiektów.

1.6 Struktura pracy

Rozdział 2 zawiera przegląd uprzednich prac związanych z tematyką badań w dostępnej literaturze. Prace te dotyczą jakości obrazu, kompresji obrazów, skuteczności i niezawodności modeli uczenia maszynowego.

W rozdziale 3 omówiono zagadnienia związane z uczeniem maszynowym i opisano metody uczenia głębokiego, a także relacje między pojęciami: sztucznej inteligencji, uczenia maszynowego i uczenia głębokiego. Przedstawiona została taksonomia zadań uczenia maszynowego i jej związki ze skalami pomiarowymi zmiennych objaśniających i objaśnianych. Wyjaśniono pojęcia uczenia cech i transferu uczenia.

W rozdziale 4 opisano detekcję obiektów jako zadanie widzenia maszynowego. Pokazano związki detekcji obiektów z innymi zadaniami widzenia maszynowego oraz jej przynależność do kategorii zadań uczenia maszynowego – jako połączenia regresji (w odniesieniu do lokalizacji obiektu) i klasyfikacji (wskazanie kategorii wykrytego obiektu). Przedstawiono wcześniejsze modele widzenia maszynowego stosowane do detekcji obiektów, a następnie szczegółowo opisano rozwój detektorów opartych na głębokich sieciach konwolucyjnych. Omówiono podział modeli detekcji na jedno- i dwuetapowe, a także scharakteryzowano zasady działania obu typów modeli, z wyjaśnieniem poszczególnych etapów. Przedstawiono trudności związane z trenowaniem modeli detekcji obiektów i prace omawiające metody przezwyciężania tych trudności.

W rozdziale 5 przedstawiono metody kompresji obrazów. Omówiono podział algorytmów na bezstratne i stratne oraz szczegółowo opisano stratną metodę kompresji JPEG. Wyjaśniono zasadę działania tego algorytmu i wskazano przyczyny utraty informacji w procesie kompresji. Przedstawiono sposób sterowania jakością za pomocą parametru Q i jego związek z kwantyzacją współczynników dyskretnej transformacji cosinusowej. Następnie przedstawiono miary jakości obrazu, a także podziały tych miar – na subiektywne i obiektywne, oraz na referencyjne i bezreferencyjne. Opisano najczęściej stosowane miary obiektywne referencyjne, w tym zastosowaną w części empirycznej miarę SSIM, czyli tzw. indeks podobieństwa strukturalnego.

W rozdziale 6 znajduje się opis badań empirycznych dotyczących negatywnego wpływu stratnej kompresji obrazów na skuteczność detekcji obiektów oraz modyfikacji treningu modeli w celu przeciwdziałania pogorszeniu tej skuteczności. Badania były podzielone na dwa etapy: etap I dotyczył pre-trenowanych modeli detekcji dostępnych w Internecie, zaś etap II – modeli wytrenowanych przez autora pracy. Dla każdego z etapów opisano zestaw eksperymentalny, przedstawiono procedurę badawczą i zaprezentowano uzyskane wyniki.

Rozdział 7 zawiera omówienie otrzymanych wyników i ich interpretację, a także wnioski i potencjalne kierunki dalszych badań na podstawie tych wyników. Przytoczone są również odniesienia do publikacji, które mogą być przydatne do poszerzenia zakresu badań opisanych w tej pracy.

W rozdziale 8 dokonano podsumowania rozprawy, wskazano na potwierdzenie jej tez i przedstawiono uwagi końcowe.

Załącznik A zawiera wykaz i rozwinięcia skrótowców używanych w tekście pracy, a także listę symboli występujących we wzorach matematycznych. W załączniku B

zamieszczono wyniki miar skuteczności detekcji uzyskane dla modeli pre-trenowanych. Załącznik C zawiera te same miary skuteczności dla modeli trenowanych w toku badań empirycznych. W załączniku D zamieszczono kody źródłowe programów służących do wykonania detekcji obiektów na zbiorze testowym oraz do wyznaczania miar uzyskanej skuteczności tej detekcji.

Rozdział 2

Przegląd literatury

W niniejszym rozdziale przedstawiono przegląd dotychczasowych prac związanych z tematem rozprawy, zarówno o charakterze badawczym, jak i teoretycznym. Stanowiły one podstawę do badań przeprowadzonych w ramach tej pracy. Dotyczą one wpływu jakości na skuteczność sieci konwolucyjnych (na przykładzie klasyfikacji) oraz ogólnej niezawodności sieci konwolucyjnych. Poszerzenie przeglądu literatury stanowią odniesienia zawarte w dalszych rozdziałach na temat uczenia głębokiego, detekcji obiektów i kompresji obrazów.

2.1 Jakość obrazu a klasyfikacja

Większość prac z dziedziny detekcji obiektów nie zajmuje się jakością obrazu wejściowego, ani nie czyni żadnych jawnych założeń na jej temat. Problematyce wpływu stratnej kompresji na skuteczność detekcji również nie poświęcono w literaturze dostatecznej uwagi, niemniej jednak można wskazać pewną grupę prac, które są z tym zagadnieniem tematycznie powiązane. Prace te dotyczą odporności (ang. *robustness*) modeli głębokich na zmiany jakości obrazu, w tym również na pogorszenie jakości wywołane metodami kompresji stratnej. Dziedziny, z którymi wiąże się ten problem – kompresja obrazu, mierzenie jakości (lub podobieństwa) obrazów, uczenie głębokie, wizja komputerowa i detekcja obiektów – są aktywnymi obszarami badań, z dużą liczbą publikacji.

W artykule [111] Zou i Pong opracowali metody rozpoznawania twarzy w niskich rozdzielczościach, nawet dla kwadratów o wymiarach 10×10 pikseli. Obniżanie rozdzielczości można traktować jako najprostszą metodę stratnego zmniejszania rozmiaru danych obrazu. W pracy podkreślono znaczenie jakości obrazu w zastosowaniach monitoringu wizyjnego. Zastosowana metoda korzysta z metod podnoszenia rozdzielczości (ang. *super resolution*), które aktualnie również są realizowane z użyciem modeli głębokich w podobnym celu, np. Bai i in. [2] – autorzy korzystają z sieci GAN (ang. *generative adversarial network*) w celu poprawy detekcji małych obiektów. Małe obiekty to stale aktualny problem, ponieważ, mimo zwiększania rozdzielczości sensorów optycznych, pojawia się możliwość i potrzeba obserwowania obiektów coraz bardziej oddalonych od

kamery (zastosowania w monitoringu wizyjnym, obrazowaniu satelitarnym oraz astronomii). Do rozwiązywania tego samego problemu – rozpoznawania twarzy przy niskiej rozdzielczości – Ren i in. [75] proponują odmienną metodę ręcznej ekstrakcji cech, która pozwoliła uzyskać dobre rezultaty bez konieczności uciekania się do przekształcenia *super resolution*.

Karam i Zhu [43] opublikowali zbiór obrazów QLFW (ang. *Quality labeled faces in the wild*), który jest oparty na zbiorze LFW [41] (ang. *Labeled faces in the wild*). Korzystając z tego popularnego zbioru do wykrywania i rozpoznawania (reidentyfikacji) twarzy, autorzy stworzyli zbiór obrazów zdegradowanych (ang. *degraded*) jakościowo, który zaproponowali do badania skuteczności algorytmów odpornych na zakłócenia jakości. Wśród użytych zakłóceń były m.in. rozmycie Gaussa, biały szum, obniżenie kontrastu oraz kompresja stratna obrazu. Autorzy określili również kryteria oceny dla tego zadania, tworząc tym samym „wyzwanie" (ang. *challenge*) dla kolejnych prac badających ten problem. Artykuł towarzyszący zbiorowi nie prezentuje jednak żadnych wyników początkowych (ang. *baseline*), które umożliwiałyby wykonanie porównań. Metoda, która uzyskuje dobre wyniki na zbiorze QLFW, została przedstawiona przez Tao i in. [89]. Polega ona na ręcznej ekstrakcji cech obrazu określanej jako ARJKSR (ang. *multi-source Adaptation Regularization Joint Kernel Sparse Representation*).

W ramach badań opisanych w pracy Basu i in. [3] stworzony został zbiór n-MNIST, służący do rozpoznawania cyfr, na podstawie zbioru MNIST[1]. Zbiór MNIST zaś został stworzony przez Lecuna na podstawie obrazów cyfr zebranych przez Narodowy Instytut Standaryzacji i Technologii (ang. *National Institute of Standards and Technology*, NIST) i zastosowany w pracy [55], prekursorskiej publikacji dla uczenia głębokiego, w której zaprezentowano rodzinę głębokich konwolucyjnych klasyfikatorów obrazu LeNet (LeNet-1, LeNet-4 i LeNet-5 – to przykłady konwencji nazewniczej stosowanej w wielu późniejszych modelach, gdzie sufiks numeryczny oznacza głębokość sieci – liczbę warstw). Zbiór n-MNIST zawiera obrazy ze zbioru MNIST, poddane pogorszeniu jakości za pomocą trzech metod:

- addytywnego szumu Gaussa,
- rozmycia ruchomego (ang. *motion blur*),
- połączenia szumu i obniżenia kontrastu.

Sposób obniżenia kontrastu nie jest opisany w artykule, ale na stronie internetowej poświęconej zbiorowi danych n-MNIST znajdują się informacje o szczegółach implementacji, w tym stosunek sygnału do szumu dla zaszumienia, kąt i odległość przesunięcia w rozmyciu ruchomym oraz metoda redukcji kontrastu – jest to przeskalowanie zakresu pikseli w dół o połowę. Obrazy i metadane zbioru są dostępne do pobranie w formacie MATLAB[2].

Obok opublikowanego zbioru obrazów autorzy prezentują swoją metodę opartą na sieciach DBN (ang. *deep belief network*) [5, 38]. Są to sieci głębokie, ale trenowane odmiennie od modeli konwolucyjnych – każda warstwa jest trenowana oddzielnie, a proces

[1] `http://yann.lecun.com/exdb/mnist/` (dostęp: 27 września 2022)
[2] `http://csc.lsu.edu/~saikat/n-mnist/` (dostęp: 27 września 2022)

treningu realizowany jest nie za pomocą wstecznej propagacji błędu, lecz z użyciem techniki optymalizacji RBM (ang. *restricted Boltzmann machines*). Autorzy porównują bezpośrednio zastosowaną sieć DBN z siecią DBN korzystającą z ekstrakcji cech z użyciem probabilistycznych drzew czwórkowych (ang. *probabilistic quadtrees*), które są metodą rekursywnego podziału przestrzeni. Obraz jest zamieniony w graf (drzewo), którego każdy poziom opisuje podział (fragmentu) obrazu na cztery ćwiartki. Wektor cech, który będzie przetworzony przez sieć DBN, powstaje z „odczytania" cech z grafu przez przeszukiwanie w głąb (ang. *depth first search*, DFS). Zarówno dla zbioru MNIST, jak i zdegradowanego jakościowo n-MNIST, lepsze wyniki klasyfikacji cyfr uzyskano z użyciem probabilistycznych drzew czwórkowych.

Ulmann i in. [93] rozpatruje minimalne warunki dla detekcji lub klasyfikacji obrazów przez modele głębokie. Minimalny fragment obrazu, przy którym możliwe jest rozpoznanie zawartości, został nazwany „atomem rozpoznania" (ang. *atom of recognition*) lub MIRC (ang. *minimal recognizable configuration*). Badanie obejmowało zaangażowanie ludzi z wykorzystaniem platformy Amazon Mechanical Turk – około 14 tys. osób oceniało 3553 wycinki wyodrębnione z 10 obrazów (bez powtórzenia któregokolwiek obrazu dla pojedynczego uczestnika). Oczekiwaną odpowiedzią było określenie, co przedstawia obraz lub stwierdzenie niemożności identyfikacji. Fragmenty były również poddawane zmniejszeniu rozdzielczości, tj. skalowane w dół i następnie w górę z użyciem interpolacji (efekt wizualny w tym przypadku jest podobny do rozmycia – następuje utrata drobnych szczegółów). Jeżeli dla danego fragmentu, po kolejnym przycięciu lub zmniejszeniu rozdzielczości, następował spadek poprawnych odpowiedzi poniżej 50%, klasyfikowano go jako MIRC.

Następnie zbiór fragmentów typu MIRC podzielono na podzbiory uczący i testowy, wytrenowano modele i przetworzono za pomocą klasyfikatora VGG-16 [83] i detektora obiektów R-CNN [27], z zastosowaniem niezbędnych do tego operacji (np. skalowanie MIRC w górę do rozmiaru wejścia sieci). Oprócz tych modeli głębokich zbadane zostały modele starszego typu oparte na połączeniu wizji komputerowej i uczenia głębokiego, takie jak histogram zorientowanych gradientów (ang. *histogram of oriented gradients*, HoG) [14] i oparty na HoG deformowalny model wieloczęściowy (ang. *deformable part-based model, deformable parts model*, DPM) [21]. Wyniki wskazywały na wyższą skuteczność klasyfikacji obiektów przez ludzi niż ta, którą uzyskano dla badanych modeli. Stąd, jako miarę błędu modelu zaproponowano *lukę rozpoznania* (ang. *recognition gap*), która dotyczy dokładności klasyfikacji (ang. *accuracy*). Jest to różnica średniej dokładności uzyskiwanej na zbiorze testowym przez ludzi i dokładności osiąganej przez model[3].

W pracy Dodge'a i Karam [16] znajdują się wyniki badań dotyczących wpływu jakości obrazu na skuteczność klasyfikacji obrazów. Klasyfikacja to problem blisko związany z detekcją obiektów, który w pewnym sensie jest jej szczególnym przypadkiem, w którym nie jest wymagane określenie lokalizacji obiektu. Autorzy proponują pięć metod pogarszania jakości obrazu, z których dwie ostatnie polegają na stratnej kompresji obrazu (podobnie jak w [43]):

[3]Dla modeli o „nadludzkiej" dokładności byłaby to wartość ujemna.

- addytywny szum Gaussa – wartości pikseli zostały zmodyfikowane o losową wartość z rozkładu Gaussa o średniej zero. Parametrem pogorszenia jakości było odchylenie standardowe σ rozkładu, które zostało przebadane w zakresie od 10 do 100 z krokiem 10.

- rozmycie Gaussa – splot obrazu z jądrem przybliżającym funkcję gęstości dwuwymiarowego rozkładu Gaussa. Odchylenie standardowe σ rozkładu badane było w zakresie od 1 do 9 z krokiem 1, a rozmiar okna, w którym była obliczana wartość splotu dla każdego piksela, wynosił czterokrotność odchylenia standardowego.

- pogorszenie kontrastu – zrealizowane poprzez mieszanie (ang. *blending*) obrazu wejściowego z jednolitym obrazem szarym (jednorodny obraz o wartościach pikseli równych połowie zakresu, tj. 128). Wartości wynikowe były średnią ważoną oryginalnej wartości piksela i 128 z wagami, odpowiednio, α i $1 - \alpha$. Parametr α, czyli współczynnik mieszania (ang. *blending factor*), przyjmował wartości od 0 do 1 z krokiem 0.1[4].

- kompresja JPEG – obrazy wejściowe zostały skompresowane algorytmem JPEG z zadaną wartością parametru Q z przedziału od 2 do 20 z krokiem 2.

- kompresja JPEG2000 – obrazy wejściowe zostały skompresowane algorytmem JPEG2000. Parametry kompresji ustawiane były za pomocą pożądanej wartości minimalnej dla wskaźnika jakości obrazu PSNR, który był badany dla przedziału od 20 do 40 z krokiem 2.

W eksperymentach wykorzystywano 1000-klasowy zbiór danych ImageNet, a dokładniej podzbiór 10 tys. obrazów z walidacyjnej części zbioru obrazów konkursu klasyfikacji obrazów ILSVRC (ang. *ImageNet Large Scale Visual Recognition Challenge*), edycja 2012 [79]. Obrazy zostały wylosowane spośród 50 tys. obrazów walidacyjnych w taki sposób, aby dla każdej z tysiąca kategorii było dokładnie dziesięć wystąpień. W procedurze badawczej obrazy były poddawane degradacji jakości z użyciem wymienionych wcześniej metod, a następnie klasyfikowane za pomocą czterech różnych głębokich sieci neuronowych wytrenowanych na zbiorze ILSVRC 2012 – części treningowej (1 281 167 obrazów):

- AlexNet [49] – klasyfikator z 5 warstwami splotowymi i 3 gęstymi.

- VGG-CNN-S [9] – podobny klasyfikator (5 warstw splotowych i 3 w pełni połączone), podwyższone parametry (większe pole receptywne).

- VGG-16 [83] – znany i popularny model z 13 warstwami splotowymi i 3 gęstymi.

[4]Wartość 0 również została uwzględniona, mimo iż oznacza proste zastąpienie obrazu pustym jednorodnym obszarem – uzyskano dla niej zerowe wyniki skuteczności klasyfikacji.

- GoogLeNet [87] – klasyfikator korzystający z warstw *incepcji*[5], agregacji wartości średniej i pojedynczej warstwy klasyfikacji.

Do oceny skuteczności klasyfikacji zastosowano dokładność *top-1* oraz *top-5*. Są to bardzo proste miary – klasyfikator zwraca rozkład przewidywanych prawdopodobieństw (ang. *confidence score*) dla każdej z tysiąca klas (ufność danej klasy). Dokładność *top-1* mierzy odsetek klasyfikacji, w których najwyższą ufność ma wzorcowa klasa dla danego obrazu, a dokładność *top-5* to odsetek klasyfikacji, w których klasa wzorcowa jest w pierwszej piątce klas o najwyższej ufności. Oznaczając te miary przez $Acc_{\text{top-1}}$ i $Acc_{\text{top-5}}$ można podać następującą zależność:

$$0 \leqslant Acc_{\text{top-1}} \leqslant Acc_{\text{top-5}} \leqslant 1 \tag{2.1}$$

Warto podkreślić, że w odróżnieniu od funkcji straty stosowanej do klasyfikacji, miary $Acc_{\text{top-1}}$ i $Acc_{\text{top-5}}$ nie uwzględniają wartości ufności, a tylko pozycję klasy wzorcowej w rankingu ufności.

Wyniki eksperymentów i wykresy wskazują na bardzo podobne zachowanie modeli – w kolejności od najlepszego do najgorszego: VGG-16, GoogLeNet, VGG-CNN-S i AlexNet. Modele utrzymują względne wartości swojej skuteczności na stałym poziomie, z wyjątkiem addytywnego zaszumienia obrazu, w którym model VGG-CNN-S wykazuje większą wrażliwość i dla wartości $\sigma > 50$ osiąga wynik nieznacznie gorszy niż AlexNet.

Osobliwie wygląda również zachowanie modeli w kompresji JPEG dla $Q \leqslant 6$, gdzie najgorszy wynik uzyskuje GoogLeNet, a AlexNet ma najlepszą wartość $Acc_{\text{top-5}}$ (ale wynosi ona zaledwie ok. 20%). Takie anomalie można wytłumaczyć bardzo silnym zdegradowaniem jakości przy tak niskim Q i właściwie losowym zachowaniem modeli.

Generalnie, różnice były widoczne pomiędzy różnymi metodami dystorsji jakości, ale w ramach każdej z metod badane modele zachowywały się podobnie. Poszczególne zaburzenia jakości wykazywały następujący wpływ na skuteczność klasyfikacji:

- szum addytywny – dla $\sigma < 10$ bardzo nieznaczny wpływ, następnie równomierny spadek miary *Acc*, za wyjątkiem modelu VGG-CNN-S (spadek szybki). Wpływ szumu określono jako „średni".

- rozmycie Gaussa – szybki spadek miar *Acc* do wartości bliskiej zeru w przedziale $1 \leqslant \sigma \leqslant 6$, również dla modelu VGG-16. Wpływ rozmycia określono jako „silny".

- kontrast – znikomy spadek miary *Acc* w przedziale $0.6 \leqslant \alpha \leqslant 1$, widoczny spadek dopiero dla $\alpha \leqslant 0.4$. Im „lepszy" model, tym wyższa odporność – AlexNet zaczyna tracić dokładność dla $\alpha \leqslant 0.6$, GoogLeNet i VGG-CNN-S mają bardzo podobne wyniki i silny spadek zaczyna się dla $\alpha \leqslant 0.3$, natomiast VGG-16 ma tylko

[5]Jest to specjalny rodzaj warstwy splotowej – filtry konwolucji w tej samej warstwie mają różne rozmiary: 1×1, 3×3 i 5×5; do tego możliwa jest redukcja wymiaru przestrzennego mapy cech poprzez zastosowanie kroku (ang. *stride*), a wówczas oprócz wyników konwolucji dołączane są oryginalne warstwy wejściowe poddane operacji *max pooling*.

kilkuprocentowy spadek przy $\alpha = 0.2$ i około dwudziestoprocentowy przy $\alpha = 0.1$ (ostatnia wartość, przy której obraz nie jest całkowicie wymazywany). Wpływ kontrastu określono jako „słaby".

- kompresja JPEG – w przedziale $8 \leqslant \mathrm{Q} \leqslant 20$ bardzo nieznaczne spadki za wyjątkiem GoogLeNet (najgorszy wynik). W przedziale $2 \leqslant \mathrm{Q} \leqslant 8$ silny spadek, ale nie do zera – zależnie od modelu $Acc_{\text{top-1}}$ przyjmuje wartości od 10% do 20%, a $Acc_{\text{top-5}}$ wartości od 20% do 30%. Wpływ kompresji JPEG określono jako „słaby".

- kompresja JPEG2000 – bardzo podobne zachowanie modeli i gładki przebieg wykresów jakości; w przedziale $30 \leqslant \mathrm{PSNR} \leqslant 40$ bardzo mały spadek miar Acc, natomiast w przedziale $20 \leqslant \mathrm{PSNR} \leqslant 30$ spadek dokładności wszystkich modeli do bardzo podobnej wartości (linie zbiegają się): $Acc_{\text{top-1}}$ nieco poniżej 20%, a $Acc_{\text{top-5}}$ nieco powyżej tej wartości. Wpływ kompresji JPEG2000 określono jako „słaby".

Podsumowując, autorzy stwierdzają, że modele głębokie mają niską odporność na rozmycie obrazu, średnią na szum, a wysoką na kontrast i stratną kompresję (tę ostatnią – w rozsądnym zakresie). W omawianej pracy nie rozpatrzono wpływu kompresji JPEG dla przedziału $20 < \mathrm{Q} \leqslant 100$, a także pozostawiono otwarte pytanie o możliwości poprawy wyników przez wzbogacenie zbioru uczącego o próbki o obniżonej jakości. Ta zidentyfikowana luka informacyjna była jednym ze źródeł inspiracji do podjęcia badań przedstawionych w niniejszej pracy.

2.2 Niezawodność sieci konwolucyjnych

Goodfellow i in. [30] analizują tzw. *przypadki adwersarialne* (ang. *adversarial examples*), czyli obrazy, które są błędnie klasyfikowane, nawet przez wiele różnych modeli, ale nie przedstawiają trudności w klasyfikacji przez ludzkiego odbiorę. Autorzy pokazują, w jaki sposób obrazy zaburzone przez dodanie odpowiednio dobranego szumu są klasyfikowane z wysoką ufnością jako zawierające obiekt zupełnie innej klasy, podczas gdy człowiek obserwuje na nich jedynie nieznaczne pogorszenie jakości obrazu. Powodem, zdaniem autorów, jest liniowość zachowania sieci i zbyt niska złożoność (np. polegająca na małej liczbie warstw sieci), umożliwiająca tzw. atak adwersarialny (procedurę, która tworzy na podstawie obrazu wejściowego przypadek adwersarialny, minimalizując jego modyfikację).

Odkrycie istnienia przypadków adwersarialnych, a także możliwości wystąpienia ich w rzeczywistych obrazach, doprowadziło do badań możliwości przeciwdziałania atakom adwersarialnym. Jednym z pomysłów jest użycie kompresji JPEG (z obniżeniem jakości) przed przetworzeniem obrazu przez sieć głęboką, zaproponowane w pracy Dasa i in. [15]. Autorzy przedstawiają kompromis, który polega na pogorszeniu wyniku klasyfikacji „zwykłych" obrazów, ale – w zamian – na uzyskaniu znacznej odporności na zmodyfikowane adwersarialnie obrazy wejściowe.

Bardziej zaawansowane podejście Jia i in. [42], nazwane mechanizmem ComDefend jest również związane z kompresją (odnosi się do tego prefiks nazwy „Com-"), ale wykonywanej nie standardowymi algorytmami, lecz z użyciem głębokich sieci neuronowych.

W kolejnej pracy Yin i in. [105] przedstawione jest podejście oparte ponownie na kompresji stratnej (tym razem jest to kompresja WebP zamiast JPEG) w połączeniu z prostymi przekształceniami obrazu (odbicie lustrzane). Wyniki przedstawiają porównanie trzech metod:

1. kompresji JPEG,
2. metody ComDefend,
3. kompresji WebP z odbiciem lustrzanym (proponowana).

Porównanie to pokazuje wysoką skuteczność ostatniej z metod, która może wynikać z faktu, że kompresja WebP gorzej „przenosi" sygnał powodujący błąd klasyfikacji przy wyższej ogólnej jakości obrazu w stosunku do kompresji JPEG. Inaczej mówiąc, optymalną ochronę przed atakiem adwersarialnym można uzyskać w kompresji JPEG dopiero przy niższej jakości, co wiąże się z niższą skutecznością działania sieci dla „zwykłych" (nie adwersarialnych) obrazów. Metoda ComDefend nie wykazuje jednoznacznej przewagi nad metodami korzystającymi ze standardowych algorytmów kompresji.

Wymienione w tym podrozdziale prace stawiają pytanie o analogię między postrzeganiem obrazu przez człowieka, a „postrzeganiem" go przez modele uczenia głębokiego – czy jest ono zbliżone, czy ma istotnie różną charakterystykę? Znalezienie odpowiedzi na to pytanie, w szczególnym aspekcie, jakim jest stratna kompresja obrazu, było celem badań podjętych w niniejszej pracy.

Rozdział 3

Uczenie maszynowe głębokie

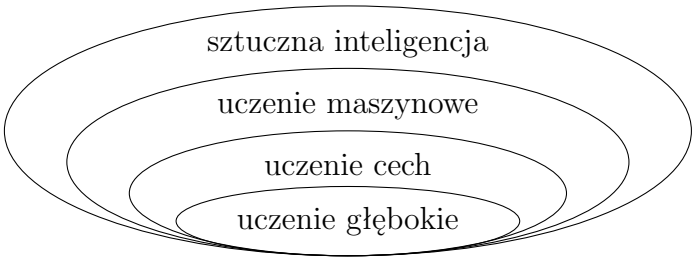
Niniejszy rozdział jest poświęcony metodom uczenia głębokiego i omawia je w takim zakresie, jaki jest niezbędny do dalszej dyskusji na temat opartej na nim detekcji obiektów. Całościowe omówienie dziedziny uczenia głębokiego dalece wykracza poza ramy tej pracy: dziedzina ta cały czas się rozwija i powstaje wiele publikacji raportujących bieżące postępy. Bardzo dobre wprowadzenie do uczenia głębokiego można znaleźć w książce Goodfellowa [29][1], a skrócony przegląd zagadnień z nim związanym jest w artykule tzw. „ojców uczenia głębokiego" [54], czyli LeCuna, Bengio i Hintona. Godny podkreślenia jest fakt, że autorzy ci zajmowali się tym, co aktualnie jest określane jako uczenie głębokie już w latach dziewięćdziesiątych XX w. oraz kierowali lub uczestniczyli w wielu projektach, które przyniosły przełomowe postępy w tej dziedzinie. W tym rozdziale zostaną wprowadzone podstawowe pojęcia związane z uczeniem głębokim, będzie przedstawiona taksonomia modeli oraz omówienie kilku ważnych pojęć z pogranicza teorii i praktyki treningu modeli głębokich.

3.1 Uczenie maszynowe a AI

Graficzna prezentacja relacji zawierania się między poszczególnymi grupami podejść sztucznej inteligencji została przedstawiona na Rys. 3.1.

Sztuczna inteligencja. AI jest to zbiorcza nazwa obejmująca wiele różnych podejść do rozwiązywania problemów, które wymagają inteligencji do rozwiązywania. Jednym z tych podejść jest uczenie maszynowe, ale istnieją inne, m.in. bazy wiedzy, systemy ekspertowe, przetwarzanie języka naturalnego oparte na analizie składniowej, systemy dialogowe oparte na rozpoznawaniu wzorców w wypowiedzi itp. Znaczna część tych metod historycznie poprzedza uczenie maszynowe i dlatego nawet powstało żartobliwe zbiorcze określenie ich jako „stare (staromodne), dobre AI" (ang. *good, old-fashioned AI*, GOFAI) [8].

[1]Książka ta jest dostępna do czytania online: `https://www.deeplearningbook.org/` (dostęp: 27 września 2022)



Ilustracja własna, inspirowana diagramem Venna w [29]

Rys. 3.1: **Zależności między pojęciami sztucznej inteligencji, uczenia maszynowego i uczenia głębokiego.**
Kategoria sztucznej inteligencji zawiera uczenie maszynowe i inne metody AI, które wymagają ręcznego tworzenia logiki. W uczeniu maszynowym mamy klasyczne metody ML oraz uczenie cech (uczenie reprezentacji) – klasyczne metody mają określone wymagania do danych wejściowych i, kiedy problem ich nie spełnia, mogą wymagać inżynierii cech. Uczenie cech to takie uczenie maszynowe, które nie wymaga ręcznej inżynierii cech. Może ono być płytkie lub głębokie: uczenie głębokie to podejście, w którym przekształcenie danych wejściowych do cech podlega uczeniu oraz istnieje hierarchia cech – niektóre („głębsze") powstają przez wytrenowane przekształcania innych cech – ma to miejsce np. w wielowarstwowych sieciach neuronowych, płytkie zaś jest pozbawione hierarchii cech – np. sieć neuronowa bez warstw ukrytych.

Uczenie maszynowe. Metody ML charakteryzują się brakiem konieczności ręcznego programowania algorytmów do rozwiązywania problemów. Są to metody, które wykorzystują gotowy algorytm (np. obliczenie wartości pewnej funkcji, która oprócz argumentów wejściowych przyjmuje dodatkowe parametry, które będą *parametrami modelu*), a do uzyskania ostatecznego rozwiązania potrzebne są dane (zbiór uczący albo inaczej treningowy). Stąd określenie tych metod jako „opartych na danych" (ang. *data-driven*). Zbiór uczący zawiera zmienne niezależne (zmienne objaśniające, predyktory, X) i zmienne zależne (objaśniane, y) – te, które model ML ma przewidywać dla nowych danych. Wartości y w zbiorze uczącym będziemy określać jako wartości wzorcowe, funkcjonują też określenia: wartości empiryczne i „prawda absolutna" (od ang. *ground truth*). Wartości produkowane przez model na podstawie X, oznaczane $\hat{y}$, są określane jako wartości teoretyczne albo przewidywane.

Prosty przykład ML to regresja liniowa: wyjście modelu to liniowa kombinacja zmiennych wejściowych (mnożniki są parametrami modelu), powiększona o wyraz wolny (ang. *intercept*). Innym przykładem jest heurystyka k najbliższych sąsiadów (ang. k *nearest neighbors*, kNN) – w tym przypadku nie ma nawet procesu uczenia, tylko jest zapamiętanie zbioru uczącego i dla nowych elementów, dla których wyznaczana jest wartość funkcji (proces określany jako predykcja lub *inferencja*), znajdowane jest k najbliższych sąsiadów, a rezultatem jest pewna agregacja wartości wzorcowej tych sąsiadów. Dla klasyfikacji binarnej może to być proste głosowanie większościowe, wówczas dla zapewnienia jednoznacznego wyniku można przyjąć k nieparzyste. Z drugiej strony, dokładny procent sąsiadów, którzy są danej klasy, może być swego rodzaju

miarą prawdopodobieństwa wyniku (ufność), która niesie dodatkową informację o sile tej predykcji.

Modele ML, które nie realizują uczenia reprezentacji, mają określone założenia co do swoich zmiennych objaśniających. Może to być konieczność normalizacji lub unitaryzacji cech, tak aby odległość Euklidesowa pomiędzy wektorami cech miała właściwy sens. W przypadku niektórych metod, kiedy w zbiorze dostępna jest duża liczba zmiennych objaśniających, ekspert ML musi wybrać taki ich podzbiór, który pomoże stworzyć lepszy model (np. usuwając cechy, które mają fałszywe korelacje z wynikiem). Istnieją metody, które pozwalają automatyzować te czynności, ale nie są one częścią samej metody ML. Podobnie z tworzeniem nowych cech na podstawie zmiennych niezależnych – np. w danych medycznych jest dostępna waga i wzrost, ale na potrzeby modelu można usunąć te dwie cechy i zastąpić je indeksem masy ciała (ang. *body mass index*, BMI) — takie operacje wymagają ingerencji eksperta w przygotowanie modelu ML, który nie ma automatycznego uczenia cech.

Uczenie cech. Uczenie cech albo zamiennie uczenie reprezentacji (ang. *representation learning/feature learning*) to grupa metod ML, które nie wymagają inżynierii cech. Nie oznacza to, że nie można wykonywać żadnych operacji na danych uczących, ale będzie to już tylko przetwarzanie wstępne (ang. *preprocessing*), a takie cechy zostaną i tak jeszcze raz przetworzone na etapie uczenia modelu ML.

Większość metod, które realizują ML, stanowią modele uczenia głębokiego. Kategoria taksonomiczna *feature learning* może być dość wąska, jeśli chodzi o modele nie-głębokie; Goodfellow [29] przytacza „płytkie autoenkodery" (ang. *shallow autoencoders*) jako przedstawicieli płytkich metod uczenia cech.

Pomiędzy kategorią „płytkiego ML" a uczeniem cech można zlokalizować maszyny wektorów nośnych/podpierających (SVM) – poprzez zastosowanie transformacji przestrzeni cech za pomocą funkcji jądrowej, co powoduje podniesienie wymiarowości przestrzeni cech, operują one niejawnie na cechach wyższego rzędu wyliczonych z cech pierwotnych. Ułatwia to liniową separację tak przetworzonej przestrzeni.

Uczenie głębokie. Metody, które są w centrum zainteresowania autora niniejszej pracy, to modele uczenia głębokiego, czyli modele uczenia cech, które wielokrotnie przetwarzają przestrzeń cech wejściowych X. Podstawowymi realizacjami uczenia głębokiego są wielowarstwowe sieci neuronowe, zarówno gęste (perceptron wielowarstwowy, ang. *multi-layer perceptron*, MLP), jak i sieci konwolucyjne (splotowe) oraz sieci rekurencyjne (ang. *recursive neural network*, RNN), które mają pętlę sprzężenia zwrotnego – wyjście z poprzedniej inferencji jest podawane na wejście następnej inferencji w tym samym neuronie.

W przetwarzaniu obrazów do ekstrakcji cech dominującą rolę odgrywają sieci splotowe [7, 26, 49, 72, 73, 76, 77]. Do analizy sekwencji (np. szeregi czasowe, ale także język naturalny – jako sekwencja symboli, którymi mogą być albo znaki alfabetu, albo leksemy/słowa) szczególnie przydatne są sieci rekurencyjne takie jak LSTM (ang. *long short-term memory*) i GRU (ang. *gated recurrent unit*). Ponieważ wideo również jest

sekwencją (obrazów), sieci RNN w połączeniu z CNN jako ekstraktorami cech, znalazły zastosowanie w przetwarzaniu wideo, m.in. w zamianie sekwencji wideo na tekstowy opis [95]. Takie połączenie różnych rodzajów sieci nie jest czymś wyjątkowym – łączenie np. sieci gęstych z konwolucyjnymi jest obecne w dużej liczbie *architektur* sieci głębokich.

Architektura sieci to określenie, które odnosi się do sposobu budowy, struktury sieci neuronowej, połączeń między warstwami itp. Jest to logiczny model sieci neuronowej, niezależny od jej fizycznej implementacji (języka programowania, biblioteki do uczenia głębokiego, akceleracji sprzętowej itp.). Sieć o danej architekturze będzie zawsze działać w ten sam sposób, z dokładnością do *hiperparametrów*.

Hiperparametr (ang. *hyperparameter*) to wartość lub cecha, która ma wpływ na ostateczną postać sieci albo na sposób uczenia modelu. Przykładem może być liczba warstw w sieci, liczba powtórzeń określonej grupy warstw, liczby neuronów albo filtrów splotowych w każdej warstwie (cechy modelu), albo np. stała uczenia i harmonogram jej zmiany w trakcie treningu, liczba iteracji procesu uczenia, wartości parametrów regularyzacji (cechy procesu uczenia modelu). Najważniejsza różnica między hiperparametrem a *parametrem* modelu ML jest taka, że proces uczenia nie zmienia wartości hiperparametrów, optymalizuje jedynie dobór parametrów dla zadanego zbioru uczącego. Optymalizacja w zakresie hiperparametrów jest możliwa poprzez wielokrotne wykonanie procesu treningu sieci, co jest kosztownym procesem; może być ona wykonana ręcznie lub przeszukiwana automatycznie (tzw. AutoML, automatyczna optymalizacja hiperparametrów, która może nawet testować różne architektury i wybrać najlepszą kombinację architektury z hiperparametrami).

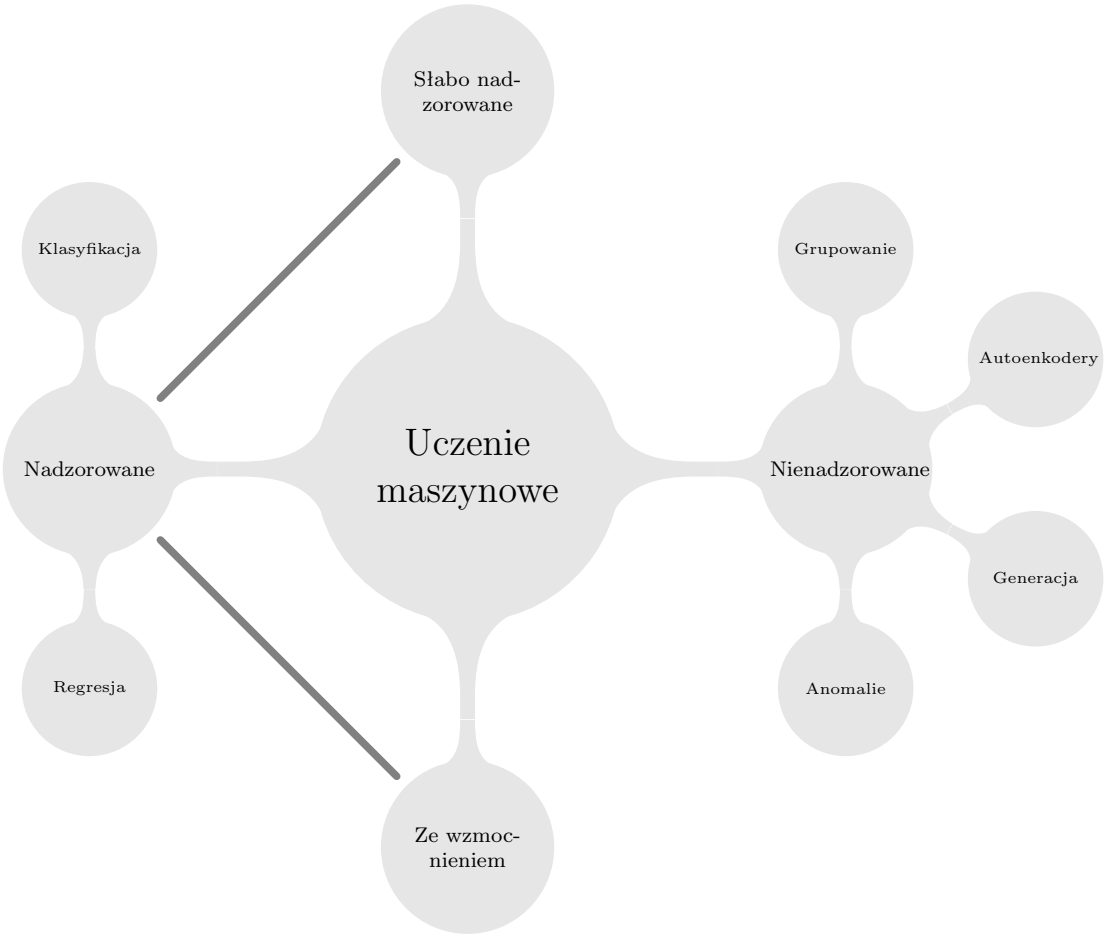
Kompresja modeli głębokich. Podsumowując taksonomię modeli uczenia maszynowego, można wspomnieć o specyficznym przypadku, który dotyczy uczenia reprezentacji, mianowicie o tworzeniu modeli płytkich na podstawie modeli głębokich, określanym jako kompresja sieci [47]. Polega to na trenowaniu modelu płytkiego, który przybliża głębokie cechy przedostatniej warstwy modelu głębokiego. Sama ostatnia warstwa także jest modelem płytkim i, dołączona za wspomnianym płytkim modelem aproksymującym jej cechy wejściowe, daje model aproksymujący całe działanie modelu głębokiego.

Doświadczenia autorów sieci PVANet pokazują, że można w ten sposób uzyskać wyniki odbiegające najwyżej o kilka procent od oryginalnej głębokiej sieci (bez realnej możliwości uzyskania wyników lepszych), za pomocą modelu o radykalnie niższej złożoności obliczeniowej [47].

3.2 Uczenie nadzorowane i nienadzorowane

Taksonomia rodzajów metod i zadań uczenia maszynowego została przedstawiona na Rys. 3.2. Podstawowy podział uczenia maszynowego (który dotyczy tak samo ucze-

nia głębokiego) to wyróżnienie metod uczenia nadzorowanego[2] (ang. *supervised learning*) i nienadzorowanego (ang. *unsupervised learning*).



Rys. 3.2: **Taksonomia uczenia maszynowego.**
Uczenie maszynowe dzieli się na nadzorowanie i nienadzorowane. W uczeniu nadzorowanym główne zadania to klasyfikacja i regresja. W uczeniu nienadzorowanym występuje grupowanie, czyli klasteryzacja (ang. *clustering*), autoenkodery: modele, które odwzorowują wejście na wyjściu, modele generacyjne, które uczą się generować nowe dane, podobne do danych uczących (np. o takich samych rozkładach cech) oraz modele wykrywające anomalie i obserwacje odstające (ang. *outliers*). Osobnymi kategoriami są: uczenie słabo nadzorowane (w którym informacja wzorcowa jest niepełna lub niedokładna), oraz uczenie ze wzmocnieniem (symulacja interakcji z wirtualnym światem, zamiast informacji wzorcowej jest system kar i nagród).

Uczenie nadzorowane. Kiedy w zbiorze uczącym dostępne są wartości empiryczne y (wzorcowe), wówczas mamy do czynienia z zagadnieniem uczenia nadzorowanego.

[2]spotykane jest określenie „uczenie z nauczycielem"

Model zwraca wartości przewidywane $\hat{y}$, a *funkcja straty* (ang. *loss function*) wyraża ilościowo, jak bardzo $\hat{y}$ różni się od y. Proces uczenia modelu ma za zadanie zminimalizować funkcję straty.

Klasyfikacja i regresja. Klasyfikacja i regresja to dwa różne zadania ML realizowane w kontekście uczenia nadzorowanego. O tym, jakiego rodzaju jest dane zadanie ML, decydują skale pomiarowe zmiennej objaśnianej, tzn. własności zbioru, do którego należą wartości y (oraz $\hat{y}$). Upraszczając, jeśli dziedzina y jest:

- dyskretna – model rozwiązuje zadanie klasyfikacji,
- ciągła – model rozwiązuje zadanie regresji.

Skale pomiarowe podlegają dalszemu podziałowi, do dyskretnych można zaliczyć:

- skala nominalna: skończony zbiór symboli, np. kolor oczu, gatunek muzyczny, rasa psa itp. Nawet jeżeli reprezentujemy wartości za pomocą liczb, to operatory relacji porządku $<$ i $>$, jak i jakiekolwiek operacje na nich nie mają sensu.
- skala porządkowa: skończony lub przeliczalny zbiór wartości, które można porównywać za pomocą operatorów $<$ i $>$, np. szkolne oceny z zachowania: naganne, nieodpowiednie, odpowiednie, bardzo dobre, wzorowe. Mogą być reprezentowane przez liczby naturalne, natomiast ich różnica i iloraz nie mają sensu.

Skale ciągłe można podzielić na:

- skalę przedziałową: ciągłe wartości z pewnego zbioru. Różnica dwóch wartości ma sens, podobnie jak relacja porządkująca. Przykład: temperatura w stopniach Celsiusza (jeżeli ciało b ma temperaturę wyższą o jeden stopień od ciała a, a ciało c ma temperaturę wyższą o jeden stopień od ciała b, to ciało c ma temperaturę wyższą o dwa stopnie wyższą od ciała a, *ale* iloraz temperatur 15 °C i -5 °C nie ma żadnego sensu).
- skalę ilorazową: wszystkie cechy skali przedziałowej plus istnienie sensu dla ilorazu wartości (oprócz dzielenia przez zero). Przykład: skala Kelvina (iloraz temperatur w tej skali ma sens fizyczny – gaz doskonały będzie miał w warunkach stałego ciśnienia objętość proporcjonalną do temperatury w skali Kelvina).

Skala pomiarowa nominalna dla zmiennej zależnej wskazuje na typowe zadanie klasyfikacji (określane też jako dyskryminacja, tj. rozróżnienie). Wartość w tej skali często jest reprezentowana jako wektor bitów (wartości Boolowskich) o długości równej rozmiarowi zbioru, i w takim wektorze jest jedna jedynka na pozycji odpowiadającej danej kategorii, a zera na pozostałych pozycjach. W wersji rozmytej, która jest stosowana w odpowiedzi modelu, będzie to wektor prawdopodobieństw, które powinny sumować się do jedności.

Skala porządkowa pozwala na zastosowanie zarówno klasyfikacji (nie korzystamy wówczas z porządku w zbiorze i traktujemy wartości jak pochodzące ze skali nominalnej), jak i regresji (potrzeba wtedy reprezentować wartości jako uporządkowany zbiór

liczb, a jeżeli model zwraca ciągłą odpowiedź, trzeba dokonać zaokrąglenia do najbliższej dyskretnej wartości).

Skale ciągłe dla zmiennej zależnej jednoznacznie wskazują na zadanie regresji.

Zależnie od zadania ML będą dostępne inne metody jego rozwiązywania (np. regresja liniowa do regresji, a regresja logistyczna do klasyfikacji), a w przypadku uczenia głębokiego dostępne będą różne funkcje straty, przykładowo:

- klasyfikacja: entropia wzajemna (ang. *binary cross-entropy*), dywergencja KL (Kullbacka-Leiblera),
- regresja: średni błąd kwadratowy (ang. *mean squared error*, MSE), metryka Euklidesowa (dla wektorów wartości).

Skale pomiarowe mają znaczenie przy przetwarzaniu wstępnym zbioru danych, np. wartości ze skali ilorazowej po liniowym przeskalowaniu na przedział $[-1, 1]$ lub $[0, 1]$ mogą już być traktowane tylko jako wartości mierzone na skali przedziałowej i trzeba brać pod uwagę konsekwencje tego faktu przy stosowaniu modelu. Zwłaszcza, że skale pomiarowe dotyczą również zmiennych objaśniających, które tak samo mogą podlegać przetwarzaniu wstępnemu – metody ML wymagają liczb, i to często o znormalizowanym rozkładzie, aby zapewnić stabilność treningu i wpływu poszczególnych cech na wynik. Dlatego skale pomiarowe i ich właściwości mają znaczenie również dla metod uczenia nienadzorowanego (w którym wszystkie zmienne są „objaśniające").

3.3 Transfer uczenia

W pedagogice *transfer* albo *transfer umiejętności* opisuje zjawisko polegające na ułatwieniu nauki pewnej umiejętności poprzez wcześniejszą naukę innej, niezwiązanej umiejętności. Zjawisko to można zaobserwować w wielu dziedzinach, np. przy analizie wyników nauczania pokrewnych przedmiotów albo języków obcych (nauka następnego języka obcego). Związane z tym badania mogą dotyczyć również niezwiązanych z pozoru umiejętności: np. wpływu nauki gry w szachy na wyniki w nauce matematyki [80] – zastąpienie części zajęć z matematyki nauką gry w szachy zaowocowało poprawą wyników badanej grupy w niektórych umiejętnościach matematycznych oraz równoważnymi wynikami w innych umiejętnościach w porównaniu do grupy kontrolnej (mimo że czas poświęcony na matematykę był krótszy o jedną godzinę w tygodniu).

Dla modeli uczenia głębokiego funkcjonuje pojęcie transferu uczenia lub uczenia transferowego (ang. *transfer learning*). Polega ono na przeprowadzeniu dwóch treningów (procesów dopasowania modelu): pierwszego na jednym zbiorze uczącym – w celu realizacji jednego zadania (np. klasyfikacja obrazów), a następnie drugiego na innym zbiorze – z innym zadaniem (np. detekcja obiektów, segmentacja obszarów itp.). Pierwszy z tych treningów jest to trening wstępny, inaczej pre-trening (ang. *pre-training*), natomiast drugi jest określany jako dostrajanie (ang. *fine tuning*).

Technicznie rzecz biorąc, można patrzeć na transfer uczenia jako na metodę inicjalizacji wag połączeń. Przy zwykłym treningu sieci wagi są inicjowane wartościami generowanymi losowo, z ewentualnym dodatkiem ograniczeń, które mają zapewnić określone

własności statystyczne wag oraz odpowiedzi warstw sieci neuronowej (np. inicjalizacja Xaviera [28] albo inne metody [50]). W przypadku transferu uczenia, dostrajanie modelu to standardowy proces treningu sieci neuronowej, z tym, że wagi są zainicjowane wartościami uzyskanymi w innym treningu (który mógł mieć losowe wartości początkowe wag połączeń).

Transfer uczenia umożliwia trenowanie modeli do nowych zadań. Trening na tym samym zbiorze i dla tego samego zadania nie jest transferem uczenia, może to być traktowane jako kontynuacja tego samego uczenia, *dotrenowywanie*. Zmiana zbioru uczącego i dostrajanie modelu mogą być korzystne w przypadku trudnych danych: wstępne uczenie na łatwiejszym zbiorze (lub podzbiorze) daje zgrubne przybliżenie, które następnie jest doskonalone na trudniejszych danych, dla których istnieje ryzyko braku zbieżności treningu w innej sytuacji.

Najciekawszy przypadek to zmiana zadania, które model realizuje: począwszy od zmiany kategorii w klasyfikacji, a skończywszy na całkowicie innym zadaniu (np. regresja zamiast klasyfikacji) dla tych samych danych wejściowych. Wówczas może się okazać, że struktura sieci musi być zmieniona (różna liczba neutronów wyjściowych prawdopodobieństw poszczególnych kategorii w klasyfikacji, albo nawet kilka warstw sieci z neuronami o zupełnie innym znaczeniu). W tej sytuacji inicjujemy tylko te wagi, które będą wspólne dla struktury sieci z pre-treningu i z dostrajania. Pozostałe wagi są inicjowane losowo, tak jak przy zwykłym treningu sieci[3].

W przypadku wielu architektur sieci splotowych będą dostępne gotowe zestawy wag pre-trenowanych np. na zbiorze ImageNet [79]. Jest to popularny punkt wyjścia do dostrajania klasyfikatorów obrazów, ale również wiele modeli wykrywania obiektów korzysta z takiego sposobu inicjalizacji wag. Ponieważ jest to inne zadanie, nie można wykorzystać całego zbioru wag, a tylko pewną liczbę początkowych warstw, która służy ekstrakcji głębokich cech obrazu; ta część jest określana jako kręgosłup detektora.

Częściowe zamrażanie wag. Kiedy dostrajanie modelu dotyczy innego zadania, wyjściowa część sieci neuronowej początkowo nie jest wytrenowana. Powoduje to duże błędy (wartości funkcji straty) i tym samym silny sygnał uczący (gradient funkcji straty). Prowadzi to do znaczących modyfikacji wag sieci w każdej iteracji, co jest korzystne dla niewytrenowanej części, ale może zaburzyć wagi kręgosłupa, tj. pretrenowanej części sieci. Aby temu zapobiec, stosuje się „zamrażanie" wag, tj. blokadę ich modyfikacji. Takie zamrożenie zmniejsza również złożoność obliczeniową treningu (dla tych warstw neuronów nie ma potrzeby liczenia gradientów i aktualizacji wartości wag, ponieważ są tylko do odczytu, mogą być współdzielone pomiędzy wątkami GPU itd.). Kiedy trening jest zaawansowany (wartości funkcji straty są niższe), można w końcowej fazie dostrajania „odmrozić" wszystkie wagi i w niektórych przypadkach uzyskać nieznaczne polepszenie, przez lepsze dostosowanie kręgosłupa sieci do konkretnego zadania.

[3]Niemniej jednak, jeżeli korzystamy z transferu uczenia dla klasyfikacji i liczba klas pozostaje taka sama, ale ich znaczenie jest zupełnie inne, to wyjściową warstwę klasyfikacji należy i tak zainicjować losowymi wartościami.

Zalety podejścia. Transfer uczenia pozwala na szybkie uzyskanie dobrych efektów uczenia modeli dla wielu nowych zadań uczenia maszynowego. Umożliwia skrócenie czasu treningu zarówno dla typowych, jak i nowych zadań, a także trening stosunkowo dużych modeli z użyciem tańszego sprzętu (GPU o mniejszej ilości pamięci RAM), szczególnie kiedy stosowane jest częściowe zamrożenie wag. Możliwe jest uczenie z użyciem mniejszych zbiorów uczących, korzystając ze zdolności uogólniającej uzyskanej na większym zbiorze zastosowanym w pre-treningu: dzięki temu można rozwiązywać zadania, dla których zbyt kosztowne jest stworzenie zbioru uczącego wystarczającego do pełnego treningu dużej sieci konwolucyjnej.

Wady podejścia. W niektórych sytuacjach transfer uczenia nie może zostać zastosowany – może nie być odpowiedniego zbioru do pre-trenowania albo dostrajany model może nie dawać lepszych wyników niż model uczony od losowej inicjalizacji. Zamrożenie początkowych warstw modelu może zapobiegać wypracowaniu ekstrakcji cech adekwatnej do rozważanego problemu.

Niemniej jednak, najważniejsze ograniczenie uczenia transferowego to ścisłe powiązanie z konkretną architekturą sieci – zmiana architektury lub hiperparametrów modelu wymaga ponownego pre-treningu. W połączeniu z dużymi rozmiarami zbiorów używanych do pre-treningu (np. ImageNet), generuje to znaczne koszty dla każdej osobnej architektury sieci, która ma być sprawdzona w docelowym zadaniu. W tej sytuacji może się okazać, że jednoetapowy trening od losowej inicjalizacji, inaczej mówiąc *od zera* (ang. *from scratch*), jest efektywniejszy: kosztuje mniej niż dwa treningi łącznie, a uzyskane wyniki są równie dobre lub lepsze.

Zastosowanie w detekcji obiektów. Transfer uczenia jest szeroko stosowany w pracach dotyczących detekcji obiektów [36,72–74,76,77]. Najczęściej pre-trening jest przeprowadzany w zadaniu klasyfikacji obrazów na zbiorze ImageNet (ang. *ImageNet Large-Scale Visual Recognition Challenge*, ILSVRC) z tysiącem rozpoznawanych kategorii. Dostrajanie, a tym samym właściwy trening detekcji obiektów, jest przeprowadzane na zbiorze dotyczącym detekcji obiektów, jak np. PASCAL VOC (ang. *visual object classes*) [20] albo Microsoft COCO (ang. *common objects in context*) [59]. Detekcja obiektów może być trenowana także dla nowych klas, z użyciem pre-trenowanych kręgosłupów, które są dostępne do pobrania, a które zostały przygotowane m.in. przez Microsoft Research Lab Asia (MSRA) i Facebook AI Research (FAIR). Wagi połączeń są dostępne m.in. dla sieci: ResNet-50, ResNet-101 i ResNeXt-101.

Z drugiej strony, eksperymentowanie z własnymi architekturami sieci jest znacznie trudniejsze, na co wskazuje sam rozmiar zbiorów: zbiór ILSVRC[4] do pre-treningu (na zadaniu klasyfikacji), to niemal 155 GB, natomiast stosunkowo duży zbiór do detekcji obiektów COCO[5] `train2017` to około 18 GB. Postęp w dziedzinie mocy obliczeniowej, dostępności pamięci dyskowej i szybkości transmisji danych w Internecie może w

[4] `https://www.kaggle.com/c/imagenet-object-localization-challenge/data` (dostęp: 27 września 2022)

[5] `https://cocodataset.org/#download` (dostęp: 27 września 2022)

przyszłości sprawić, że te rozmiary będą nieistotne, ale w chwili obecnej taka różnica stanowi istotne utrudnienie dla małych jednostek badawczych i niezależnych badaczy.

Możliwości treningu modeli detekcji bezpośrednio, od losowej inicjalizacji wag sieci, zbadano w pracy He [33]. Uzyskane rezultaty pokazały, że uczenie transferowe architektur dostępnych w bibliotece Detectron2 [103] nie jest konieczne do uzyskania skutecznego modelu detekcji obiektów. Końcowy wynik skuteczności detekcji nie różni się istotnie od modeli pre-trenowanych i dostrajanych. W tym celu potrzebne jest przedłużenie procesu uczenia około trzykrotnie w stosunku do samego dostrajania, np. zamiast 27 epok (całkowitych przebiegów zbioru `train2017`) należy ich zastosować 81. W zamian uzyskuje się możliwość zmian w architekturze (np. tworzenie większych lub mniejszych modeli od tych dostępnych standardowo) bez konieczności kosztownego treningu wstępnego.

3.4 Uczenie ze wzmocnieniem

Uczenie ze wzmocnieniem (ang. *reinforcement learning*, RL) jest inspirowane psychologią. W psychologii badania nad warunkowaniem zachowań (ang. *conditioning*) zapoczątkował Iwan Pawłow, który badał zachowania psów. Proces warunkowania, czyli wyrabiania w przedmiocie reakcji pożądanych z punktu widzenia warunkującego, obejmuje stosowanie wzmocnienia pozytywnego (ang. *positive reinforcement*) i negatywnego (ang. *negative reinforcement*), czyli mówiąc krótko: nagród i kar. Wzmocnienie w procesie warunkowania zostało również opisane w sposób matematyczny [46]. Warunkowanie zwierząt jest też określane tresurą, a w odniesieniu do ludzi funkcjonuje w różnych kontekstach: marketing, terapia uzależnień, resocjalizacja itp.

Technika RL czerpie z psychologii poprzez stosowanie funkcji nagrody, która może być ujemna lub dodatnia (jak pozytywne i negatywne wzmocnienie). Jest to różnica w stosunku do funkcji straty, ponieważ funkcja straty jest minimalizowana, a funkcja nagrody jest maksymalizowana; poza tym funkcja straty ma znane minimum (kiedy odpowiedź modelu jest równa wzorcowej), a maksimum funkcji nagrody nie musi być znane. Pozostałe elementy systemu RL [86] to model świata, w którym działa *agent*, a także dostępne dla niego bodźce (informacje o stanie symulacji), oraz możliwe do podjęcia akcje (ang. *policy*). Przykładem RL jest k-ręki bandyta (ang. *k-armed bandit*): zbiór akcji jest dyskretny i w każdym kroku czasowym agent wybiera jedną z nich. Innym pojęciem związanym z RL jest funkcja wartości (ang. *value function*), która jest sumą zdobytych nagród od początku symulacji.

Spektakularne sukcesy RL dotyczą gier, które łatwo zasymulować w komputerze, np. klasycznych gier platformowych z 8-bitowych maszyn Atari [64] oraz szachów [102]. Szczególnie te ostatnie oraz gra Go [25] były historycznie uznawane (np. przez Alana Turinga w latach 50 XX w.) za przykłady „prawdziwego AI". Niemniej jednak, ze współczesnego punktu widzenia nie są to dobre przykłady [29], ponieważ zasady obydwu tych gier można łatwo opisać za pomocą kilkudziesięciu linii kodu źródłowego. Znacznie trudniejsze jest widzenie maszynowe oraz przetwarzanie języka naturalnego (konwersacja komputera z człowiekiem w sposób nieodróżnialny od innego człowieka

to tzw. test Turinga).

Do wad RL należy m.in. duża trudność uczenia modeli. Funkcja nagrody bardzo często jest rzadka i nieciągła, np. wiele akcji daje zerową nagrodę i dopiero ciąg decyzji prowadzi do pozytywnego lub negatywnego rezultatu – w ten sposób RL jest podobny do rekurencyjnych sieci neuronowych i analizy sekwencji. Dlatego jedno z podejść do uczenia modeli RL nie wymaga wstecznej propagacji błędu, ale opiera się na bezgradientowej optymalizacji za pomocą algorytmów ewolucyjnych (ang. *evolutionary algorithms*) [65].

Możliwe jest zastosowanie RL do problemów uczenia nadzorowanego, korzystając z funkcji straty jako sygnału nagrody. Można dokonać pewnej „grywalizacji" (ang. *gamification*) problemu, która jednak ma znaczenie praktyczne i np. w ten sposób uzyskać klasyfikator z uwzględnieniem ważności cech [52]. Zastosowanie takiego podejścia mogłoby posłużyć do medycznej diagnozy *online*, gdzie kolejne cechy, które należy pozyskać są wynikiem procedury diagnostycznej, która ma swój koszt (ujemne wzmocnienie). Agent w danym kroku czasowym ma do wyboru zlecenie kolejnego badania (pozyskanie nowej cechy) albo zwrócenie diagnozy (poprawna: wysoka nagroda, błędna: wysoka kara). Taki model, po skutecznym nauczeniu, maksymalizuje trafność diagnozy i minimalizuje koszt jej postawienia.

Reasumując, RL pozwala na rozwiązywanie problemów, dla których nie potrafimy w danej sytuacji podać odpowiedzi wzorcowej (jak w uczeniu nadzorowanym), ale po dłuższym czasie możemy określić, jaki był wynik sumy tych odpowiedzi. Jest to kolejny krok w stronę automatyzacji znajdowania rozwiązań w sytuacji braku możliwości podania formalnego algorytmu. Ogólnie, RL jest bardzo interesującą i obiecującą techniką, której zastosowania znacząco wykraczają poza tworzenie AI do sterowania postaciami w grach komputerowych.

3.5 Podsumowanie

Przedstawiony zarys tematyki uczenia głębokiego pozwala scharakteryzować używane w niniejszej pracy metody jako uczenie głębokie (z użyciem głębokich sztucznych sieci neuronowych) i nadzorowane (z pełną informacją o oczekiwanej odpowiedzi modelu).

Rozdział 4

Detekcja obiektów

Niniejszy rozdział zawiera opis zadania detekcji obiektów, charakterystykę starszych metod opartych na widzeniu maszynowym i aktualnych metod opartych na uczeniu głębokim. Detekcję obiektów opisano poprzez określenie danych wejściowych i postaci oczekiwanych wyników. Następnie skrótowo zaprezentowano wcześniejsze podejścia, które nie wykorzystywały głębokiego uczenia maszynowego. Metody detekcji obiektów oparte na uczeniu głębokim przedstawiono szczegółowo, z uwzględnieniem budowy sieci neuronowych i podziału na składowe elementy funkcjonalne, po czym dokonano przeglądu znanych z literatury modeli realizujących detekcję obiektów.

4.1 Zadanie detekcji obiektów

Detekcja obiektów nie poddaje się w prosty sposób definicji formalnej, takiej jak „dla danego grafu ważonego znaleźć cykl odwiedzający jednokrotnie wszystkie wierzchołki, który minimalizuje sumę wag krawędzi" (problem komiwojażera), albo „dla ciągu symboli alfabetu (napisu) podać indeksy wystąpień zadanego spójnego podciągu" (problem wyszukiwania wzorca w tekście). Wynika to z tego, że trudno jest matematycznie zdefiniować fakt „znajdowania się" obiektu na obrazie. Wykrywanie krawędzi i prostych obiektów geometrycznych (linie, elipsy) może być opisane za pomocą warunków określonych np. na zachowaniu gradientu intensywności w pewnym spójnym podzbiorze pikseli.

Natomiast obiekty, które ma wykrywać detekcja obiektów, są znacznie bardziej skomplikowane – wewnątrz każdej klasy może występować bardzo duża zmienność, ze względu na następujące czynniki:

- rozmiar obiektów (rzeczywisty i na obrazie),
- kształt obiektów,
- ułożenie – poza (ang. *pose*), orientacja w przestrzeni,
- barwa obiektów,

- kontekst – otoczenie obiektu, tło,
- warunki oświetleniowe i pogodowe sceny widocznej na obrazie (również pora dnia, miejsce wewnątrz lub na zewnątrz budynku itp.),
- jakość obrazu zawierającego obiekt.

Klasy obiektów, które aktualnie można wykrywać na obrazach, są bardzo liczne. Detekcja obiektów na zbiorze ImageNet (ILSVRC [79] obejmuje tysiąc klas, a wariant detektora obiektów YOLOv2 o nazwie YOLO9000 [73] ma możliwość detekcji obiektów należących do ponad 9000 kategorii, w dodatku zorganizowanych hierarchicznie (np. wykrycie psa na obrazie przypisze go do klas: zwierzę, pies i do klasy związanej z konkretną rasą).

Zadania detekcji obiektów w większości prac nie definiuje się jako formalnego problemu, ale podaje się opis, jak np. Oksuz: „Detekcja obiektów jest to jednoczesna estymacja kategorii i położenia wystąpień obiektów na danym obrazie"[1] [66]. Można jednak formalnie zdefiniować, jak wyglądają dane wejściowe dla zadania detekcji obiektów i jaka jest postać oczekiwanych wyników.

Dany jest zbiór N kategorii (klas obiektów), które należy wykrywać. Detekcja obiektów przyjmuje na wejściu obraz, zazwyczaj kolorowy obraz RGB, i zwraca wyniki detekcji obiektów, które mają postać krotek $(r_1, r_2, r_3, r_4, c_i, p)$, gdzie:

- r_1, r_2, r_3, r_4 są liczbami opisującymi minimalny prostokąt obejmujący wykryty obiekt, równoległy do krawędzi obrazu (prostokąt obiektu). Różne warianty interpretacji tych liczb przedstawiono w dalszej części tekstu.
- $c_i \in \mathbb{Z} \cap [1, N]$ to indeks kategorii wykrytego obiektu.
- $p \in [0, 1]$ jest prawdopodobieństwem teoretycznym (estymowanym przez model) poprawności danej detekcji, wyraża ufność modelu.

Wyróżnia się następujące konwencje reprezentacji prostokąta obiektu:

- x, y, w, h – współrzędne środka prostokąta (x, y) oraz jego wymiary: w – szerokość, h – wysokość.
- x_0, y_0, w, h – współrzędne lewego górnego narożnika prostokąta (x_0, y_0) oraz jego wymiary: w – szerokość, h – wysokość.
- x_0, y_0, x_1, y_1 – współrzędne lewego górnego narożnika prostokąta (x_0, y_0) oraz przeciwnego, prawego dolnego narożnika (x_1, y_1).

[1] Object detection is the simultaneous estimation of categories and locations of object instances in a given image.

Niezależnie od konwencji, opisanie prostokąta obiektu wymaga czterech liczb. Biblioteki do detekcji obiektów (np. Detectron2 [103]) i do przetwarzania wyników tej detekcji (np. COCO API) zawierają funkcje automatycznego przeliczania reprezentacji prostokątów pomiędzy konwencjami.

Odnosząc detekcję obiektów do taksonomii zadań uczenia maszynowego, można ją sklasyfikować jako połączenie kilku zadań. Model uczenia maszynowego, który realizuje detekcję obiektów, jest zatem modelem wielozadaniowym (ang. *multi-task*). Jako zagadnienie uczenia nadzorowanego, wchodzą w grę zadania klasyfikacji i regresji; dokładny zestaw tych zadań zależy od sposobu realizacji detekcji obiektów. Może on obejmować następujące zadania uczenia maszynowego:

- wskazanie kategorii obiektu – klasyfikacja (wiele rozłącznych kategorii albo, przy N kategoriach *niezależnych*, N osobnych klasyfikacji binarnych),
- lokalizacja obiektu – regresja (stanowią ją 4 osobne regresje współrzędnych lub rozmiarów prostokąta obiektu),
- klasyfikacja prostokąta, czy faktycznie zawiera obiekt jakiejkolwiek klasy (w niektórych modelach) – klasyfikacja (binarna),
- maska położenia obiektu (w modelach, które dodatkowo segmentują instancje) – klasyfikacja (maski binarne) lub regresja intensywności (maski rozmyte, ang. *heatmap*).

4.2 Widzenie maszynowe

Wiele prac dotyczących detekcji obiektów zaczyna się zdaniem „detekcja obiektów to ważny/kluczowy/fundamentalny problem w dziedzinie widzenia maszynowego" (przykłady: [2, 48, 66]).

Wczesne metody, których przegląd można znaleźć w książce Shapiro i Stockmana [82], opierały się na prostych algorytmicznych podejściach do wykrywania obiektów. Były możliwe do zastosowania tylko w warunkach prostej separacji obiektów od tła, np. poprzez progowanie wartości pikseli, a następnie analizę spójnych obszarów w binarnych maskach. Takie podejścia mogą być skuteczne w warunkach kontrolowanych, np. w przemysłowym widzeniu maszynowym, dla obiektów podświetlonych od tyłu (ang. *backlight*).

Proste obiekty geometryczne (linie, łuki, koła, elipsy) mogły być wykrywane za pomocą detekcji krawędzi i transformacji Hougha. Bardziej skomplikowane obiekty wykrywano za pomocą ręcznej ekstrakcji cech, po której stosowano modele klasycznego uczenia maszynowego. Przykładem klasy obiektów, której poświęcono szczególną uwagę jest ludzka twarz. Wykrywano ją początkowo tylko „od przodu" (*en face*) i tylko w kwadratowym obszarze. Jednym ze stosowanych podejść było traktowanie kwadratu pikseli jako macierzy kwadratowej, a następnie redukcja wymiaru cech (co wynikało z ograniczeń metod uczenia maszynowego dostępnych w tamtym czasie) – np. przez

analizę głównych składowych (ang. *principal component analysis*, PCA) albo rozkładu według wartości osobliwych (ang. *singular value decomposition*, SVD).

Przełomem w dziedzinie wykrywania twarzy było użycie ekstrakcji cech za pomocą falek Haara, które były następnie klasyfikowane za pomocą kaskadowego klasyfikatora komitetowego zbudowanego z użyciem algorytmu AdaBoost. Od autorów pracy, którymi są Viola i Jones [97], detektor ten jest nazywany detektorem Violi-Jones'a. Analizę działania tego detektora można znaleźć w pracy Wanga [99].

Ulepszenia metod widzenia maszynowego połączonych z uczeniem maszynowym polegały z jednej strony na wynajdywaniu lepszych deskryptorów cech (ang. *feature descriptor*), a z drugiej na stosowaniu skuteczniejszych metod uczenia maszynowego. Przykładem takiego postępu jest detektor osób korzystający z histogramów zorientowanych gradientów (HoG) oraz klasyfikatora SVM, który został zaproponowany w pracy Dalala i Brigsa [14]. Wyznaczenie wartości deskryptora HoG następuje w każdym oknie obrazu i polega na zliczeniu kierunków gradientu jasności pikseli. Liczba kątów (możliwych kierunków gradientu), która wyznacza długość deskryptora, jest parametrem, który można dobrać doświadczalnie, zależnie od klasy wykrywanego obiektu – wykrywanie sylwetek ludzkich działa skutecznie przy dziewięciu kubełkach histogramu HoG, natomiast detekcja np. kotów może być skuteczniejsza przy większej rozdzielczości kątowej deskryptora.

Ostatnim, najbardziej skomplikowanym podejściem, które bezpośrednio poprzedziło „rewolucję uczenia głębokiego" (2012), był model DPM (ang. *deformable parts model*), czyli „deformowalny" model wieloczęściowy. Ekstrakcja cech w tym modelu również jest oparta na deskryptorze HoG, ale zamiast całego obiektu wykrywane są poszczególne części (np. dla człowieka może być to osobno: głowa, tułów, ręce i nogi). Wykryte części powinny być w określonych położeniach względem siebie (np. głowa wyżej niż tułów), ale dopuszczone są odchylenia do wzorcowego położenia (deformacje obiektu). Same części nie są deformowalne, dlatego czasem, w celu uniknięcia niejasności, używana jest angielska nazwa *deformable part-based models*, która nawiązuje do tytułu pracy Felzenszwalba [21]. Model ten stanowił złoty standard detekcji obiektów do czasów rozpowszechnienia metod uczenia głębokiego.

Późniejsze prace, takie jak metoda MMOD (ang. *max-margin object detection*) autorstwa Kinga [48], pokazały, że metody łączące widzenie maszynowe i uczenie maszynowe mogą być nadal rozwijane i ulepszane. W przypadku metody MMOD, zastosowanie dokładniejszych metod treningu modeli pozwoliło, pomimo stosowania tylko jednego filtra HoG, na uzyskanie rezultatów lepszych niż dawał model DPM.

4.3 Uczenie głębokie w detekcji obiektów

Detekcja obiektów realizowana za pomocą głębokiego uczenia maszynowego jest aktualnie najbardziej skutecznym podejściem do rozwiązywania tego zadania. Przegląd zastosowań uczenia głębokiego w detekcji obiektów można znaleźć w pracy Liu [60], razem z odniesieniami do blisko powiązanych zagadnień: klasyfikacji obrazu, wykrywania twarzy, detekcji osób (pieszych), wykrywania pojazdów i bieżących postępów

w dziedzinie uczenia głębokiego, wykorzystywanych w detekcji obiektów.

Taksonomii modeli detekcji obiektów można dokonać w odniesieniu do dwóch podziałów:

- detektory jednoetapowe (ang. *one-stage*) i dwuetapowe (ang. *two-stage*), określane również jako modele realizujące detekcję gęstą (ang. *dense*) i detekcję rzadką (ang. *sparse*),
- modele, które korzystają z piramidy cech głębokich (ang. *feature pyramid*), i modele, które korzystają z cech zwracanych przez ostatnią warstwę konwolucyjną ekstrakcyjnej części sieci („kręgosłupa").

Przykłady nazw modeli detekcji obiektów w podziale według tych dwóch kryteriów zostały zebrane w Tabeli 4.1.

Tabela 4.1: **Przykłady detektorów obiektów jedno- i dwuetapowych.**

	jednoetapowe	dwuetapowe
z piramidą cech	RetinaNet [58] YOLOv4† [7]	Faster R-CNN FPN [57, 76] Mask R-CNN [34]
bez piramidy cech	YOLOv1 [72] YOLOv2 [73] YOLOv3 [74] SSD [61]	Fast R-CNN [26] Faster R-CNN C4 [76] Faster R-CNN DC5 [76]

† – opcja piramidy cech.

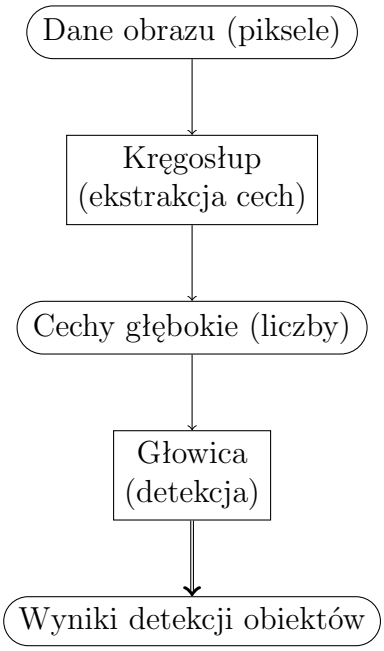
4.3.1 Przepływ danych w detektorze obiektów

Detektory obiektów oparte na splotowych sieciach neuronowych mogą być zrealizowane jako wiele lub jedna sztuczna sieć neuronowa, ale niezależnie od tego można w nich wyróżnić osobne funkcjonalne bloki, które realizują określone podzadania. Razem tworzą one współpracujący zespół, przetwarzający dane podobnie jak taśma produkcyjna. Można to określić jako potok przetwarzania danych, którego ogólny, wysokopoziomowy schemat został zobrazowany na Rys. 4.1.

Zwięzły opis struktury – „potok przetwarzania detekcji obiektów" (ang. *object detection pipeline*) – można znaleźć w pracy Boczkowskiego [7]. Opisano tam poszczególne etapy przetwarzania i rozwiązania stosowane na każdym z nich. Między innymi, obok kręgosłupa i głowicy, Boczkowski wyróżnia również „szyję", tj. opcjonalny etap reorganizacji wyekstrahowanych cech, np. poprzez budowę piramidy cech głębokich.

4.3.2 Kręgosłup detektora obiektów

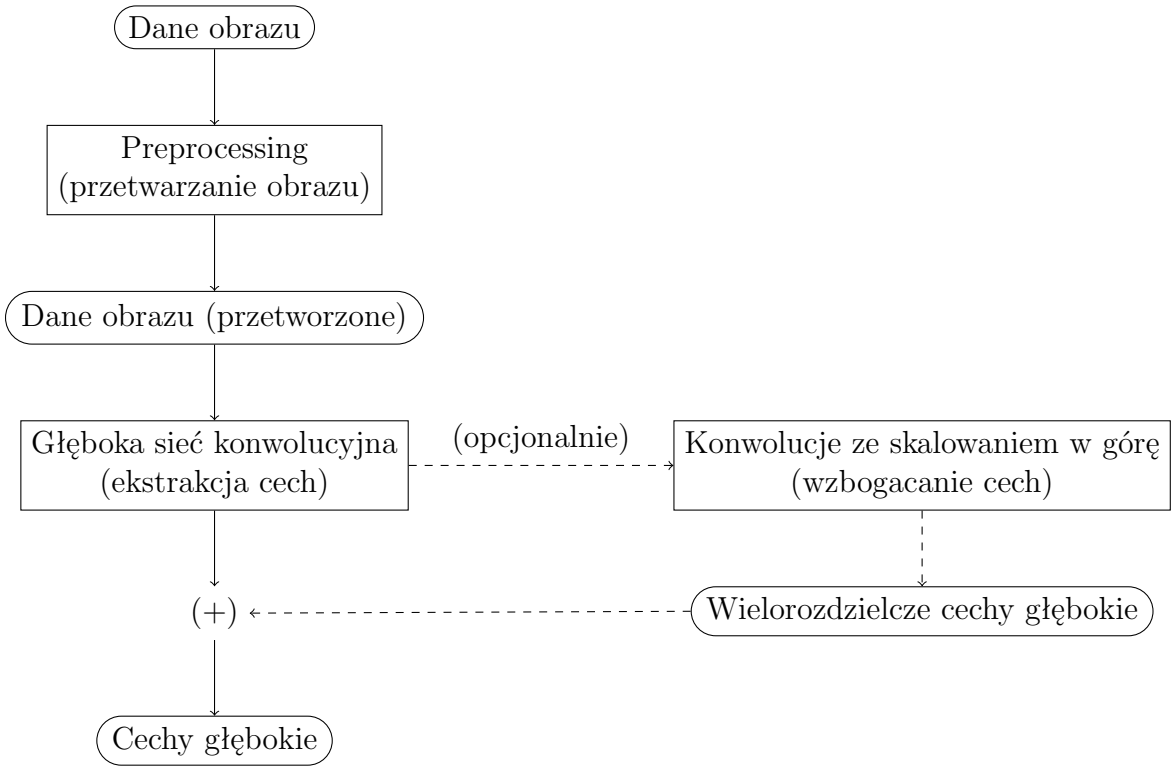
Podobnie jak klasyfikacja obrazów, detekcja obiektów rozpoczyna się ekstrakcją cech. Część modelu odpowiedzialna za ekstrakcję cech głębokich jest określana mianem



Rys. 4.1: **Struktura detektora obiektów w uczeniu głębokim.**
W strukturze detektorów obiektów opartym na głębokich sieciach splotowych można wyróżnić ekstraktor cech („kręgosłup") oraz część odpowiedzialną za lokalizację i klasyfikację obiektów („głowica"). Prostokąty na rysunku oznaczają bloki funkcjonalne, a zaokrąglone bloczki reprezentują dane wejściowe i wyjściowe. Podwójną strzałką oznaczono strumień zmiennej liczby wyników detekcji obiektów.

kręgosłupa. Ta część przyjmuje na wejściu obraz, a zwraca cechy głębokie, tj. wektor liczb lub dwuwymiarową siatkę wielowymiarowych wektorów (mapę cech), w których każda pozycja (wymiar) odpowiada za jakąś abstrakcyjną cechę, wyznaczoną w procesie trenowania modelu. Diagram pokazujący strukturę kręgosłupa, z opcjonalną ścieżką wzbogacania cech do postaci wielorozdzielczej piramidy, został przestawiony na Rys. 4.2.

Wzbogacaniem tym jest budowanie piramidy cech, do którego stosuje się fuzję cech (ang. *feature fusion*), tj. powiązanie wysokopoziomowych cech głębokich (o dużym poziomie abstrakcji) z bardziej precyzyjnymi przestrzennie cechami niskopoziomowymi (kopiowanymi z wcześniejszych, płytszych warstw sieci splotowej). Dołączanie (poprzez punktowe sumowanie) cech z wybranych płytszych warstw jest połączone ze skalowaniem w górę odpowiedzi z warstw głębszych. Operacja ta jest powtarzana kilkakrotnie, a każdy pośredni wynik jest zwracany do dalszego przetwarzania i dlatego przy tym rozwiązaniu kręgosłup nie zwraca pojedynczej mapy cech, ale zbiór map o rosnącym wymiarze przestrzennym (piramidę). Piramida cech głębokich została zaproponowana i opisana w pracy Lina [57], gdzie na kręgosłup z tym rozwiązaniem użyto określenia „sieć piramidy cech" (ang. *feature pyramid network*, FPN). W niniejszej pracy zastosowano do modeli, w których obecne jest to rozwiązanie, określenie „model [korzystający] z FPN", a do modeli zwracających pojedynczą mapę cech – „model bez



Rys. 4.2: **Kręgosłup głębokiego modelu detekcji obiektów.**
Oznaczenia: prostokąty – elementy przetwarzające, prostokąty zaokrąglone – dane, (+) – punktowe (ang. *element-wise*) sumowanie map cech z piramidą cech głębokich.

FPN" lub „model niekorzystający z FPN".

Opcjonalnie w detekcji obiektów występuje przetwarzaniem wstępne obrazu wejściowego. Zwykle na tym etapie dokonuje się dopasowania rozmiarów obrazu do rozmiaru wejścia sieci – może to być skalowanie (ang. *resize*) albo przycięcie (ang. *crop*). Wyjątkiem są sieci w pełni konwolucyjne (ang. *fully convolutional networks*, FCN), które przetwarzają cały obraz, niezależnie od wielkości. W tego rodzaju sieciach obraz wejściowy może mieć zmienny rozmiar, wyjście modelu przyjmuje rozmiar zależny od rozmiaru wejścia, ale za to ze złożonością obliczeniową proporcjonalną do pola powierzchni obrazu (liczby pikseli). Pierwszym zastosowaniem sieci FCN była segmentacja semantyczna [62,69], która polega na klasyfikacji poszczególnych pikseli, a wynikiem jest gęsta mapa klas obiektów lub wielokanałowy obraz prawdopodobieństw poszczególnych klas. Taki model może łatwo zostać zamieniony na klasyfikator obrazów – wystarczy dodać warstwę globalnego uśredniania wszystkich cech (ang. *global average pooling*, GAP) i w ten sposób powstaje klasyfikator o dynamicznym rozmiarze obrazu wejściowego. W detekcji obiektów np. sieć YOLOv2 [73] jest w całości siecią w pełni konwolucyjną, dotyczy to zarówno etapu ekstrakcji cech, jak i głowica detekcji.

Częstą operacją w przetwarzaniu wstępnym jest również *normalizacja* wartości próbek (składowych piksela), np. do przedziału $[0, 1]$, $[-\frac{1}{2}, \frac{1}{2}]$ albo $[-1, 1]$. Dokonuje się to przez odjęcie średniej wartości i podzielenie przez amplitudę (obie wartości wyznaczane

są empirycznie na zbiorze uczącym), albo odjęcie połowy zakresu (128) i podzielenie przez rozmiar zakresu (256), co jest deterministyczne, ale może być mniej pożądane (np. z uwagi na brak symetrii rozkładu wyniku przekształcenia), kiedy wartości występujące w rzeczywistych obrazach są z węższego przedziału, np. [30, 180].

W następnej kolejności wstępnie przetworzony obraz podlega ekstrakcji cech z użyciem głębokiej konwolucyjnej sieci neuronowej. Obok wspomnianego wcześniej określenia „kręgosłup" używany jest dla tej sieci także termin „ciało" (ang. *body*) (np. w rodzinie modeli YOLO [72]).

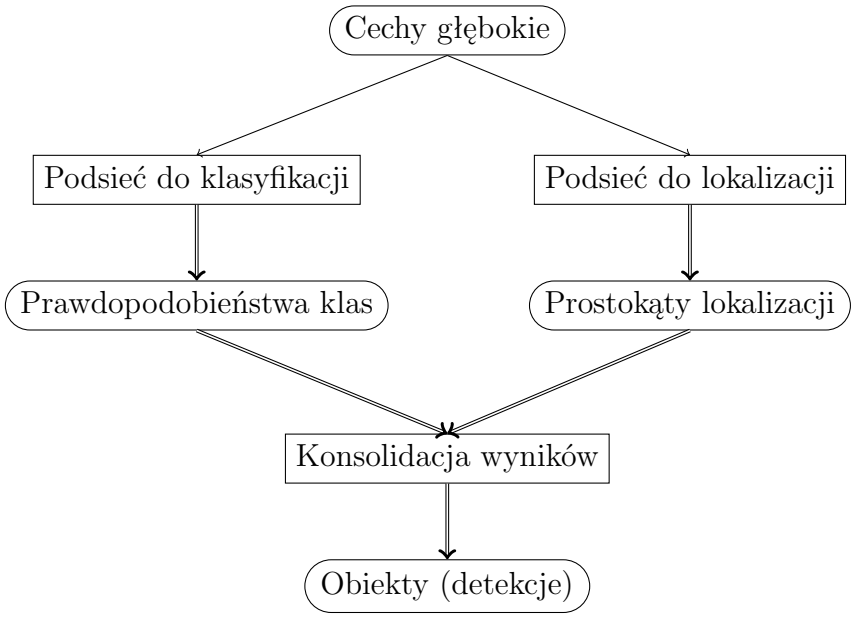
Wczesne modele używały w tym miejscu albo dedykowanych sieci splotowych (np. model OverFeat [81]), albo (używanych w tym czasie do klasyfikacji obrazów) wariantów sieci VGG. Na przykład sieć VGG-16 [83], która była zwycięskim modelem w konkursie ILSVRC (w edycji 2014), znalazła zastosowanie w pierwszych modelach R-CNN. W nowszych architekturach stosuje się do ekstrakcji cech głębokie sieci takie jak: ResNet [36], ResNeXt [104], DarkNet [7], a także (oferujące niższą złożoność obliczeniową) rodziny modeli: MobileNet [40], EfficientNet [88] i inne.

Wybór sieci do ekstrakcji cech może być podyktowany wymaganiami skuteczności detekcji i wydajności w warunkach ograniczonych zasobów obliczeniowych. Dwie, ostatnie wśród wymienionych, rodziny modeli oferują serie modeli o zróżnicowanych parametrach: mniejsze, o niższej skuteczności, i większe, o wyższej (w pracach im poświęconych zastosowanym kryterium skuteczności jest dokładność klasyfikacji).

W ekstrakcji cech może również być pomocne zastosowanie alternatywnych wersji operacji splotu (konwolucji) w sieci neuronowej. Na przykład konwolucje rozszerzone (ang. *dilated convolution*) [106] pozwalają na zwiększenie przestrzennego rozmiaru pola receptywnego, bez zwiększenia liczby parametrów modelu. Liczba wejściowych cech może być nadal 9 (jak w konwolucji z jądrem 3×3), ale obszar z którego są one brane może mieć rozmiar 5×5 (dylatacja o jeden piksel), 7×7 (dylatacja o dwa piksele) itp. Innym, silniejszym uogólnieniem operacji splotu w sieci są konwolucje deformowalne (ang. *deformable convolution*) [108]. Oprócz zwiększenia obszaru receptywnego kolejnej warstwy, pozwalają one na zmianę kształtu: dla każdego punktu z pola receptywnego uczone są wektory przesunięć (offsety), które modyfikują w dowolnym kierunku punkt, z którego pobierana jest cecha do operacji splotu. Ponieważ po przesunięciu współrzędne nie są już całkowitymi indeksami w tensorze cech, do wyznaczenia wartości cech stosuje się interpolację dwuliniową [108].

4.3.3 Detektor jednoetapowy

Ten rodzaj architektur dokonuje detekcji bezpośrednio na cechach wyjściowych z kręgosłupa (ekstraktora cech). Głowica takiego detektora może mieć dwie osobne podsieci konwolucyjne: pierwszą do klasyfikacji obiektów (wyznaczanie prawdopodobieństwa klas, a czasem samej detekcji – klasyfikacja binarna obiekt/tło), a drugą do generowania prostokątów zawierających obiekty, która zwraca dla każdego obiektu klasyfikowanego przez pierwszą sieć współrzędne odpowiedniego prostokąta (zadanie regresji). Taka budowa głowicy dla detektora jednoetapowego, która jest przedstawiona na Rys. 4.3, została zastosowana m.in. w detektorze RetinaNet [58].



Rys. 4.3: **Głowica detektora jednoetapowego.**
Detektory, które zawierają głowicę jednoetapową, są określane również jako detektory gęste. Osobne ścieżki przetwarzania do lokalizacji i klasyfikacji obiektów korzystają z tych samych cech głębokich.
Taka struktura występuje m.in. w RetinaNet [58], alternatywnie może być użyta jedna wielozadaniowa sieć neuronowa, jak w modelach z rodziny YOLO [72–74], gdzie za różne zadania odpowiedzialne są grupy kanałów wyjściowej warstwy sieci.
Znaczenie kształtów: prostokąty – elementy przetwarzające, prostokąty zaokrąglone – dane.
Znaczenie strzałek: pojedyncze – stały strumień danych dla jednego obrazu, podwójne – zmienna liczba danych, zależna od liczby obiektów.

Detekcja jednoetapowa jest również nazywana detekcją *jednostrzałową* (ang. *single-shot detection*) [61] albo detekcją gęstą [58], w odróżnieniu od dwuetapowej, która jest *rzadka*. Pierwsza z tych nazw bierze się z tego, że przetwarzanie obrazu (ekstrakcja cech) jest wykonywana tylko raz w procesie detekcji [7], co nie było regułą we wcześniejszych modelach dwuetapowych. Druga nazwa, detekcja gęsta[2], pochodzi od tego, że każdy punkt (wektor cech) w dwuwymiarowej tablicy cech (ang. *feature map*), może być użyty do detekcji pewnej ustalonej liczby obiektów. Takiemu punktowi mapy cech odpowiada na obrazie kwadratowy obszar, i ten obszar może być określany jako *komórka* obrazu. Liczba obiektów wykrywanych w takiej komórce jest hiperparametrem głowicy – dla każdego obiektu musi być przewidziana struktura opisująca detekcje (współrzędne prostokąta, wektor prawdopodobieństw klas).

Lokalizacja i klasyfikacja mogą być realizowane przez osobne podsieci, tak jak w Re-

[2]Sama sieć RetinaNet nawiązuje nazwą do siatkówki oka, tj. błony położonej na dnie oka, gęsto pokrytej neuronami fotoreceptorowymi. Ta analogia nazewnicza jest stosowana również dla wyświetlaczy typu *retina*, które charakteryzuje wysoka gęstość upakowania pikseli.

tinaNet [58], albo przez jedną w pełni konwolucyjną, tzn. niekorzystającą z warstw gęstych (ang. *dense, fully connected*), sieć neuronową, tak jak w sieci YOLOv2 [73].

Kotwice. Kotwica, tj. potencjalny obiekt w warstwie wyjściowej detektora, jest to model prostokąta obiektu *a priori* (ang. *box prior*). Na etapie treningu modelu są do niego dopasowywane obiekty zbioru uczącego o określonych cechach (rozmiar, stosunek wysokości do szerokości). W czasie wykonania modelu (inferencji) właśnie takie obiekty będą wykrywane przez neurony warstwy wyjściowej przypisane do tej konkretnej kotwicy.

Z użyciem kotwic wiąże się zmiana sposobu regresji współrzędnych prostokąta obiektu – zamiast bezpośrednio wyznaczać współrzędne x, y, w, h, sieć wyznacza parametry przekształcenia prostokąta kotwicy t_x, t_y, t_w, t_h. Współrzędne prostokąta określają następujące równania (m.in. w sieci YOLOv2 i YOLOv3 [73, 74]):

$$x = c_x + t_x \tag{4.1}$$
$$y = c_y + t_y \tag{4.2}$$
$$w = p_w \cdot t_w \tag{4.3}$$
$$h = p_h \cdot t_h \tag{4.4}$$

Znaczenie zmiennych:

- x, y – współrzędne środka prostokąta obiektu,
- w, h – szerokość i wysokość prostokąta obiektu,
- c_x, c_y – współrzędne lewego górnego rogu komórki obrazu, w której wykrywany jest obiekt,
- p_w, p_h – szerokość i wysokość prostokąta kotwicy.

Użycie kotwic nie gwarantuje poprawy wyników detekcji, ale pozwala na specjalizację określonych części sieci do wykrywania specyficznych obiektów i pozwala na dokładniejsze wyznaczanie prostokątów obiektów.

Wyznaczenie rozmiarów poszczególnych prostokątów kotwic może być wykonywane albo arbitralnie (np. RetinaNet [58] korzysta z trzech kotwic o proporcjach 1:2, 1:1 i 2:1), albo empirycznie na podstawie analizy statystycznej zbioru uczącego (np. jedna z konfiguracji YOLOv2 [73] używa pięciu kotwic, o proporcjach: 26.1:41.5, 57.3:108.8, 100.4:225.4, 143.8:416.2 i 370.8:383.4). Analiza ta ma następujący przebieg: liczba kotwic jest ustalana *a priori*, a następnie obiekty są grupowane na taką liczbę grup algorytmem k-średnich. Wynikowe rozmiary kotwic to przeciętne rozmiary prostokątów w każdej grupie.

Użycie kotwic dotyczy również detektorów dwuetapowych (szczególnie w etapie pierwszym) i zostało wprowadzone m.in. w sieci Faster R-CNN [76].

Zapobieganie powtórzeniom obiektów. Przy jednoetapowej detekcji możliwe jest zjawisko wielokrotnego wykrywania obiektów (np. przez różne kotwice w tej samej komórce, tj. wektorze cech mapy wyjściowej). Wówczas potrzebny jest mechanizm eliminacji takich powtórzeń (na etapie konsolidacji wyników), np. za pomocą usuwania obserwacji nie-maksymalnych (ang. *non-maximum suppression*, NMS). Polega ono na przetworzeniu wyników detekcji obiektów w kolejności malejącej ufności i usuwaniu tych prostokątów obiektów, które mają dużą powierzchnię przecięcia z innymi, rozpatrzonymi wcześniej, obiektami o wyższym prawdopodobieństwie. Technika ta jest stosowana w wielu detektorach, m.in. w YOLOv2 i YOLOv3 [73, 74].

Alternatywą dla NMS jest grupowanie prostokątów (ang. *box pooling*) i wyznaczanie uśrednionego prostokąta dla danej grupy (każda grupa zwróconych prostokątów, powinien odpowiadać jednemu obiektowi na obrazie). Do grupowania potrzebna jest pewna miara podobieństwa albo odległości prostokątów, oraz rozwiązanie problemu liczby obiektów w obrazie (dla niektórych algorytmów, np. k-średnich, potrzebne jest przyjęcie *a priori* założenia o tej liczbie). W efekcie, zastosowanie tego podejścia daje dobre efekty w prostych sytuacjach (jeden obiekt na obrazie, znana z góry liczba obiektów do wykrycia) – przykładem jest klasyfikacja obrazów *ikonicznych*, tj. zawierających jeden obiekt. Taka klasyfikacja z lokalizacją jest prostszym zadaniem niż detekcja obiektów (która jednocześnie wykrywa wiele obiektów różnych klas), ale jest szczególnym przypadkiem, ponieważ może być rozwiązywana bez modyfikacji przez modele detekcji obiektów. Przykładem zastosowania grupowania prostokątów detekcji jest model OverFeat [81].

4.3.4 Detektor dwuetapowy

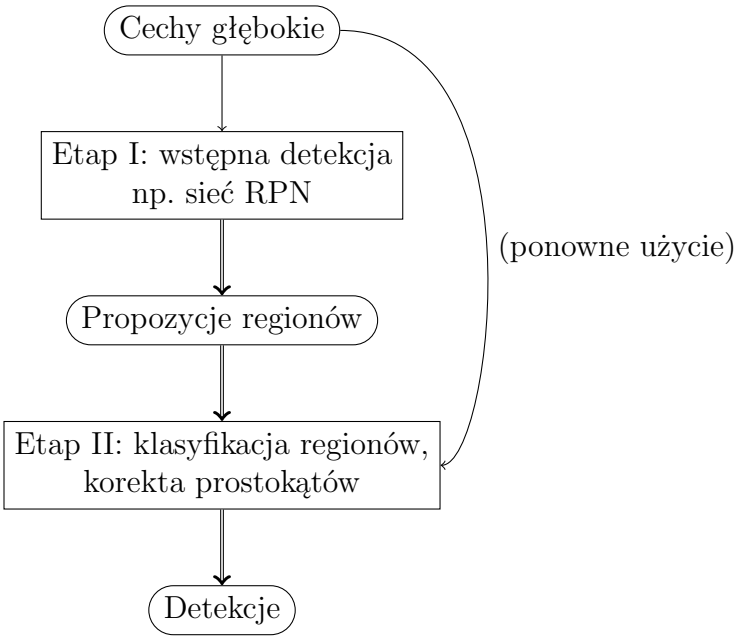
Modele detekcji obiektów realizujące to zadanie dwuetapowo, jak zostało to przedstawione na Rys. 4.4, mają w swojej strukturze rozdzielone dwa bloki funkcjonalne:

1. Mechanizm proponowania regionów, który zwraca prostokąty (lokalizacje), tzw. propozycje regionów (ang. *region proposal*), w których na obrazie może występować obiekt dowolnej z wykrywanych klas; mechanizm ten nie musi być realizowany jako sieć neuronowa, ale kiedy jest to CNN, to cały blok jest określany jako sieć propozycji regionów (ang. *region proposal network*, RPN).

2. Głowica przetwarzająca propozycje regionów (ang. *ROI head*), która odrzuca je lub zwraca obiekty detekcji.

Detekcja dwuetapowa jest również nazywana detekcją rzadką, ponieważ klasyfikacja i lokalizacja są wykonywane tylko na wybranych regionach, które zostały wygenerowane przez pierwszy etap, tj. na propozycjach regionów. Dzięki temu nie przetwarza się w drugim etapie całego obrazu lub zbioru wszystkich wyekstrahowanych cech.

Obecnie do realizacji pierwszego etapu stosowana jest głęboka sieć neuronowa, która jest określana jako sieć propozycji regionów, RPN. Nie jest to jednak jedyne możliwe podejście, wcześniej w tym etapie stosowano inne metody, nawet nie związane z uczeniem maszynowym, np. algorytm widzenia maszynowego *selective search*, który został



Rys. 4.4: **Głowica dwuetapowego detektora obiektów.**
Przedstawiono przypadek, kiedy oba etapy detekcji korzystają z cech głębokich; możliwe jest również użycie innych mechanizmów propozycji regionów niż sieci głębokie, wówczas danymi wejściowymi do etapu I są surowe dane obrazu.
Propozycje regionów razem z cechami głębokimi stanowią dane wejściowe do etapu II, w którym każda propozycja może zostać odrzucona lub przyjęta.
Znaczenie strzałek: pojedyncze – stały strumień danych dla jednego obrazu, podwójne – zmienna liczba danych, zależna od liczby obiektów.

użyty w pierwszym detektorze R-CNN [27]. W związku z tym, R-CNN nie jest jako całość modelem głębokim, ale stanowi podejście mieszane (inaczej: hybrydowe), pomiędzy uczeniem głębokim i innymi metodami.

Pierwszy etap może również być traktowany jako prosty detektor jednoetapowy (gęsty), który wykrywa obiekty na wyekstrahowanych cechach głębokich albo bezpośrednio na danych obrazu (jeżeli posiada wbudowaną ekstrakcję cech). Ta detekcja nie obejmuje klasyfikacji obiektów, ale, oprócz współrzędnych prostokątów zawierających obiekt, każda propozycja zawiera również prawdopodobieństwo (ufność modelu) określaną po angielsku zgrabnym rzeczownikiem *objectness* (dosł. „obiektowość", fakt bycia obiektem), który jest neologizmem spotykanym w pracach na temat dwuetapowej detekcji obiektów oraz niektórych opracowaniach dotyczących detekcji jednoetapowej (np. YOLO [73]). Taki „wskaźnik obiektowości" pozwala na zastosowanie progu przy wykonywaniu detekcji w taki sposób, aby odrzucać mniej prawdopodobne propozycje, co upraszcza obliczenia w etapie drugim.

Ponieważ w drugim etapie będą analizowane wyłącznie regiony wygenerowane przez etap pierwszy, złożoność obliczeniowa rzadkiej detekcji obiektów jest zmienna, zależna od liczby znalezionych propozycji obiektów, a ta zależy od zawartości obrazu. W ten

sposób można taką procedurę zakwalifikować pod względem złożoności obliczeniowej jako *wrażliwą na dane wejściowe*. Związek danych wyjściowych ze złożonością obliczeniową jest bardziej skomplikowany, ponieważ drugi etap może odrzucić większość lub wszystkie propozycje obiektów, ale wszystkie zostaną przetworzone. Jest to różnica w stosunku do detekcji gęstej, która zazwyczaj ma w przybliżeniu stałą złożoność obliczeniową (dla stałego rozmiaru wejścia sieci; dla w pełni konwolucyjnych sieci, które „skanują" cały obraz będzie ona zależna od rozmiarów obrazu, ale nie od zawartości).

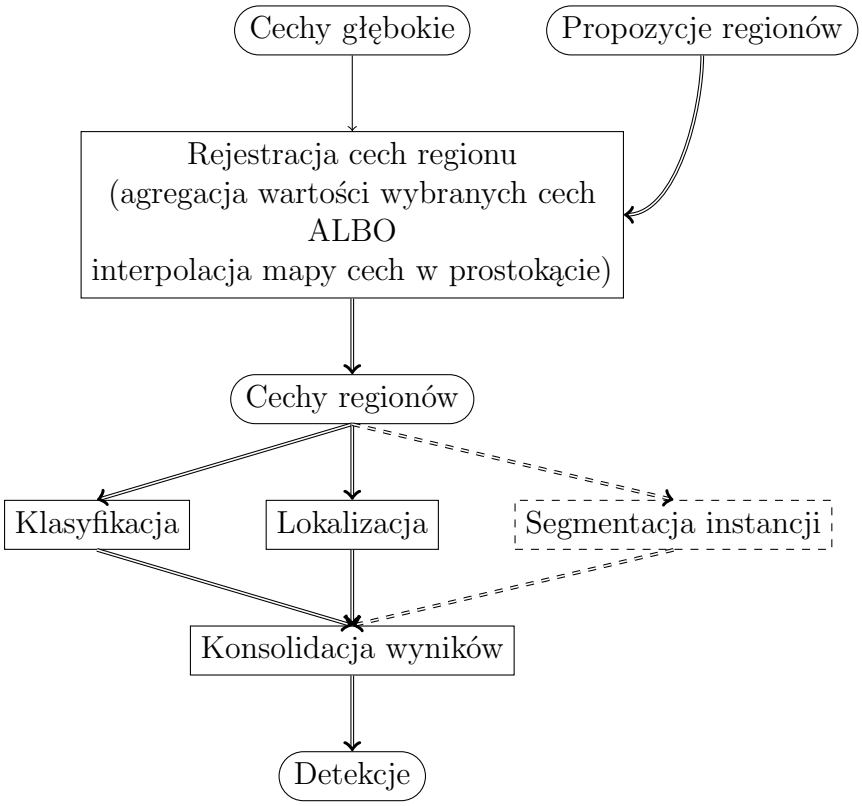
Specjalizacja etapów. Podział na etapy wiąże się z rozdziałem zadań i kryteriów jakości dla każdego z etapów. Najważniejszą cechą etapu pierwszego jest wysoka czułość (ang. *sensitivity, recall*), tj. znalezienie jak największej liczby obiektów obecnych na obrazie. Drugi etap powinien cechować się dużą precyzją (ang. *precision*), jeśli chodzi o akceptowanie propozycji obiektów, a także dużą dokładnością ich klasyfikacji. Ponieważ kryteria te są rozdzielone, to taki kaskadowy model osiąga dobre rezultaty, ale za cenę większego zapotrzebowania na moc obliczeniową, na co zwracają uwagę autorzy metod jednoetapowych [61, 72]. Można tu dostrzec pewną analogię do znanej z inżynierii oprogramowania zasady pojedynczej odpowiedzialności (ang. *single responsibility principle*), która wyraża się w powiedzeniu *do one thing, do it well* („robić jedną rzecz, ale dobrze").

Detektor jednoetapowy powinien optymalizować oba te kryteria (czułość i precyzję detekcji) w ramach jednolitego modelu, a ostatecznie kompromisem pomiędzy nimi steruje użytkownik dostosowując próg ufności (*confidence score threshold*) detekcji. Intuicyjnie sugeruje to, że dla modeli jednoetapowych próg ufności może mieć szczególnie ważne znaczenie.

Struktura etapu II. Schemat budowy etapu II detektora rzadkiego jest przedstawiony na Rys. 4.5. Zastosowane tam określenie *rejestracja* cech regionu nawiązuje do pojęcia rejestracji w przetwarzaniu obrazu i widzeniu maszynowym – chodzi w niej o przesunięcie i przycięcie obrazu w taki sposób, aby nadawał się do dalszego przetwarzania. W przypadku detektorów dwuetapowych obejmuje to dwa kroki:

- projekcję prostokąta propozycji obiektów na mapę cech głębokich,
- agregację cech objętych obrazem prostokąta na mapie cech do wektora cech wejściowych, które stanowią wejście do klasyfikatora obiektu i lokalizatora (regresora współrzędnych prostokąta wyjściowego) – tzw. głowicy regionu.

Jako *głowicę regionu* można określić funkcjonalny fragment sieci neuronowej, który przetwarza propozycje regionów (prostokąty), wygenerowane przez sieć propozycji regionów i cechy regionu (cechy głębokie obrazu, wyekstrahowane przez kręgosłup, a następnie wybrane dla konkretnej propozycji regionu) zwracając finalne detekcje. Zadaniami głowicy są: klasyfikacja, lokalizacja i, dla detektorów, które dodatkowo dokonują segmentacji instancji, generowanie masek obiektów.



Rys. 4.5: **Etap II rzadkiego detektora obiektów.**
Rejestracja cech regionów dla każdego regionu (propozycji obiektu) zwraca wybrane lub przekształcone cechy głębokie, specyficzne dla danego regionu (cechy regionu). Następnie, w głowicy detektora dwuetapowego można wyróżnić blok funkcjonalny, który przetwarza propozycje regionów i cechy regionów na detekcje obiektów. Detektory dwuetapowe mogą w głowicy regionu realizować dodatkowo zadanie segmentacji instancji, tj. generować („rysować") maskę obiektu – zbiór pikseli albo wielokąt wskazujący położenie obiektu dokładniej niż jego prostokąt zawierający. Znaczenie strzałek: pojedyncze – stały strumień danych dla jednego obrazu, podwójne – zmienna liczba danych, zależna od liczby obiektów.

Jeżeli mechanizm propozycji regionów działa właściwie, to obszar wskazany przez propozycję stanowi ikoniczny obraz pewnego obiektu, który może być przetwarzany za pomocą „klasyfikacji obrazu z wskazaniem prostokąta zawierającego", w taki sposób jak w sieci OverFeat.

Pojedyncza ekstrakcja cech głębokich. Wczesne podejścia dwuetapowe, takie jak R-CNN [27], nie korzystały ani z sieci propozycji regionów RPN, ani z rejestracji cech przypisanych do regionu (ang. *ROI pooling*). Aby pokazać, jak istotna jest obecność tych elementów, trzeba dokładniej przyjrzeć się oryginalnemu modelowi R-CNN. Model ten generował propozycje regionów z użyciem algorytmu *selective search* (statystyczne grupowanie pikseli w obiekty, algorytm klasycznego widzenia maszynowego)

i dla każdej z tych propozycji wykonywana była ekstrakcja cech głębokich za pomocą konwolucyjnej sieci VGG-16 [83] (metoda uczenia głębokiego), a wektor tych cech był wejściem do klasyfikatora SVM (metoda klasycznego uczenia maszynowego).

Każdy z elementów tego potoku przetwarzania musiał być osobno trenowany, ale najbardziej kosztowna była wielokrotna ekstrakcja cech głębokich – nie jeden raz dla całego obrazu, ale dla każdej propozycji regionu (a tych może być kilkadziesiąt lub kilkaset). Każda propozycja regionu była wyodrębniana z obrazu (wycięta), następnie dopasowywana do wejścia sieci (skalowana, być może z różnymi skalami w pionie i w poziomie) i podawana na wejście sieci neuronowej.

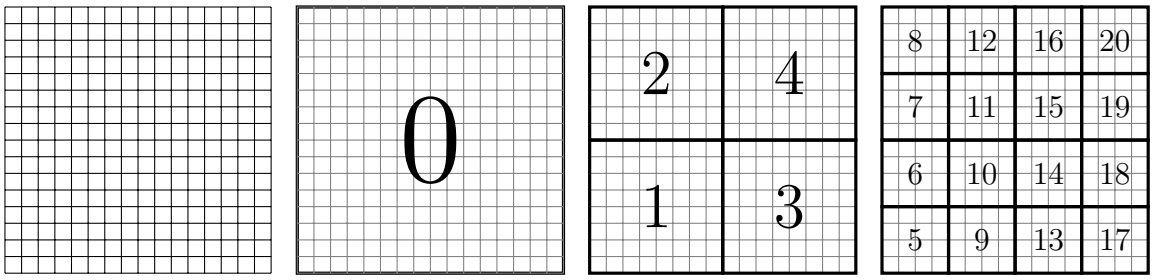
Jedyna możliwość optymalizacji takiego podejścia polegała na zrównolegleniu poprzez grupowanie obrazów we *wsady* (ang. *batch*), tj. paczki o określonej liczbie elementów, zgrupowane w wielowymiarowej tablicy, które są przetwarzane jednocześnie dzięki dużej liczbie jednostek wykonawczych dostępnych w akceleratorze GPU. Ten problem nie występuje w detektorach jednoetapowych, co podkreślają nazwy takich architektur:

- YOLO [72] (ang. *you only look once*), dosł. „patrzysz tylko raz".
- SSD [61] (ang. *single-shot detector*), dosł. „detektor jednostrzałowy".

Pierwotne znaczenie odnosi się właśnie do ekstrakcji cech głębokich, co jest kosztowną operacją ze względu na dużą liczbę mnożeń z akumulacją (ang. *multiply-accumulate*) liczb zmiennoprzecinkowych.

Rozwiązanie tego problemu w detektorach dwuetapowych przyniosły dwa modele: SPP-net [35], a następnie Fast R-CNN [26], który korzysta z idei *przestrzennej piramidy cech* (ang. *spatial pyramid pooling*, SPP) do radykalnego ulepszenia detekcji dwuetapowej. Przekształcenie SPP było stosowane przed upowszechnieniem się uczenia głębokiego (uczenia cech) również do cech i deskryptorów tworzonych ręcznie, metodami widzenia maszynowego, np. w algorytmie SPM (ang. *spatial pyramid matching*) [53]. Autorzy SPP-net wykorzystują tę operację, wyjaśnioną na Rys. 4.6, do uniezależnienia modeli klasyfikacji obrazów i detekcji obiektów od rozmiaru obrazu wejściowego. Cechy głębokie wyekstrahowane np. siecią w pełni konwolucyjną mogą mieć zmienny rozmiar, ale po przejściu przez warstwę przestrzennej piramidy cech przyjmują postać wektora cech o stałym rozmiarze, który jest dalej przekazywany na wejście głowicy klasyfikacyjnej lub detekcyjnej.

Począwszy od modelu Fast R-CNN, detektory rzadkie również wykonują tylko pojedynczą ekstrakcję cech głębokich przez sieć neuronową, którą w oryginalnym modelu Fast R-CNN jest VGG-16. Kluczową innowacją dla tej optymalizacji wydajności jest operacja *poolingu* (dosł. „tworzenia puli"), to jest ustalenia (zwykle redukcji) wymiaru cech. W dziedzinie baz danych jest to określane mianem agregacji lub funkcji agregujących (w połączeniu z klauzulą `GROUP BY` pozwalającą na wyznaczenie *agregatu* wartości na podstawie podzbioru; przykłady to `SUM`, `AVG`, `MIN`, `MAX` itp.). W konwolucyjnych sieciach neuronowych stosuje się agregację przede wszystkim z użyciem dwóch operacji: maksimum (ang. *max pooling*) oraz średniej arytmetycznej (ang. *average pooling*).



Rys. 4.6: **Piramida przestrzenna cech.**
Agregacja (ang. *pooling*) cech za pomocą piramidy przestrzennej pozwala na redukcję dwuwymiarowego zbioru cech (siatki liczb) o dowolnym rozmiarze do wektora o ustalonym rozmiarze.
Na rysunku przedstawiono przykład dla kwadratowej siatki zredukowanej do 21-elementowego wektora cech za pomocą trzypiętrowej przestrzennej piramidy cech – jeżeli do *poolingu* zastosujemy średnią arytmetyczną, to zerowy indeks wektora będzie zawierał średnią wszystkich cech, indeksy 1–4 będą zawierały średnie każdej ćwiartki, a indeksy 5–20 średnie cech w kwadratach o polu równym $\frac{1}{16}$ pola siatki wejściowej.

Kolejnym krokiem w rozwoju były sieci z rodziny Faster R-CNN [76], a także ich kolejne wersje [77], które ulepszają działanie modeli (np. ulepszenia kręgosłupa przez zastosowanie ekstrakcji cech przez ResNeXt i budowania piramidy cech przez FPN).

Sieć propozycji regionów. Najważniejszym czynnikiem, który odróżnia Faster R-CNN od Fast R-CNN, jest zastosowanie sieci propozycji regionów RPN. Przyspieszenie wynika z tego, że ta sieć korzysta dokładnie z tych samych cech głębokich co etap drugi – te same cechy głębokie służą do generacji propozycji, a następnie są poddawane rejestracji dla każdej z tych propozycji i w tej postaci zostają użyte do ostatecznej detekcji obiektów.

Upraszczanie i ujednolicanie modeli do pojedynczej sieci neuronowej nie jest w procesie rozwijania metod detekcji obiektu procesem jednokierunkowym. Przykładem hybrydowego podejścia, które jest późniejsze niż jednolity model Faster R-CNN, jest praca dotycząca sieci propozycji regionów ERPN (ang. *enhanced RPN*) [11]. Autorzy proponują nowy mechanizm propozycji regionów, który oprócz głębokiej sieci neuronowej do propozycji regionów wykorzystuje klasyfikator SVM, którego hiperparametry były optymalizowane z użyciem metaheurystyki roju cząstek (ang. *particle swarm optimization*, PSO). Klasyfikator ten, określany jako PSO-SVM, jest wytrenowany na wyekstrahowanych cechach głębokich i używany do odrzucania negatywnych propozycji regionów. Można takie rozwiązanie scharakteryzować jako mechanizm uwagi (ang. *attention*), który kieruje przetwarzanie sieci neuronowej w stronę bardziej obiecujących regionów, poprawiając precyzję. Dzięki szybkości działania SVM, a tym samym efektywnej filtracji propozycji regionów, uzyskano wysoką wydajność. Podejście to pokazuje, że metody hybrydowe (jak pierwszy model R-CNN) stanowią nadal aktualne podejście, które można stosować dla różnych zadań.

Segmentacja instancji. Następnym dwuetapowym modelem, ważnym w rozwoju metod detekcji obiektów, jest Mask R-CNN [34]. Jest to wielozadaniowa sieć, która oprócz detekcji obiektów realizuje również segmentację instancji. Mimo to, możliwe było uzyskanie rezultatów równie dobrych lub lepszych niż Faster R-CNN – model dedykowany do detekcji obiektów. Istotnym składnikiem tego sukcesu było zastąpienie operacji agregacji cech w prostokącie propozycji regionu (*ROI pooling*) przez dostosowanie cech wejściowych dla maski obiektu do położenia regionu (*ROI alignment*), które zachowuje rozmiar projekcji tego prostokąta na mapę cech głębokich. Ze względu na niecałkowite współrzędne prostokąta, konieczna jest interpolacja dwuliniowa wektorów cech, które są na granicy obszaru wyznaczonego przez projekcję propozycji regionu.

Następnie, wszystkie „punkty" (wektory cech) przechodzą następnie przez sieć w pełni konwolucyjną do klasyfikacji masek (ang. *mask FCN*), która klasyfikuje każdy piksel regionu obiektu do wykrywanych kategorii. Jest to klasyfikacja niezależna, tzn. wyjście sieci ma aktywację sigmoidalną dla każdej klasy – bez operacji *softmax*, która służy do klasyfikacji kategorii wyłączających się wzajemnie.

Ta prosta zmiana powoduje, że liczba tworzonych masek jest równa liczbie wykrywanych kategorii obiektów – część głowicy odpowiadająca za rysowanie masek zwraca maski dla każdej klasy, a na koniec wybierana jest ta maska (kanał wyjściowy FCN), która należy do kategorii, którą wskazał klasyfikator obiektów w sieci klasyfikującej (równoległy wobec podsieci FCN fragment głowicy II etapu). W ten sposób Mask R-CNN jest detektorem, który w całości pokrywa schemat II etapu przedstawiony na Rys. 4.5.

Na przykładzie Mask R-CNN można opisać segmentację instancji jako detekcję obiektów, a następnie semantyczną segmentację każdego z regionów i wybór maski właściwej dla klasy wykrytego obiektu. Prostokąt zawierający nie wymaga w tym przypadku sieci, która wykona regresję współrzędnych prostokąta, ponieważ wystarczy wyliczyć minimalny prostokąt zawierający maskę obiektu. W praktyce równoległa ścieżka sieci wyznacza ten prostokąt.

Segmentacja instancji jest trudnym zadaniem i skutecznie realizowana jest przede wszystkim w detektorach dwuetapowych, a w pracy o sieci YOLOv3 [74] (jednoetapowym detektorze) jej autor raportuje trudności, które napotkał przy próbie segmentacji instancji za pomocą tej sieci. Pozornym wyjątkiem od tej reguły jest praca na temat sieci RetinaMask [24], która opisuje „segmentację instancji z detekcją jednoetapową" lub „dodanie segmentacji instancji do detekcji jednoetapowej niskim kosztem". Jednak to, co rzeczywiście kryje się pod tymi twierdzeniami, to użycie sieci RetinaNet (która jest detektorem gęstym, jednoetapowym) do detekcji obiektów, a następnie po niej zastosowanie *ROI alignment* (jak w Mask R-CNN) i sieci FCN do segmentacji semantycznej regionu każdego zwróconego obiektu.

A zatem, jest to wariant detekcji dwuetapowej, w którym wszystko poza klasyfikacją pikseli maski jest wykonywane w etapie I. Nie można jednak tego nazwać „gęstą segmentacją instancji", ponieważ segmentacja poszczególnych instancji jest robiona w sposób rzadki. Niemniej jednak, sieć RetinaMask jest wartościowym osiągnięciem technicznym i inżynieryjnym, ponieważ korzysta z szybkiej detekcji zapewnionej przez RetinaNet i minimalizuje złożoność obliczeń w etapie II poprzez wykonywanie w nim

jedynie segmentacji pikseli w wykrytych prostokątach.

4.3.5 Trening detektorów obiektów

Trenowanie głębokich konwolucyjnych detektorów obiektów jest trudnym zadaniem [66]. Ważnym źródłem trudności jest niezrównoważenie (ang. *imbalance*) w danych uczących, które dotyczy wielu kwestii.

Jedną z nich jest rzadkość obiektów do wykrycia w stosunku do rozmiaru przestrzeni możliwych prostokątów na obrazie. Prostokąty pozbawione obiektów, czyli „tło" w obrazie, przewyższają liczbowo instancje obiektów o kilka rzędów wielkości. W przypadku sieci RetinaNet [58] zastosowano *skoncentrowaną* funkcję straty (ang. *focal loss*, FL), która jest rozszerzeniem funkcji entropii krzyżowej (ang. *cross-entropy*, CE).

Funkcja straty CE jest oparta na pojęciu entropii (w sensie Shannona) $H(X)$, która jest miarą nieuporządkowania dla dyskretnego rozkładu prawdopodobieństw $P(X) = \{p(x_i)\}$ na zbiorze $X = \{x_i\}$, który spełnia warunek:

$$\sum_{x_i \in X} p(x_i) = 1 \tag{4.5}$$

.

Entropia takiego rozkładu wynosi:

$$H(P) = -\sum_{x_i \in X} p(x_i) \log(x_i) \tag{4.6}$$

.

Do celów konstrukcji funkcji straty można użyć logarytmu o dowolnej podstawie – dwójkowego, naturalnego albo dziesiętnego. Kiedy dane są 2 rozkłady $P(X)$ i $Q(X)$, można wyznaczyć dla nich entropię krzyżową. Jest to asymetryczna operacja (kolejność argumentów istotna). Entropia krzyżowa rozkładu P *względem* rozkładu Q wyraża się wzorem:

$$H(P, Q) = -\sum_{x_i \in X} p(x_i) \log q(x_i) \tag{4.7}$$

Intuicyjnie, biorąc pod uwagę zachowanie funkcji logarytmicznej w przedziale $(0, 1]$, można zauważyć, że jest to miara podobieństwa tych rozkładów: wartości duże (bliskie 1) będą mnożone przez swój logarytm (bliski 0) i *vice versa* dla wartości bliskich 0. Dla rozkładów o dużych różnicach wystąpi sytuacja, kiedy $p(x_i)$ jest wysokie, a $q(x_i)$ niskie, co da w wyniku iloczyn dwóch dużych wartości.

Na podstawie entropii krzyżowej sformułowano funkcję straty CE, używaną nie tylko w uczeniu głębokim, ale również w klasycznym uczeniu maszynowym (regresja liniowa, regresja logistyczna). Przyjmując $y \in \{0, 1\}$ za wartość empiryczną (wzorcową), a $\hat{y} \in (0, 1)$ za odpowiedź modelu, tj. wartość ufności dla założenia $y = 1$, funkcję CE oblicza się według następującego wzoru:

$$\mathrm{CE}(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}) \tag{4.8}$$

Ta funkcja przyjmuje wysoką wartość zarówno dla błędu I i II rodzaju, tj. dla fałszywej klasyfikacji negatywnej (FN), jak i dla fałszywej klasyfikacji pozytywnej (FP).

Problem, na który zwracają uwagę autorzy RetinaNet, polega na tym, że minimalizowanie funkcji straty CE jest możliwe przez wykrywanie większej liczby obiektów, ale również przez zwiększanie ufności klasyfikacji negatywnej próbek „tła". Innymi słowy, trening modelu „koncentruje się" na łatwych przypadkach. Rozwiązaniem jest skoncentrowana funkcja straty FL, która ma wyższe wartości dla źle sklasyfikowanych obiektów trudnych. Aby zapisać tę funkcję w sposób zwięzły, można wprowadzić wielkość p_t, tj. ufność poprawnej klasyfikacji (niezależnie od tego, czy jest negatywna, czy pozytywna). Przedstawiając równanie 4.8 w następującej, alternatywnej postaci:

$$CE(y, \hat{y}) = \begin{cases} -\log \hat{y}, & \text{jeżeli } y = 1, \\ -\log(1 - \hat{y}), & \text{w przeciwnym wypadku,} \end{cases} \tag{4.9}$$

możemy zdefiniować wartość p_t następująco:

$$p_t = \begin{cases} \hat{y}, & \text{jeżeli } y = 1, \\ 1 - \hat{y}, & \text{w przeciwnym wypadku.} \end{cases} \tag{4.10}$$

Takie podstawienie upraszcza zapis funkcji CE:

$$CE(y, \hat{y}) = CE(p_t) = -\log p_t \tag{4.11}$$

Skoncentrowana funkcja straty FL modyfikuje funkcję CE przez pomnożenie przez czynnik zależny od wartości p_t i parametru γ:

$$FL(y, \hat{y}) = FL(p_t) = -(1 - p_t)^{\gamma} \log p_t \tag{4.12}$$

Dla parametru $\gamma = 0$ funkcja FL upraszcza się do funkcji CE. Działanie „koncentrujące" zachodzi dla $\gamma > 0$, sieć RetinaNet z kręgosłupem ResNet-50 została wytrenowana przy wartości parametru $\gamma = 2$.

Problem rozwiązany przez wprowadzenie funkcji straty FL jest określany jako nierównowaga tło-obiekt (ang. *background-foreground imbalance*). Użycie funkcji FL w innym jednoetapowym detektorze, YOLOv3 [74] nie przynosi poprawy skuteczności, ponieważ głowica w tym detektorze ma osobną wartość *objectness*, która jest mnożona przez wektor prawdopodobieństw klas przy ostatecznym określaniu ufności detekcji. Przypomina to mechanizm uwagi („atencji"), który jest stosowany m.in. w modelach uczenia głębokiego z dziedziny przetwarzania języka naturalnego. Zrównoważenie tło-obiekt jest ogólnym problemem w detekcji obiektów, metoda MMOD [48] była skutecznym rozwiązaniem tego problemu, które było lepsze niż stosowane wcześniej podpróbkowanie regionów tła przy trenowaniu modeli detekcji (co powodowało gorsze uogólnianie na różnorodne tła analizowanego obrazu).

Oprócz problemu zrównoważenia tło-obiekt jest również problem zrównoważenia obiekt-obiekt (ang. *foreground-foreground*), które wynika z nierównej proporcji obiektów różnych klas w zbiorze uczącym. Obiekty najczęstszych klas (np. obiekty klasy

osoba) stanowią znaczną część zbioru wszystkich obiektów, a na końcu rankingu znajduje się „długi ogon" (ang. *long tail*) klas, które mają po kilka wystąpień. Tego rodzaju rozkład częstości jest określany jako rozkład Zipfa [109], który jest spotykany w wielu innych sytuacjach, np. w rozkładzie częstości słów używanych w konkretnym języku naturalnym.

4.4 Ocena skuteczności detekcji obiektów

Podstawowe pojęcia. Aby precyzyjnie opisać proces oceny skuteczności modelu realizującego detekcję obiektów potrzebne są odpowiednie pojęcia. Na przykład używając określenia *obiekt* trudno może być wskazać, czy chodzi o to, co jest widoczne na obrazie, czy o metadane instancji obiektu w zbiorze testowym, czy o reprezentację wykrytego obiektu, jaką zwraca model. Dlatego, wprowadzone tu zostaną nazwy i skrótowce, które jednoznacznie określą kryjące się pod nimi znaczenie.

Fragment obrazu, który przedstawia jakiś obiekt, to *obiekt do wykrycia* (ODW). Jest to zbiór pikseli, które stanowią wizualną reprezentację np. osoby, zwierzęcia lub jakiegoś nieożywionego przedmiotu, który nie musi być widoczny w całości, ani nawet nie musi stanowić spójnego obszaru (np. kiedy fragmenty są przesłonięte). ODW musi jednak być pojedynczym egzemplarzem („jedną sztuką", ang. *instance*) danej klasy obiektów.

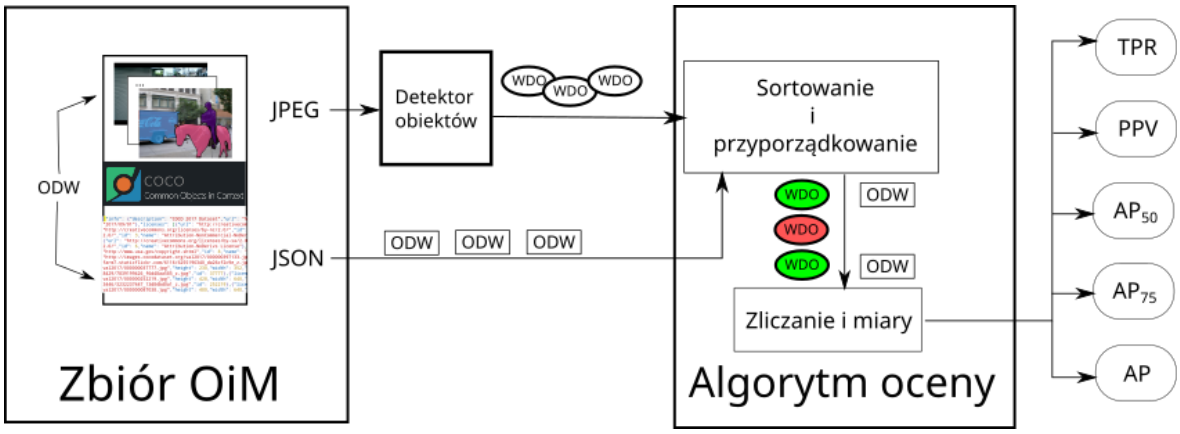
Na skutek wykonania modelu detekcji obiektów, po podaniu obrazu na wejście, otrzymujemy na wyjściu *wyniki detekcji obiektów*. Pojedynczy wynik detekcji obiektów (WDO) jest to rekord informacji dotyczącej położenia, klasy i ufności, z jaką model postuluje obecność ODW danej klasy w konkretnym prostokątnym obszarze obrazu.

Aby móc w ogóle przeprowadzić ocenę skuteczności detekcji obiektów, potrzebny jest zbiór obrazów wraz z opisem ODW na tych obrazach (ang. *object detecion dataset*). Taki zbiór danych i obrazów może stanowić zarówno zbiór uczący, jak i walidacyjny lub testowy dla problemu detekcji obiektów. W skrócie można przyjąć określenie *zbiór obrazów i metadanych* (zbiór OiM). Schematycznie proces oceny skuteczności detekcji obiektów został przedstawiony na Rys. 4.7.

4.4.1 Ocena pojedynczego wyniku

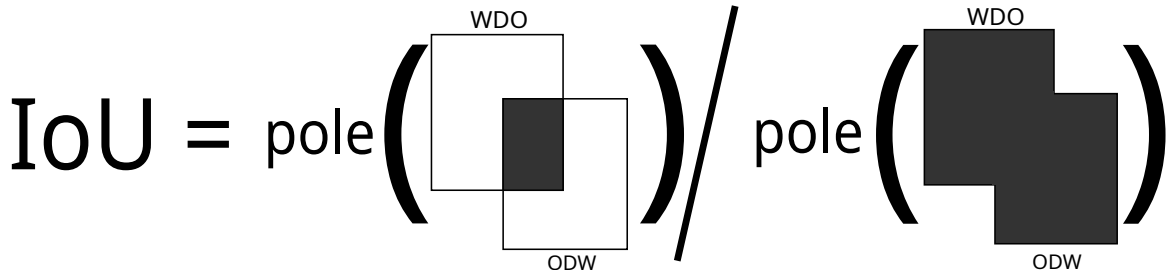
Podstawą wyznaczania wszystkich miar skuteczności jest określenie dla każdego WDO, czy jest on prawidłowy, czy nie. W języku angielskim WDO, przez analogię do klasyfikacji binarnej, są określane jako obserwacje pozytywne (ang. *positive*), odpowiednio: prawdziwe (TP) i fałszywe (FP).

A zatem, podstawą dla miar skuteczności detekcji obiektów jest klasyfikacja pojedynczego WDO jako TP lub FP. Dany WDO jest poprawny dla pewnego ODW ze zbioru OiM wtedy, jeżeli jest on tej samej klasy, oraz lokalizacja jest wystarczająco zgodna z prostokątem ODW. Miarą tej zgodności jest wartość ilorazu pól powierzchni: części wspólnej prostokątów i ich sumy w sensie zbioru (ang. *intersection over union*, IoU). W statystyce taka wartość jest nazywana indeksem Jaccarda. Minimalna war-



Rys. 4.7: **Schemat wyznaczania miar skuteczności detekcji obiektów.** Do wyznaczenia miar skuteczności niezbędny jest zbiór obrazów i metadanych (OiM), np. zbiór COCO, w którym są opisane obiekty do wykrycia (ODW). Wyniki detekcji obiektów zostają najpierw posortowane wg malejącej ufności, a następnie przypisane do odpowiednich ODW. Zliczanie poprawnych WDO jest podstawą wyznaczania wynikowych miar.

tość IoU, dla której uznaje się WDO za prawidłowy, to próg IoU (ang. *IoU threshold*), oznaczany jako T_{IoU}. Typowe wartości T_{IoU} to 0.5 lub 0.75, a maksymalnie stosuje się wartości do 0.95. Wyjaśnienie geometryczne wartości IoU pomiędzy prostokątami WDO i ODW znajduje się na Rys. 4.8.



Rys. 4.8: **IoU – miara ilorazu części wspólnej i sumy zbiorów prostokątów.** ODW – obiekt do wykrycia, prostokąt wzorcowy, WDO – wynik detekcji obiektów, prostokąt zwrócony przez model. Wartość IoU wynosi maksymalnie 1, kiedy prostokąty pokrywają się, a minimalnie 0, kiedy mają puste przecięcie.

ODW, dla których nie został znaleziony żaden WDO, to obserwacje negatywne, a dokładniej fałszywie negatywne (FN). Wyznaczenie tej ostatniej wartości nie następuje bezpośrednio – mając liczbę TP należy odjąć ją od liczby wszystkich ODW obecnych w zbiorze OiM.

Inaczej niż przy klasyfikacji binarnej, nie występuje pojęcie obserwacji prawdziwie negatywnej (TN). Liczność zbioru „potencjalnych WDO" jest zdecydowanie za wysoka

(rzędu $O(k \cdot n^2)$, gdzie n jest liczbą pikseli w obrazie[3], a k liczbą klas w zbiorze OiM), żeby rozpatrywanie „prawidłowo niewykrytych" obiektów miało jakikolwiek sens.

W dalszym tekście zostały przyjęte następujące oznaczenia:

- TP – liczba poprawnych WDO,
- FP – liczba błędnych WDO,
- FN – liczba niewykrytych ODW.

4.4.2 Miary zależne od progu ufności

Próg ufności – T_c – jest parametrem, który dla konkretnego zbioru WDO wskazuje, które WDO będą brane pod uwagę przy wyznaczaniu miar skuteczności. Są to te WDO, których ufność p jest większa lub równa T_c.

Z jednej strony, próg T_c jest niezbędny w niektórych modelach detekcji, aby określić, jakie WDO zostały rzeczywiście zwrócone dla danego obrazu. Dotyczy to zwłaszcza detekcji gęstej, jak w przypadku sieci RetinaNet. Z drugiej strony, dla prawidłowo wytrenowanego modelu ufność będzie miała pewien rozkład, który wykazuje dodatnią korelację z prawidłowością WDO, inaczej mówiąc – TP będą miały wysokie wartości p, a FP niskie.

A zatem, po ustaleniu progów T_{IoU} i T_c można wyznaczyć liczby TP, FP i FN, które można traktować jako pewne bezwzględne miary skuteczności na danym testowym zbiorze OiM. Bardziej praktyczne jest jednak posługiwanie się miarami względnymi, takimi jak czułość (TPR), która określa odsetek ODW, które zostały odnalezione przez model:

$$\mathrm{TPR} = \frac{TP}{TP + FN}. \tag{4.13}$$

Sama czułość jeszcze nie wystarcza do stwierdzenia, czy dany detektor obiektów jest dobry i możliwy do stosowania w praktyce – oprócz tego ważne jest, aby model zwracał jak najmniej FP. Dla dwóch modeli o takiej samej wartości TPR, lepszy będzie ten, który będzie miał mniejszą liczbę FP, a więc większą proporcję TP do FP, a ponieważ w przypadku idealnym $FP = 0$, to lepiej brać pod uwagę proporcję TP do liczby wszystkich zwróconych WDO. Proporcja ta nosi nazwę precyzji (PPV) i zdefiniowana jest następująco:

$$\mathrm{PPV} = \frac{TP}{TP + FP}. \tag{4.14}$$

Idealny detektor obiektów ma TPR i PPV równe jeden, ale w rzeczywistych modelach zachodzi pewien kompromis (ang. *tradeoff*), o którym decyduje wybór wartości T_c – zazwyczaj wysokie T_c wpływa dodatnio na PPV i ujemnie na TPR, a niskie odwrotnie. W ten sposób problem oceny detektora obiektów staje się dwukryterialny. Można

[3]Czynnik n^2 wynika z możliwości wskazania prostokąta między każdą parą pikseli.

porównywać ze sobą różne detektory, lub różne wartości T_c dla tego samego detektora, pod względem relacji dominacji w sensie Pareto: sytuacja, w której obie miary są jednocześnie wyższe, wskazuje obiektywnie lepszą alternatywę.

W niektórych sytuacjach pożądana może być redukcja tych dwóch miar do jednego kryterium. Jednym ze sposobów, który to umożliwia jest użycie średniej harmonicznej TPR i PPV, która jest nazywana F1 (ang. *F1-score*):

$$\mathrm{F1} = \left(\frac{\mathrm{TPR}^{-1} + \mathrm{PPV}^{-1}}{2}\right)^{-1} = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}. \tag{4.15}$$

Niekoniecznie można uznawać tę miarę za obiektywną, ponieważ w praktycznym zastosowaniu ważność TPR i PPV może nie być równa. Jeżeli koszt wystąpienia FN jest dla użytkownika modelu znacząco wyższy lub niższy niż koszt wystąpienia FP, model maksymalizujący F1 nie będzie optymalny. Mimo to, F1 jest popularną miarą, ponieważ ma interesujące własności – np. dana wartość F1 narzuca minimalne wartości TPR i PPV, jakie muszą być osiągnięte. Równanie 4.15 jest symetryczne dla PPV i TPR, dlatego wystarczy pokazać to dla PPV. Wzór na PPV jako funkcja F1 i TPR jest następujący:

$$\mathrm{PPV} = \frac{\mathrm{F1} \cdot \mathrm{TPR}}{2 \cdot \mathrm{TPR} - \mathrm{F1}}. \tag{4.16}$$

Minimalne PPV uzyskujemy, gdy TPR = 1:

$$\min_{\mathrm{F1}}(\mathrm{PPV}) = \frac{\mathrm{F1}}{2 - \mathrm{F1}}. \tag{4.17}$$

Podobnie, minimalne TPR uzyskujemy, gdy PPV = 1:

$$\min_{\mathrm{F1}}(\mathrm{TPR}) = \frac{\mathrm{F1}}{2 - \mathrm{F1}}. \tag{4.18}$$

Przykładowo, dla F1 = 0.5 minimalna wartości każdej z miar TPR i PPV wynoszą $\frac{0.5}{2-0.5} = \frac{1}{3}$ (a pozostała z nich ma wówczas wartość 1).

4.4.3 Miary średniej precyzji (AP)

Miarą innego typu, która również opisuje skuteczność detekcji obiektów za pomocą jednej liczby, jest średnia precyzja – AP. Wcześniej (np. w artykule opisującym zbiór OiM PASCAL VOC [20]) wariant tej miary był określany jako „przeciętna średnia precyzja" – mAP, jednak w tej nazwie kryje się redundancja (*przeciętna* oraz *średnia*), na co zwraca uwagę np. Redmon [74]. Zależnie od tego, względem czego jest uśredniana ta precyzja, otrzymujemy różne miary, które mają zbliżony charakter. Do agregacji wyników stosuje się prostą średnią arytmetyczną.

Średnia precyzja jest obliczana na podstawie krzywej precyzja-czułość (ang. *precision-recall curve*). Zgodnie z wcześniej opisanym kompromisem, w którym zmiana progu uf-

ności rozpatrywanych WDO może powodować zmiany PPV i TPR, należy najpierw posortować WDO malejąco pod względem ufności. Jak wskazują Henderson i Ferrari [37], miara AP nie bierze pod uwagę ufności poszczególnych WDO, a tylko wzajemne relacje porządku tych wartości.

Miara AP jest wyznaczana dla całego zbioru, nie dla poszczególnych obrazów, dlatego na takiej posortowanej liście sąsiadują ze sobą WDO pochodzące z różnych obrazów. Każdy WDO na liście jest oznaczony jako TP lub FP, zależnie od tego, czy został do niego przypisany ODW, czy nie. Można wówczas zdefiniować ciągi TP_k i FP_k, które są liczbą, odpowiednio, TP i FP wśród k pierwszych elementów tej listy. Na podstawie tych ciągów, można określić ciągi kroczącej precyzji PPV_k i kroczącej czułości TPR_k, analogicznie do wzorów 4.14 i 4.13, gdzie precyzja dla k obiektów o najwyższej ufności to:

$$\mathrm{PPV} = \frac{TP_k}{TP_k + FP_k}, \tag{4.19}$$

a czułość w tym wypadku należy odnosić do całego zbioru ODW, którego liczność jest oznaczona przez GT (od ang. *ground truth*):

$$\mathrm{TPR}_k = \frac{TP_k}{GT}. \tag{4.20}$$

W ten sposób, nawet dla prawidłowych WDO, dla małego k wartości TPR_k są niewielkie. Stopniowo ze wzrostem TP_k wartości TPR_k również rosną lub pozostają na tym samym poziomie, innymi słowy ciąg TPR_k jest monotoniczny (niemalejący).

Niestety, własność ta nie zachodzi dla ciągu PPV_k – generalnie, ponieważ dla prawidłowo działających modeli detekcji obiektu, WDO o najwyższej ufności są prawidłowe, na początku PPV_k ma wartość zbliżoną do jedności (minimalna wartość to zero, gdyby przypadkiem WDO o najwyższej ufności był nieprawidłowy). Kiedy pojawiają się FP, wartość kroczącej precyzji spada (rośnie mianownik we wzorze 4.19), a po każdym TP precyzja wzrasta – tym wolniej, im jest bliższa jedności[4] (rośnie zarówno licznik, jak i mianownik we wzorze 4.19).

Aby krzywa precyzji była monotoniczna w funkcji k, a tym samym w funkcji ufności WDO, można zdefiniować zmodyfikowany ciąg monotonicznej precyzji:

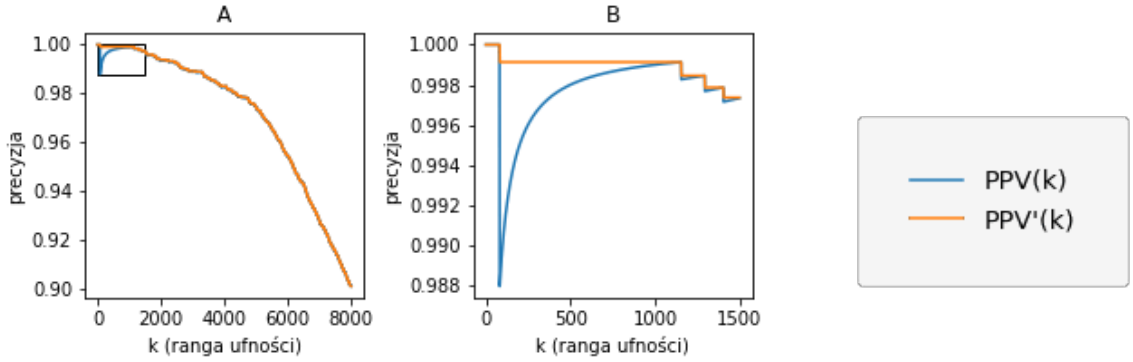
$$\mathrm{PPV}'_k = \max_{i \geqslant k}(\mathrm{PPV}_i). \tag{4.21}$$

Przykładowy ciąg PPV_k i odpowiadający mu ciąg PPV'_k przedstawiono na wykresie na Rys. 4.9.

Monotonizacja ciągu PPV_k jest zrealizowana przez pojedynczą iterację „od końca”, którą przedstawia Listing 4.1. Jest to fragment kodu użyty w badaniach do niniejszej rozprawy, który jest dostępny w repozytorium GitHub[5]. Krocząca precyzja po mono-

[4]Ściśle rzecz biorąc, są dwie sytuacje, kiedy PPV_k jest równie PPV_{k-1}: dla wartości zero, kiedy k-ty WDO jest nieprawidłowy, oraz dla wartości jeden, gdy k-ty WDO jest prawidłowy. Innymi słowy, stale występują albo tylko TP, albo tylko FP.

[5]`https://github.com/tgandor/urban_oculus/blob/master/evaldets/metrics.py` (dostęp:



Rys. 4.9: **Ciąg kroczącej precyzji wraz z jego monotonizacją.**
(A) Krocząca precyzja PPV_k w funkcji rangi ufności WDO z odpowiadającą jej monotoniczną wersją PPV'_k, (B) Powiększenie zaznaczonego prostokątem fragmentu wykresu (A).

tonizacji jest tam określana jako *interpolowana*, zgodnie z nazewnictwem przyjętym w bibliotece COCOapi, która również zawiera implementację obliczania miar AP.

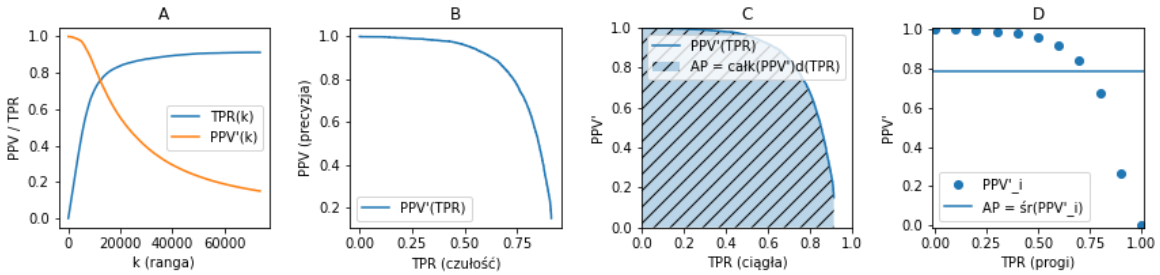
Listing 4.1: Obliczanie ciągu kroczącej precyzji w wersji monotonicznej

```
def interpolated_PPV(ppv: np.array):
    ppvl = ppv.tolist()
    for k in range(len(ppvl)-1, 0, -1):
        ppvl[k-1] = max(ppvl[k-1], ppvl[k])
    return np.array(ppvl)
```

Następnym krokiem obliczania średniej precyzji jest konstrukcja krzywej precyzja-czułość i wyznaczenie pola pod tą krzywą. Konstrukcja polega na stworzeniu funkcji przedziałami liniowej, której węzłami są kolejne pary elementów ciągów TPR_k (jako wartości x) i PPV'_k (jako wartości y). Przykład tych ciągów i wynikowej krzywej precyzja-czułość jest przedstawiony na Rys. 4.10 (A) i (B).

Krzywa precyzja-czułość dla danych wyników detekcji jest określona tylko do wartości maksymalnej czułości, tj. do ułamka wykrytych ODW (wartość miary TPR). Średnia precyzja jest całką oznaczoną tej krzywej w przedziale od zera do wartości TPR (albo do jedności, ale z założeniem, że precyzja dla wartości powyżej TPR wynosi zero). Geometryczna interpretacja tak określonej miary AP znajduje się na Rys. 4.10 (C). Takie sformułowanie nie jest ani wygodne obliczeniowo ani wydajne, dlatego stosuje się w zamian dyskretne przybliżenie. Polega ono na próbkowaniu wartości precyzji w określonych punktach czułości, które można oznaczyć jako $ppv'(r)$, gdzie r to wartość próbkująca TPR, a $ppv'(r)$ można zdefiniować następująco:

27 września 2022)



Rys. 4.10: **Krzywa precyzja-czułość i miara AP jako przybliżenie jej całki.** (A) Czułość i precyzja w funkcji malejącej rangi ufności, (B) Precyzja w funkcji kroczącej czułości, (C) Wartość średniej precyzji jako całka oznaczona krzywej precyzja-czułość, (D) Dyskretne przybliżenie średniej precyzji, średnia arytmetyczna interpolowanych wartości precyzji.

$$ppv'(r) = \begin{cases} \mathrm{PPV}'_k, \text{ gdy } r \leqslant \max(\mathrm{TPR}_k), \text{ gdzie } k = \min\{k \text{ takie, że } \mathrm{TPR}_k >= r\}, \\ 0, \text{ gdy } r > \max(\mathrm{TPR}_k). \end{cases} \tag{4.22}$$

Obliczanie wartości $ppv'(r)$ zostało zaprogramowane w sposób przedstawiony na Listingu 4.2. Argumenty `tpr` i `ppvi` to tablice wypełnione ciągami TPR_k i PPV'_k, a trzeci argument to opcjonalna tablica punktów próbkowania TPR. Domyślnie jest przyjmowane 101 próbek od 0 do 1 z krokiem 0.01. Przekazując inną tablicę można uzyskać wyniki takie jak dla miary mAP ze zbioru OiM PASCAL VOC [20], w której stosowane jest 11 próbek, z krokiem 0.1. Funkcja `np.searchsorted` zwraca minimalne indeksy tablicy (listę), których wartości są nie większe od podanych progów (tablica `thresholds`). Są to indeksy, które wskazałby algorytm sortowania przez proste wstawianie jako miejsce wstawienia wartości do tablicy tak, aby porządek był zachowany. Odpowiadają one wartości k ze wzoru 4.22. Wynikiem „funkcji" $ppv'(r)$ jest wartość w tablicy `ppvi` spod tych indeksów, o ile indeks nie pokazuje za ostatni element (wstawianie wartości na koniec ciągu). W takiej sytuacji należy zwrócić wartość 0, co implementacja w Listingu 4.2 realizuje przez wstępne wypełnienie wyniku zerami (funkcja `np.zeros_like`) i przerwanie wypełniania tablicy wyników przy pojawieniu się indeksu spoza tablicy `ppvi`.

Listing 4.2: Obliczanie próbkowanych wartości precyzji

```
def resample_pr_curve(tpr, ppvi, thresholds=None):
    """Find precision values for recall thresholds."""
    if thresholds is None:
        thresholds = np.linspace(0.0, 1.00, 101, endpoint=True)

    precision = np.zeros_like(thresholds)
    inds = np.searchsorted(tpr, thresholds, side="left")

    for ri, pi in enumerate(inds):
```

```
        if pi >= len(ppvi):
            break
        precision[ri] = ppvi[pi]

    return thresholds, precision
```

Próbkowane wartości precyzji (11 próbek) oraz linię prostą obrazującą średnią arytmetyczną próbek przedstawia Rys. 4.10 (D). Finalnie, wartość miary AP dla zbioru WDO wyznaczona jest właśnie przez tę średnią próbek precyzji dla progów czułości r pochodzących z przyjętego zbioru $\mathbf{R}$:

$$AP = \frac{1}{|\mathbf{R}|} \sum_{r \in \mathbf{R}} ppv'(r), \mathbf{R} = [0, 0.01, \ldots, 1.0]. \tag{4.23}$$

Nazwy szczegółowych miar typu AP. Konkretne miary skuteczności oparte na średniej precyzji wykorzystują opisany wcześniej schemat, z zastosowaniem dodatkowego dzielenia zbioru WDO na podzbiory, wielokrotnego kwalifikowania ich jako TP lub FP oraz uśredniania wyników cząstkowych. W podstawowym przypadku miara mAP używana w zbiorze OiM PASCAL VOC [20] uśredniała wyniki AP uzyskane dla WDO każdej klasy ODW z osobna. Można powiedzieć, że taka „średnia po wszystkich klasach" przeciwdziała niezrównoważeniu zbioru – ODW klasy `person` jest dużo więcej niż klasy `toothbrush`, więc połączenie WDO tych klas w jedną krzywą precyzja-czułość skutkowałoby brakiem wpływu rzadkich klas na ocenę skuteczności.

Klasyfikacja prawidłowości w mierze mAP była wykonywana raz, dla progu $T_{IoU} = 0.5$. Dlatego inne oznaczenia tej miary (spotykane np. w publikacjach, które stosują kilka różnych miar), to $\text{mAP}_{0.5}$ lub mAP@0.5. Dla zbioru OiM COCO [59] nie stosuje się w ogóle nazwy mAP, tylko AP z określonym oznaczeniem (indeks dolny w tekście drukowanym lub sufiks w kodzie źródłowym) – np. mAP to AP_{50}. W tym przypadku 50 to wartość progu T_{IoU} wyrażona w procentach. Biblioteka COCO API wyznacza również miarę AP_{75}, dokładnie na tej samej zasadzie (średnia AP poszczególnych klas), ale z progiem $T_{IoU} = 0.75$.

Główna i najbardziej ogólna miara zbioru COCO to **AP** (wyróżnienie w tej sekcji, w celu odróżnienia od AP, tj. „zwykłej" średniej precyzji ustalonego zbioru WDO). Miara **AP** to średnia *dziesięciu* miar AP, od AP_{50} do AP_{95} – czyli średnich precyzji wyznaczanych z wartościami progu T_{IoU} pochodzącymi ze zbioru $\mathbf{T}$:

$$\mathbf{AP} = \frac{1}{|\mathbf{T}|} \sum_{t \in \mathbf{T}} \text{AP}_t, \text{ gdzie } \mathbf{T} = [0.5, 0.55, \ldots, 0.95]. \tag{4.24}$$

Miary AP poszczególnych klas są również miarami typu **AP**, czyli są uśrednione z dziesięciu wartości T_{IoU}. Każdy z tych cząstkowych wyników jest średnią stu próbek PPV przy różnym progu TPR. Miary AP-klasy (np. `AP-person`, `AP-car`) są średnimi tysiąca takich próbek. Wynikowa miara **AP**, która jest średnią AP po klasach WDO (których jest 80), uśrednia 80 tys. próbek, które pochodzą z ośmiuset krzywych precyzja-czułość, dla każdej klasy i progu T_{IoU}, który dotyczy nakładania się WDO z ODW. Taki poziom ogólności czyni tę miarę adekwatną do problemów badanych w ni-

niejszej rozprawie, niezależnie od jej ograniczeń i krytyk, które zostaną omówione w rozdziale 7, w podrozdziale „Ocena detekcji obiektów".

Rozdział 5

Kompresja obrazów

Niniejszy rozdział traktuje o metodach kompresji obrazu, włączając w to podejścia bezstratne oraz stratne. Szczególną uwagę poświęcono algorytmowi JPEG, który został użyty w eksperymentach obliczeniowych w części badawczej pracy. W ostatniej sekcji znajduje się wyjaśnienie zagadnień związanych z jakością obrazu i metodami jej badania oraz taksonomia miar jakości obrazu, a także bardziej szczegółowy opis miary SSIM (indeksu podobieństwa strukturalnego), która została użyta w części eksperymentalnej.

5.1 Rodzaje kompresji obrazów

Obrazy przetwarzane lub generowane przez komputer można podzielić na rastrowe i wektorowe. Grafika wektorowa zawiera geometryczny opis obrazu za pomocą prymitywów graficznych (m.in. punkty, linie, łuki, trójkąty i inne wielokąty), które są reprezentowane jako liczby zmiennoprzecinkowe i pozwalają na odtworzenie obrazu w dowolnej skali (jest to reprezentacja dowolnie *skalowalna*, w granicach dokładności liczb w komputerze). Grafika rastrowa natomiast składa się ze wzoru (rastra) elementów – pikseli (ang. *picture element*, pixel), a każdy z nich ma konkretną wartość, która może oznaczać kolor lub intensywność, lub np. gęstość fragmentu ciała w skali Hounsfielda. Kształt i przestrzenne ułożenie pikseli może być różny, w szczególności w przypadkach sprzętu (wyświetlacze, maszyny drukarskie), ale najczęściej jest to siatka prostokątna zbudowana z kwadratów (tzw. aspekt jest równy jeden), a wartość piksela to jedna do kilku liczb całkowitych, których znaczenie zależy od użytej przestrzeni barw. Liczba składowych piksela to jednocześnie liczba kanałów obrazu – kanał jest to dwuwymiarowy obraz pojedynczej składowej, który można reprezentować w komputerze jako dwuwymiarową tablicę liczb.

Istnieją formaty zapisu obrazów do plików w formie nieskompresowanej (*surowej*). Przykładami takich formatów są binarny format BMP (ang. *bitmap*) oraz grupa formatów PNM (ang. *portable anymap*), wśród których są zarówno formaty tekstowe, jak i binarne. Zazwyczaj przechowywanie danych obrazu w takiej postaci jest niepraktyczne ze względu na duży rozmiar plików, jakkolwiek istnieją pewne zalety zapisu obrazu w taki sposób:

- jednoznacznie określony (niezależny od zawartości), stały dla danego rozmiaru rastrowego, rozmiar pliku obrazu,
- zerowy narzut obliczeniowy na przygotowanie obrazu do prezentacji lub przetwarzania,
- prostota implementacji ładowania obrazu,
- niski narzut formatu przy małych obrazach.

Z tych powodów, użycie nieskompresowanych danych może być zasadne, np. w systemach wbudowanych o mocno ograniczonych możliwościach, w językach lub środowiskach programowania, które nie mają odpowiednich bibliotek do kompresji i dekompresji obrazu, oraz w sytuacjach, w których narzut obliczeniowy na dekompresję lub kompresję jest niedopuszczalny lub niepożądany (np. testy wydajnościowe algorytmów przetwarzania obrazu).

Poza tymi szczególnymi przypadkami, powszechne jest zastosowanie *skompresowanych* formatów zapisu obrazu. Formaty te, a dokładniej mówiąc algorytmy kompresji, które za nimi stoją, można podzielić na bezstratne i stratne [96]. Metody bezstratne charakteryzują się całkowicie bezbłędnym odwzorowaniem danych obrazu po dekompresji. Najprostszą metodą osiągnięcia tego celu jest zastosowanie algorytmu kompresji ogólnego przeznaczenia (jak np. Deflate stosowany w formacie plików ZIP lub LZMA stosowany w plikach 7-ZIP). Przegląd metod bezstratnych można znaleźć w pracy Starosolskiego [84], która opisuje metody bezstratne stosowanie w obrazowaniu medycznym. Jest to jedna z dziedzin, w której często wymagane jest bezstratne przetwarzanie informacji wizualnej.

W tej samej pracy można znaleźć również inny podział, który dotyczy zarówno wybranych metod bezstratnych, jak stratnych – podział na podejścia transformacyjne i predykcyjne. W podejściu transformacyjnym następuje przekształcenie danych do postaci, która w większym stopniu poddaje się kompresji (możliwość odwrócenia wyniku tego przekształcenia w sposób całkowity lub częściowy określa wtedy, czy dana metoda jest stratna czy nie). Podejście predykcyjne polega na generowaniu danych na podstawie np. już dekompresowanej informacji. Ze względu na dużą redundancję informacji w obrazie – np. przeważającego podobieństwa sąsiednich pikseli oraz „płynności" zmian poziomów kanałów – możliwe jest „przewidywanie" fragmentów obrazu, bez przechowywania ich w wyniku kompresji. Stratność zależy w tym przypadku od obecności lub braku informacji korygującej predykcję. Dla konkretnego algorytmu mogą nawet występować elementy obydwu tych podejść.

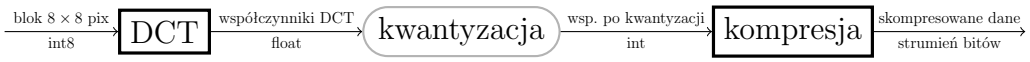
5.2 Kompresja JPEG

Powszechnie wykorzystywaną metodą stratnej kompresji obrazów kolorowych jest algorytm JPEG [98], którego nazwa pochodzi od grupy roboczej, która standaryzowała go na przełomie lat osiemdziesiątych i dziewięćdziesiątych XX wieku (ang. *Joint Photograpic Experts Group*). Na standard składają się dwie części – (1) algorytm JPEG

(wraz ze specyfikacją danych wejściowych i wyjściowych, trybów pracy itp.) oraz (2) format plików JFIF (ang. *JPEG file interchange format*), który specyfikuje, w jaki sposób dane wyjściowe algorytmu JPEG oraz metadane obrazu reprezentowane są na poziomie bajtów w pliku z rozszerzeniem `jpg` lub `jpeg`[1].

Co prawda JPEG nie był pierwszym formatem stratnej kompresji obrazu [98], ale w tym przypadku niewątpliwie godny podkreślenia jest fakt, że format ten stanowi owoc wysiłku standaryzacyjnego w sytuacji istnienia różnych, wzajemnie niekompatybilnych algorytmów i formatów opracowanych przez różnych producentów, często bez ogólnej dostępności implementacji. Stworzenie otwartego i ogólnodostępnego standardu utorowało dla obrazów cyfrowych drogę do nowych obszarów zastosowań, nowego oprogramowania i nowych użytkowników.

Przepływ danych w algorytmie JPEG został, z pominięciem znacznej liczby szczegółów, zilustrowany na Rys. 5.1.



Rys. 5.1: **Przepływ danych w algorytmie kompresji JPEG.**
Ogólny schemat kroków kompresji JPEG – zaokrąglony prostokąt oznacza operację stratną.
Nad strzałkami podano informację o danych wejściowych i wyjściowych poszczególnych kroków. Pod strzałkami umieszczono określenie typu tych danych. DCT oznacza dyskretną transformację cosinusową, a kompresja – „kodowanie entropii", tj. kodowanie Huffmana.

Szczegółowe omówienie procesu kompresji w algorytmie JPEG zostało zaprezentowane w Tabeli 5.1, która przedstawia kroki następujące po sobie przy kompresji, a Tabela 5.2 – dekompresji obrazu kolorowego (wielokanałowego). Jest to cykl kompresji-dekompresji (ang. *round trip*).

Opis w zaprezentowanych tabelach pokazuje, że kwantyzacja nie jest jedynym krokiem w procesie zapisu obrazu w skompresowanym formacie JPEG, który przyczynia się do utraty danych (stratności).

Transformacja bloków. Stratna kompresja standardu JPEG opiera się na dyskretnej transformacji cosinusowej (DCT) [1]. Tym samym, algorytm ten można zaliczyć do podejść transformacyjnych. Transformacja wprost (ang. *forward DCT*), określana również jako DCT-II, przetwarza blok 8×8 próbek ($f(x, y) \in [0, 255]$) na blok 8×8 współczynników $F(u, v)$ według następującego wzoru [98]:

$$F(u,v) = \frac{1}{4}C(u)C(v)[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}], \qquad (5.1)$$

gdzie $C(u)$ i $C(v)$ są stałymi normalizującymi, zdefiniowanymi następująco:

[1]Stąd pliki w formacie JFIF są potocznie nazywane plikami JPEG.

Tabela 5.1: **Kroki w kompresji JPEG obrazu kolorowego.**

Krok	Stratny	Opis
Zmiana przestrzeni barw	tak	Konwersja przestrzeni RGB na zmodyfikowaną przestrzeń $YC_b\ C_r$ (wartości z przedziału $[0, 255]$).
Utworzenie jednostek MCU	tak†	Minimalna jednostka kodowania (ang. *minimal coding unit*, MCU) jest blokiem pikseli o rozmiarze 8×8, 8×16, 16×8 lub 16×16. Dla składowych C_b i C_r rozmiar przechowywany w MCU wynosi zawsze 8×8 próbek.
Przesunięcie	nie	Odjęcie 128 od wartości próbek, zmiana zakresu z $[0, 255]$ na $[-128, 127]$.
Transformacja DCT	tak‡	Bloki zostają poddane dyskretnej transformacji cosinusowej.
Kwantyzacja	tak	Próbki w blokach są dzielone przez odpowiadające im we właściwej tablicy kwantyzacji dzielniki (składowa Y przez elementy QT1, a składowe C_b i C_r przez elementy QT2).
Kodowanie entropii	nie	Elementy bloków są wyczytywane po *krzywej zygzakowej*, następnie zapisane w kodowaniu długości serii (ang. *run-length encoding*, RLE), a ta postać jest zamieniana na strumień bitów za pomocą kodowania Huffmana*.
(Obraz zapisany na dysk lub transmitowany przez sieć)		

† – Dotyczy tylko obrazów, w których stosuje się subsampling chrominancji.
‡ – Są 3 metody: "fast" (całkowitoliczbowa, szybka), "int" (całkowitoliczbowa, domyślna) i "float" (zmiennoprzecinkowa, w przybliżeniu bezstratna).
* – Standard przewiduje również, w praktyce nieużywane, zastosowanie kodowania arytmetycznego zamiast kodowania Huffmana.

$$C(u), C(v) = \begin{cases} 1/\sqrt{2}, & \text{jeżeli } u, v = 0, \\ 1, & \text{w przeciwnym wypadku.} \end{cases} \tag{5.2}$$

Innymi słowy, stała przed sumą dla współczynnika DCT $F(0,0)$ jest równa $\frac{1}{4} \cdot C(0) \cdot C(0) = \frac{1}{8}$, dla pozostałych elementów pierwszego wiersza $F(0, v)$ jest to $\frac{1}{4} \cdot C(0) \cdot 1 = \frac{\sqrt{2}}{8}$, podobnie dla pozostałych elementów pierwszej kolumny, a dla wszystkich pozostałych elementów mają stale wartość $\frac{1}{4}$.

Transformacja odwrotna, określana również jako DCT-III (ang. *inverse DCT*), jest zdefiniowana w następujący sposób:

Tabela 5.2: **Kroki w dekompresji JPEG obrazu kolorowego.**

Krok	Stratny	Opis
(Obraz wczytany z dysku lub odebrany przez sieć)		
Dekodowanie entropii	nie	Dekodowanie Huffmana, rekonstrukcja jednostek MCU.
Odwrotna transformacja DCT	tak‡	Bloki zostają poddane odwrotnej dyskretnej transformacji cosinusowej.
Skalowanie współczynników	nie	Współczynniki DCT w blokach są mnożone przez odpowiadające im elementy z właściwej dla danej składowej tablicy kwantyzacji (przybliżona odwrotność kwantyzacji).
Przesunięcie	nie	Dodanie 128 do wartości próbek, powrotna zmiana zakresu z $[-128, 127]$ na $[0, 255]$.
Dekonstrukcja MCU	tak†	Składowe C_r and C_b są interpolowane do rozmiaru MCU.
Zmiana przestrzeni barw	tak	Powrotna konwersja ze zmodyfikowanej przestrzeni $YC_b\ C_r$ na przestrzeń RGB.

† – Dotyczy tylko obrazów, w których stosuje się subsampling chrominancji.
‡ – Są 3 metody: "fast" (całkowitoliczbowa, szybka), "int" (całkowitoliczbowa, domyślna) i "float" (zmiennoprzecinkowa, w przybliżeniu bezstratna).

$$f(x,y) = \frac{1}{4}[\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}]. \tag{5.3}$$

Kwantyzacja. Transformata (wynik transformacji) DCT, w postaci bloków współczynników DCT, jest następnie poddawana kwantyzacji. Kwantyzacja oznacza dyskretyzację – zamianę ciągłej wartości współczynnika na wartość dyskretną, w szczególności na liczbę całkowitą. W algorytmie JPEG jest to realizowane przez podzielenie współczynników DCT, $F(u,v)$ przez dzielniki z tablicy kwantyzacji (QT), z zaokrągleniem do najbliższej liczby całkowitej. Wartość dzielnika $Q(u,v) \in [1, 255]$ zależy od położenia (u,v) w bloku współczynników. Wartości $Q(u,v)$ są liczbami całkowitymi o rozmiarze jednego bajta, co wynika ze specyfikacji formatu pliku JFIF. Dzielniki QT są zapisywane w wynikowym pliku JFIF do wykorzystania w procesie dekompresji.

Efekt kwantyzacji jest tylko w przybliżeniu odwracany podczas dekompresji: skwantowane współczynniki, przed odwrotną transformacją DCT, są mnożone przez odpowiadający im element tablicy QT, co daje liczby o podobnym rzędzie wielkości, ale niedokładnej wartości. Im wyższe wartości dzielnika $Q(u,v)$, tym mniej dyskretnych wartości skwantowanego współczynnika $\lfloor\frac{F(u,v)}{Q(u,v)}\rceil$ jest możliwych. Współczynniki o ni-

skiej wartości bezwzględnej (mniejszej niż połowa wartości dzielnika z tablicy QT) po kwantyzacji stają się zerami. Notacja $\lfloor x \rceil$ oznacza zaokrąglenie do najbliższej liczby całkowitej, to znaczy:

$$\lfloor x \rceil \stackrel{\text{def}}{=} \begin{cases} \lfloor x + 0.5 \rfloor, & \text{jeżeli } x \geqslant 0, \\ \lceil x - 0.5 \rceil, & \text{jeżeli } x < 0. \end{cases} \tag{5.4}$$

Zmiana przestrzeni barw. Standardową reprezentacją obrazów cyfrowych do wyświetlania na ekranach komputerowych jest addytywna przestrzeń RGB, której składowe (czerwony, zielony, niebieski) są reprezentowane za pomocą 8-bitowych liczb całkowitych. Natomiast do transmisji i kompresji jest używana inna przestrzeń, która składa się z trzech kanałów:

- Y – luminancja (ang. *luma*): suma ważona składowych RGB, miara jasności piksela odpowiadająca wrażeniu odbieranemu przez ludzkie oko,
- C_b – pierwsza składowa chrominancji (ang. *chroma*): wartość, która odpowiada za odchylenie koloru w stronę niebieskiego,
- C_r – druga składowa chrominancji: wartość, która odpowiada za odchylenie koloru w stronę czerwonego.

Stąd pochodzi nazwa tej tej przestrzeni: $YC_b\,C_r$. Konwersja do przestrzeni $YC_b\,C_r$ przed kompresją oraz konwersja powrotna do RGB po dekompresji mogą wprowadzać niewielkie błędy, ponieważ współczynniki konwersji są liczbami zmiennoprzecinkowymi (lub stałoprzecinkowymi ułamkami)[2], a wartości wynikowe są zaokrąglane do liczb całkowitych o rozmiarze jednego bajta ($[0, 255]$).

Subsampling chrominancji. Do utraty informacji w obrazie, obok kwantyzacji i konwersji przestrzeni barw, może przyczynić się również technika subsamplingu (inaczej: *podpróbkowania*) chrominancji (ang. *chroma subsampling*). Dotyczy ona obrazów kolorowych i jest opcjonalną operacją bezpośredniego usuwania 50 lub 75 % informacji o kolorze (chrominancji), po konwersji przestrzeni kolorów, a przed transformacją DCT. Ta operacja ma związek z tworzeniem minimalnych jednostek kodowania (MCU). Jednostki MCU są to najmniejsze fragmenty obrazu, które są kompresowane razem. Dla obrazów monochromatycznych jednostkę MCU stanowi po prostu pojedynczy blok 8×8 pikseli (próbek), które w procesie kwantyzacji są dzielone przez dzielniki z jednej tablicy QT. Obrazy kolorowe mogą mieć w jednostce MCU trzy, cztery lub sześć bloków 8×8 próbek w następujących konfiguracjach (według rozmiaru przestrzennego):

- MCU 8×8 pikseli: brak subsamplingu, trzy bloki współczynników DCT: Y, C_b, C_r.

[2]Szczegółowy opis konwersji przestrzeni barw można znaleźć w implementacji opatrzonej szczegółowymi komentarzami pod adresem: `https://github.com/libjpeg-turbo/libjpeg-turbo/blob/master/jccolor.c`

- MCU 16×8 pikseli: subsampling 2×1 (poziomy), cztery bloki współczynników DCT: dwa bloki Y (lewy i prawy), C_b, C_r.
- MCU 8×16 pikseli: subsampling 1×2 (pionowy), cztery bloki współczynników DCT: dwa bloki Y (górny i dolny), C_b, C_r.
- MCU 16×16 pikseli: subsampling 2×2 (poziomy i pionowy), sześć bloków współczynników DCT: cztery bloki Y (ułożone w kwadrat 2×2), C_b, C_r.

Algorytm JPEG używa tylko dyskretnej transformacji cosinusowej o rozmiarze 8×8 próbek, chociaż w innych algorytmach możliwe jest określenie innych rozmiarów jednostki kodowania (ang. *coding unit*). Przykładowo, w algorytmie JPEG2000 możliwa jest transformacja całego obrazu jako jednej jednostki kodowania, albo użycie bloków 8×8, albo całego szeregu rozmiarów pośrednich [71].

Jak widać, subsampling chrominancji dotyczy jednostek MCU, których rozmiar jest większy niż 8×8. Wynika to z stąd, że kanały chrominancji C_b i C_r– nośniki informacji o kolorze – zapisywane w każdej jednostce MCU mają zawsze rozmiar 8×8. Większe jednostki MCU, które mają dwa (8×16 lub 16×8) lub cztery (16×16) bloki luminancji (Y), podczas zapisu dokonują subsamplingu w jednym lub dwóch wymiarach.

Podczas odtwarzania pikseli, przy dekompresji, dane o kolorze muszą być interpolowane (nadpróbkowane, skalowane w górę), tak aby pokryć całą powierzchnię jednostki MCU. Nie oznacza to jednak, że sąsiednie piksele mają ten sam kolor (tę samą wartość), ponieważ na ostateczny „kolor" piksela w przestrzeni barw RGB ma wpływ również kanał Y, który jest zachowywany w pełnej rozdzielczości. Właściwości ludzkiego wzroku powodują, że pomimo możliwości zaobserwowania efektów i artefaktów po powiększeniu obrazu (aberracje chromatyczne, interpolowane barwy nieobecne w obrazie źródłowym), spontaniczne oglądanie obrazów i wideo z subsamplingiem chrominancji nie pozwala na odróżnienie ich od obrazów zapisanych bez tej techniki [92].

5.2.1 Sterowanie jakością w algorytmie JPEG

JPEG, podobnie jak większość algorytmów kompresji stratnej, pozwala na uzyskanie kompromisu pomiędzy siłą kompresji (wyrażoną przez stosunek rozmiaru danych wejścowych do wyjściowych), a pogorszeniem jakości obrazu.

Parametr jakości Q. Jednym ze sposobów parametryzacji jakości w kompresji JPEG jest użycie parametru Q (ang. *JPEG quality*), który jest liczbą całkowitą z przedziału $[1, 100]$, gdzie niższe wartości oznaczają silniejszą kompresję i gorszą jakość, a wyższe wartości zapewniają wyższą jakość za cenę większego rozmiaru danych wyjściowych (słabszej kompresji).

Przykład kompresji obrazu z różnymi wartościami parametru Q został przedstawiony na Rys. 5.2.

Powszechnie stosowana implementacja algorytmu JPEG autorstwa Independent JPEG Group[3] (IJG) korzysta z zestawu predefiniowanych tablic kwantyzacji zależnie

[3] `https://ijg.org/` (dostęp: 27 września 2022)

Źródło obrazu: `http://r0k.us/graphics/kodak/kodak/kodim04.png`, domena publiczna

Rys. 5.2: **Wpływ parametru Q na jakość obrazu w kompresji JPEG.** Przykładowy obraz skompresowany z różnymi ustawieniami jakości JPEG, wybranymi za pomocą parametru Q. Górny wiersz – obraz w naturalnej wielkości. Dolny wiersz – fragment obrazu powiększony ośmiokrotnie, z interpolacją do najbliższego sąsiada (ang. *nearest neighbor interpolation*). Jakość obrazu po kompresji została zmierzona za pomocą indeksu podobieństwa strukturalnego – SSIM (ang. *structural similarity index*) – pomiędzy obrazem oryginalnym a skompresowanym.

od wartości parametru Q. Większość poziomów używa dwóch osobnych QT, pierwsza – QT1 – do składowej Y (luminancja), a druga – QT2 – do obydwu składowych chrominancji (C_b i C_r), zazwyczaj z wyższymi wartościami (silniejsza kwantyzacja) niż QT1.

Tablice kwantyzacji dla wybranych wartości Q są przedstawione w Tabeli 5.3. Pełne zestawienie tablic kwantyzacji JPEG dla $Q \in [1, 255]$ używanych w implementacji IJG oraz w badaniach związanych z niniejszą rozprawą zostało umieszczone w repozytorium GitHub autora[4].

Wartość Q = 100, mimo intuicyjnie narzucającej się analogii z „procentami jakości", nie oznacza kompresji bezstratnej. Jest to minimalny poziom kompresji w bazowej wersji algorytmu JPEG (ang. *baseline JPEG*) i wiąże się z najmniejszą stratnością – QT1 i QT2 mają wszystkie wartości równe jeden. Przy niezastosowaniu subsamplingu chrominancji uzyskuje się najlepszą jakość obrazu, na jaką pozwala JPEG; użycie wartości $Q(u, v) = 1$ powoduje jedynie zaokrąglenie współczynników DCT do wartości całkowitych, bez zmiany rzędu wielkości.

Dla Q = 1 wszystkie dzielniki w QT1 i QT2 mają wartość 255. W połączeniu z podpróbkowaniem chrominancji uzyskuje się najgorszą możliwą jakość przy najmniejszym rozmiarze danych wyjściowych; niemniej jednak takie ustawienie jakości może nie mieć

[4] `https://github.com/tgandor/urban_oculus/tree/master/jpeg/quantization` (dostęp: 27 września 2022)

Tabela 5.3: **Wybrane tablice kwantyzacji w zależności od parametru Q.**

Q	QT1 (Y lub obraz monochromatyczny)								QT2 (C_r i C_b)							
60	13	9	8	13	19	32	41	49	14	14	19	38	79	79	79	79
	10	10	11	15	21	46	48	44	14	17	21	53	79	79	79	79
	11	10	13	19	32	46	55	45	19	21	45	79	79	79	79	79
	11	14	18	23	41	70	64	50	38	53	79	79	79	79	79	79
	14	18	30	45	54	87	82	62	79	79	79	79	79	79	79	79
	19	28	44	51	65	83	90	74	79	79	79	79	79	79	79	79
	39	51	62	70	82	97	96	81	79	79	79	79	79	79	79	79
	58	74	76	78	90	80	82	79	79	79	79	79	79	79	79	79
80	6	4	4	6	10	16	20	24	7	7	10	19	40	40	40	40
	5	5	6	8	10	23	24	22	7	8	10	26	40	40	40	40
	6	5	6	10	16	23	28	22	10	10	22	40	40	40	40	40
	6	7	9	12	20	35	32	25	19	26	40	40	40	40	40	40
	7	9	15	22	27	44	41	31	40	40	40	40	40	40	40	40
	10	14	22	26	32	42	45	37	40	40	40	40	40	40	40	40
	20	26	31	35	41	48	48	40	40	40	40	40	40	40	40	40
	29	37	38	39	45	40	41	40	40	40	40	40	40	40	40	40
90	3	2	2	3	5	8	10	12	3	4	5	9	20	20	20	20
	2	2	3	4	5	12	12	11	4	4	5	13	20	20	20	20
	3	3	3	5	8	11	14	11	5	5	11	20	20	20	20	20
	3	3	4	6	10	17	16	12	9	13	20	20	20	20	20	20
	4	4	7	11	14	22	21	15	20	20	20	20	20	20	20	20
	5	7	11	13	16	21	23	18	20	20	20	20	20	20	20	20
	10	13	16	17	21	24	24	20	20	20	20	20	20	20	20	20
	14	18	19	20	22	20	21	20	20	20	20	20	20	20	20	20
96	1	1	1	1	2	3	4	5	1	1	2	4	8	8	8	8
	1	1	1	2	2	5	5	4	1	2	2	5	8	8	8	8
	1	1	1	2	3	5	6	4	2	2	4	8	8	8	8	8
	1	1	2	2	4	7	6	5	4	5	8	8	8	8	8	8
	1	2	3	4	5	9	8	6	8	8	8	8	8	8	8	8
	2	3	4	5	6	8	9	7	8	8	8	8	8	8	8	8
	4	5	6	7	8	10	10	8	8	8	8	8	8	8	8	8
	6	7	8	8	9	8	8	8	8	8	8	8	8	8	8	8

Tablice te są używane m.in. w implementacji `libjpeg` grupy IJG. Q = 96 jest najczęściej stosowaną wartością w zbiorze obrazów używanym w części eksperymentalnej niniejszej pracy.

praktycznego zastosowania ani w przetwarzaniu obrazów za pomocą modeli głębokich (co pokazuje część badawcza niniejszej pracy), ani w przechowywaniu obrazów do prezentacji ludzkiemu odbiorcy.

Wyjątkiem od tej reguły mogą być specyficzne zastosowania np. w celu ukrywania informacji (ang. *obfuscation*), np. znany autorowi pracy przypadek monitoringu publicznego miejsca za pomocą kamery internetowej dostępnej bez autoryzacji w sieci firmowej. Celem było umożliwienie wykrycia (przez człowieka) obecności innych osób, aby uniknąć niepotrzebnych kontaktów międzyludzkich, ale w taki sposób, aby nie naruszać prywatności i nie ujawniać tożsamości ludzi widocznych na obrazie. Rozwiązanie polegało na nadawaniu strumienia wideo w kodowaniu MJPEG (ang. *motion JPEG*), które stosuje algorytm JPEG, z jakością Q = 1. W ten sposób cel podstawowy został zrealizowany z zachowaniem anonimowości transmitowanych osób, bez możliwości ich identyfikacji i śledzenia, oraz z bardzo niskimi wymaganiami przepustowości sieci (silna redukcja rozmiaru przesyłanych danych).

5.3 Miary jakości obrazu

O ile określenie „jakość obrazu" jest w dużej mierze intuicyjne, o tyle, w celu mierzenia jakości, potrzeba ją dokładniej zdefiniować. We wcześniejszym tekście pojawiają się odniesienia do jakości obrazu, np. stwierdzenie, że kompresja stratna pogarsza (a dokładniej: może pogarszać) jakość obrazu. Ocena jakości obrazu stanowi całą dziedzinę badań i publikacji, którą oznacza się skrótem IQA (ang. *image quality assessment*).

Rodzaje oceny jakości. Przede wszystkim, jakość obrazu może być określona w sposób obiektywny lub subiektywny. Jakość subiektywna to jakość „dla człowieka" i nie jest ona wynikiem jakiejś algorytmicznej operacji na obrazie, ale oceną wystawioną przez ludzkich obserwatorów. Pojedyncza ocena może się różnić od innych i stąd właśnie pochodzi nazwa jakości subiektywnej. Jeżeli jednak zostanie zaangażowana większa badawcza grupa osób, to oceny uczestników badania (ang. *test subjects*, to określenie można skojarzyć z subiektywnością), mimo że osobno są różne, ale po uśrednieniu stają się zbieżne i powtarzalne, nawet na innej grupie badawczej.

Miary obiektywne natomiast nie zależą od oceniającego podmiotu i mogą być wyliczane w sposób zautomatyzowany. Dla zadanego wejścia zawsze zostanie wyznaczona konkretna wartość, która będzie liczbą. Miary subiektywne potrzebują korzystać z różnych technik, takich jak porównywanie obrazów parami lub większymi grupami (obserwator ma wskazać najlepszy z danego porównania), albo ocena w skali psychometrycznej (np. skala Likerta – pięciostopniowa skala przekonania ankietowanego: „zdecydowanie tak", „tak", „nie wiem", „nie", „zdecydowanie nie"). Wymaga to odpowiedniego opracowania statystycznego zbioru odpowiedzi.

A zatem, miary subiektywne najlepiej opisują jakość w sensie atrakcyjności wizualnej obrazu dla ludzkiego obserwatora, ale są bardzo kosztowne do wyznaczenia. Co więcej, największym problemem jest to, że wyniki miar dla jednego obrazu nie mogą zostać w żaden sposób zastosowane do innego, co wyklucza zastosowanie w jakimkol-

wiek zautomatyzowanym procesie. Stąd celem tworzenia miar obiektywnych jest jak najlepsza zgodność z wynikami oceny subiektywnej.

Miary referencyjne i bezreferencyjne. Miary jakości obrazu można podzielić na referencyjne i bezreferencyjne. Przy wyznaczaniu miar referencyjnych musi być dostępny obraz referencyjny – „oryginał", względem którego porównuje się oceniany obraz. Algorytm lub obserwator ma za zadanie określić zgodność badanego obrazu z tym oryginałem. Można przyjąć jakąś wartość, np. jeden, jako całkowitą zgodność (np. dla obrazu identycznego, lub równoważnego w jakiejś relacji), a inną, np. zero, za „całkowicie niezgodny".

Inaczej jest w przypadku miar bezreferencyjnych. Wówczas dostępny jest tylko jeden obraz i należy dla niego określić „bezwzględną" jakość. Dla człowieka jest to możliwe – obserwator zdjęcia ocenia świadomie lub podświadomie ostrość, obecność lub brak rozmycia, kolory, kontrast itp. Dla podejścia obiektywnego jest to znacznie trudniejszy przypadek – potrzeba określić (lub wytrenować, w przypadku uczenia maszynowego) cechy, które posiadają obrazy o wysokiej lub niskiej jakości. Dodatkowa trudność polega na tym, że treść obrazu może przedstawiać coś, co wygląda w rzeczywistości jak obraz o niskiej jakości (np. przedmiot, którego wzornictwo wykorzystuje efekt *pikselizacji*), ale sam obraz jest bardzo dobrej jakości.

Przykładem trudności bezreferencyjnej oceny jakości obrazu może być własny projekt autora niniejszej rozprawy, który dotyczył rozpoznawania (identyfikacji) twarzy. Akwizycja obrazu prowadzona była w sposób ciągły, jako sekwencja wideo i do oceny należało wybierać „dobre ujęcia". Zaobserwowano, że rozmyte obrazy twarzy wykazują za słabe podobieństwo do wzorca. Przy tworzeniu bazy osób dynamicznie (*online*), przyjęcie za wzorzec takiego rozmytego (np. przez rozmycie ruchu) obrazu, skutkowało przypisaniem do tej tożsamości wielu innych, podobnie rozmytych, obrazów twarzy. Zaproponowane rozwiązanie miało polegać na badaniu ostrości obrazu poprzez różnicę danej klatki z tą samą klatką przekształconą rozmyciem Gaussa – w założeniu, ostry obraz będzie dawał większą, a rozmyty – mniejszą różnicę. Jest to badanie liczby i intensywności krawędzi (wartości wektorów gradientów jasności) na obrazie. Niestety, taka miara nie sprawdziła się w praktyce, ponieważ wynik dla obrazu bardziej zależał od zawartości niż od faktycznej jakości – rozmyty obraz osoby ubranej w koszulę w kratkę uzyskiwał wynik wyższy, niż ostry obraz osoby w gładkiej koszuli.

Na szczęście, w przypadku kompresji adekwatne są miary referencyjne, ponieważ obrazem odniesienia jest po prostu nieskompresowany oryginał. Można i stosuje się miary obiektywne, co umożliwia zautomatyzowaną ocenę jakości obrazu, należy jednak pamiętać o ograniczeniach tych miar.

Miary jakości obrazu stosowane dla kompresji. Jeżeli potraktuje się porównanie z obrazem oryginalnym jako aproksymację funkcji, a każdy piksel jako węzeł, to można zastosować miary błędu aproksymacji oparte na różnicy wartości pikseli – różnicę bezwzględną oraz średni błąd kwadratowy MSE:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(p_i - \hat{p_i}\right)^2, \tag{5.5}$$

gdzie p_i to wartość i-tego piksela w obrazie referencyjnym, $\hat{p_i}$ to wartość odpowiadającego mu piksela w obrazie ocenianym, n to liczba pikseli w obrazie. Pochodna miara, której rząd wielkości odpowiada zakresowi wartości pikseli obrazu, to pierwiastek miary MSE (ang. *root mean squared error*, RMSE):

$$RMSE = \sqrt{MSE}. \tag{5.6}$$

Na średnim błędzie kwadratowym opiera się miara stosunku szczytowego sygnału do szumu (ang. *peak signal to noise ratio*, PSNR). Jest to wartość podawana w decybelach (skala logarytmiczna). Dla obrazu 8-bitowego, w którym maksymalna wartość piksela („szczytowy sygnał") wynosi 255, wartość PSNR wyraża się wzorem:

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right). \tag{5.7}$$

Ta miara jest używana m.in. przez Rabbaniego [71], do porównywania jakości obrazów skompresowanych stratnie algorytmami JPEG i JPEG2000. W implementacjach JPEG2000 można zadawać stopień kompresji przez podanie minimalnej wartości PSNR, jaka ma być osiągnięta przy porównaniu wyniku kompresji z obrazem oryginalnym. Miara PSNR ma niewątpliwe zalety, a jedną z nich jest użycie skali logarytmicznej, która (oczywiście w pewnym ograniczonym przedziale) odpowiada ludzkiej percepcji. Psychometria dla różnych bodźców wykazała, że narządy zmysłów właśnie w taki sposób odbierają różnice względne w poziomie określonej wielkości, np. wysokość dźwięku (oktawa, czyli podwojenie częstotliwości), natężenie światła, głośność dźwięku itp. W ten schemat wpisuje się błąd odtwarzania obrazu, określany jako szum (czyli zakłócenia), w odróżnieniu od sygnału, którym jest rzeczywista informacja wizualna. To co jest odbierane (badany obraz) jest sumą tych dwóch informacji. Natomiast „szczytowy sygnał" to sztuczna stała, która ma być górnym ograniczeniem możliwej wartości szumu, co zapewnia, że wynik miary będzie nieujemny.

Mimo tych zalet, miara PSNR ma bardzo istotne ograniczenie – jest ona uśrednieniem punktowych różnic, w związku z czym nie bierze pod uwagę w ogóle kontekstu i wzajemnych relacji sąsiednich pikseli, nie mówiąc o większym otoczeniu. Drugą wadą jest traktowanie błędu jako globalnej średniej, bez analizowania nawet jego rozkładu – na przykład zwiększenie wartości pikseli o pewną stałą („translacja" wartości) lub pomnożenie przez jakiś czynnik („skalowanie" wartości) znacznie mniej wpływa na odbiór jakości obrazu niż np. losowy szum, a wszystkie te trzy przypadki mogą mieć identyczną wartość MSE, a tym samym PSNR. Trzecią, praktyczną wadą PSNR, jest nieokreślona („nieskończona") wartość dla porównania dwóch identycznych obrazów (MSE równe 0). W przypadku algorytmu JPEG2000, w wynikowym strumieniu stosowane są tzw. warstwy (ang. *layer*), które nałożone na wynik dają coraz dokładniejsze przybliżenia obrazu wejściowego. Zadając jakość należy podać wartości PSNR w decybelach, jako rosnący ciąg wartości. Jeśli ostateczny wynik ma być bezstratny, ostatnią wartością jest

zero (w taki sposób poradzono sobie z niemożliwością reprezentacji bezstratnej jakości za pomocą PSNR).

Kolejną miarą wartą wzmiankowania jest uniwersalny indeks jakości obrazu (ang. *universal image quality index*, UIQI) autorstwa Wanga i Bovika [100]. Żeby opisać interpretację UIQI, warto jest wprowadzić pewną funkcję podobieństwa dwóch wartości x i y ($x > 0, y > 0$), $sim(x, y)$ o wartościach z przedziału $(0, 1]$:

$$sim(x, y) = \frac{2xy}{x^2 + y^2} \tag{5.8}$$

Z dziedziny wykluczono 0, ponieważ, gdy $x = y = 0$, wówczas funkcja ta ma wartość nieokreśloną – występuje symbol nieoznaczony $\frac{0}{0}$. Można jednak obliczyć granicę funkcji w tym punkcie, za pomocą reguły de l'Hospitala (oznaczanej jako $\overset{\text{H}}{=}$):

$$\lim_{y=x, x\to 0} sim(x, y) = \lim_{x\to 0} \frac{2x^2}{x^2 + x^2} = \lim_{x\to 0} \frac{x^2}{x^2} \overset{\text{H}}{=} \lim_{x\to 0} \frac{2x}{2x} \overset{\text{H}}{=} \lim_{x\to 0} \frac{2}{2} = 1 \tag{5.9}$$

W związku z tym można przyjąć, że z definicji $sim(0, 0) := 1$. Korzystając z tego zapisu, można określić miarę UIQI w następujący sposób:

$$UIQI(x, y) = r_{xy} \cdot sim(\overline{x}, \overline{y}) \cdot sim(\sigma_x, \sigma_y), \tag{5.10}$$

gdzie: r_{xy} jest wartością korelacji (Pearsona) między wartościami odpowiadających sobie pikseli, $\overline{x}$ to średnia wartość pikseli obrazu x, a σ_x oznacza odchylenie standardowe pikseli obrazu x.

Z wzoru 5.10 wynika, ze obrazy do porównania muszą mieć identyczne rozmiary rastrowe, aby można było wyznaczyć korelację wartości pikseli. Nie jest to jakieś wyjątkowe wymaganie dla miary referencyjnej (MSE również to zakłada). Można również zaobserwować, że miara UIQI jest symetryczna (kolejność argumentów nie jest istotna).

UIQI jako iloczyn trzech podobieństw – korelacji, podobieństwa wartości średniej i podobieństwa zmienności (odchylenia standardowego) – łączy je wszystkie w jedną wartość, dzięki czemu prosta zmiana jasności lub kontrastu zachowuje wyższe podobieństwo do obrazu referencyjnego niż losowe zaszumienie.

Odchylenia standardowe każdego z obrazów (σ_x i σ_y) występują nie tylko w ostatnim czynniku, ale także są ukryte w pierwszym, tj. korelacji, która wyraża się wzorem:

$$r_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}, \tag{5.11}$$

gdzie σ_{xy} to kowariancja obrazów:

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \overline{x})(y_i - \overline{y}) \tag{5.12}$$

Po podstawieniu wyrażeń 5.8 i 5.12 do wzoru 5.10 powstaje wzór:

$$UIQI(x, y) = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \cdot \frac{2 \cdot \overline{x} \cdot \overline{y}}{\overline{x}^2 + \overline{y}^2} \cdot \frac{2 \cdot \sigma_x \sigma_y}{\sigma_x^2 + \sigma_y^2} \tag{5.13}$$

Odchylenia standardowe w pierwszej potędze skracają się i pozostają we wzorze jedynie kwadraty tych odchyleń, tj. wariancje każdego z obrazów:

$$\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2. \tag{5.14}$$

Po wymnożeniu i uporządkowaniu czynników otrzymuje się wzór w postaci, w której przedstawili go Wang i Bovik [100], tj.:

$$UIQI(x,y) = \frac{4\sigma_{xy}\,\overline{x}\,\overline{y}}{(\overline{x}^2+\overline{y}^2)(\sigma_x^2+\sigma_y^2)}. \tag{5.15}$$

Wyznaczanie miary UIQI nie jest wykonywane dla całych obrazów, ale są one dzielone na kafelki 8×8 i każdy fragment jest porównywany osobno, a następnie wszystkie wartości są agregowane średnią arytmetyczną, dając w wyniku indeks podobieństwa UIQI.

Niestety, prosta implementacja wyliczania tej miary w praktyce będzie mało użyteczna, właśnie z powodu występowania symboli nieoznaczonych $\frac{0}{0}$ dla obszarów o jednolitej wartości pikseli. Jeżeli oba z porównywanych obrazów zawierają w odpowiadającym sobie miejscu obszar 8×8 o stałej wartości pikseli (nawet różnej, tzn. $x_i \neq y_i$), wówczas wartość miary będzie nieokreślona (ang. *not-a-number*, NaN).

I właśnie ten problem rozwiązuje późniejsza praca tych samych[5] autorów [101], która wprowadza miarę SSIM (ang. *Structural SIMilarity [index]*), tj. indeks podobieństwa strukturalnego:

$$SSIM(x,y) = \frac{(2\,\overline{x}\,\overline{y}+c_1)+(2\,\sigma_{xy}+c_2)}{(\overline{x}^2+\overline{y}^2+c_1)\,(\sigma_x^2+\sigma_y^2+c_2)}. \tag{5.16}$$

Stałe c_1 i c_2 ($c_1 \ll 1$ i $c_2 \ll 1$) stanowią stabilizujące składniki odpowiednio dla wartości średniej pikseli i dla wariancji (i kowariancji) obrazów. Dzięki temu uniknięto niestabilności numerycznej miary SSIM w sytuacji, kiedy $\overline{x} \approx \overline{y} \approx 0$ lub $\sigma_x^2 \approx \sigma_y^2 \approx 0$.

W ten sposób miara SSIM zachowuje opisane wcześniej teoretyczne zalety UIQI, a jest pozbawiona jej praktycznych ograniczeń. Pozostałe różnice tych miar mają charakter techniczny – indeks SSIM jest obliczany nie dla rozłącznych „kafelek" 8×8, ale dla nakładającego się przesuwnego okna o rozmiarze 9×9 (lub innym, ale nieparzystym).

Miara SSIM jest bardzo rozpowszechniona (liczba cytowań pracy [101] przekracza 20 tysięcy), nie tylko w dziedzinie IQA, ale w wielu innych zastosowaniach, także w uczeniu maszynowym [6]. Powstało kilka odmian miary SSIM, a także została ona zaadaptowana do analizy jakości wideo (miara jakości w wersji temporalnej).

Dobre miary jakości obrazu, takie jak SSIM, odegrały kluczową rolę w rozwoju algorytmów kompresji obrazów i wideo, umożliwiając obecną rewolucję w dziedzinie komunikacji, która obejmuje kolejno: publikowanie wideo w internecie (serwisy VoD), następnie strumieniowanie treści na żywo, a w końcu dwustronną komunikację wideo,

[5] Współautorami byli dodatkowo Sheikh i Simoncelli, którzy wykryli problemy w mierze UIQI i pomogli je rozwiązać.

która coraz częściej uzupełnia i zastępuje komunikację głosową (telefoniczną).

Rozdział 6

Badania empiryczne

Niniejszy rozdział opisuje badania empiryczne, które składały się z dwóch etapów realizacji celów rozprawy. Pierwszy etap był poświęcony zbadaniu wpływu kompresji stratnej na skuteczność detekcji obiektów z użyciem uczenia głębokiego. Drugi etap nakierowany był na zbadanie możliwości poprawy skuteczności detekcji na obrazach, których jakość została pogorszona wskutek kompresji.

W pierwszym podrozdziale przedstawiono przyjęte w pracy praktyki, które zapewniają odtwarzalność przeprowadzonych badań: użycie ogólnodostępnych zbiorów danych, publiczne udostępnienie danych wynikowych oraz oprogramowania użytego w realizacji badań – zarówno autorskiego, napisanego specjalnie w tym celu, jak i udostępnionego przez innych autorów.

W dalszej części opisano konfigurację zestawu eksperymentalnego dla etapu I eksperymentów – testowy zbiór OiM, sposób i rezultaty degradacji jakości poprzez kompresję obrazu, wykorzystane miary skuteczności detekcji obiektów oraz zestawienie testowanych modeli.

Następnie przedstawiono wyniki pierwszego etapu eksperymentów, który dotyczył badania dziewięciu pre-trenowanych modeli detekcji obiektów na reprezentatywnym zbiorze testowym w całym zakresie wartości parametru kompresji Q.

Opis wyników etapu II, w którym prowadzony był trening modeli detekcji obiektów, jest poprzedzony opisem zestawu eksperymentalnego użytego w tym etapie, ze wskazaniem elementów wspólnych z etapem I. Opisano różnice dotyczące próbkowania wartości Q, architektur modeli i procedury ich treningu, a także konfiguracji sprzętowej stanowisk, na których był on przeprowadzony.

Badane były dwie różne architektury do detekcji obiektów, z których każda była poddana treningowi w czterech różnych konfiguracjach. Uzyskane modele detekcji zostały następnie zbadane analogicznie jak pre-trenowane modele w etapie I, jednak z rzadszym próbkowaniem parametru Q. Uzyskane wyniki zaprezentowano z użyciem tabel i wykresów oraz opisano poczynione na ich podstawie obserwacje. Dalszą dyskusję rezultatów i wnioski zamieszczono w kolejnym rozdziale „Omówienie rezultatów i wnioski".

6.1 Odtwarzalność badań

Cecha odtwarzalności badań (ang. *reproducibility*), tj. możliwości replikacji eksperymentów, jest fundamentalna dla metody naukowej [94]. Szczególne znaczenie ma ona w dziedzinie uczenia głębokiego. Crane [13] wskazuje przyczyny problemów z odtwarzalnością badań dotyczących uczenia maszynowego – jedną z nich jest popularność tematyki, która wiąże się z dużą liczbą publikacji – i wskazuje praktyki, które są pomocne przy zapewnieniu tej odtwarzalności. Dziedziny, w których stawia się najwyższe wymagania dotyczące odtwarzalności, to m.in. medycyna [78] i konkursy uczenia głębokiego [39]. Powstają również specjalne narzędzia, które mają służyć ułatwieniu replikacji eksperymentów uczenia głębokiego, takie jak biblioteka w języku Python o nazwie `dtoolAI` [32].

Niniejsze badania zostały przeprowadzone z zachowaniem zalecanych praktyk odtwarzalności. W szczególności zadbano o:

- udostępnienie kodu źródłowego oprogramowania stworzonego na potrzeby przeprowadzenia badań na platformie GitHub[1],
- składowanie danych wynikowych z eksperymentów w publicznym repozytorium danych Dataverse[2],
- użycie ogólnodostępnego zbioru COCO[3] jako danych wejściowych,
- użycie ogólnodostępnych i otwartoźródłowych bibliotek PyTorch, COCO API i Detectron2, oraz aplikacji ImageMagick, w badawczym kodzie źródłowym.

Wyniki uzyskane w pierwszym etapie można łatwo poddać replikacji, ponieważ eksperymenty obliczeniowe miały charakter deterministyczny (dzięki użyciu modeli pretrenowanych). Nie jest to regułą w badaniach nad uczeniem głębokim, ze względu na czynnik losowy obecny w procesie treningu modeli.

Odtwarzalność drugiego etapu eksperymentów, w którym występował trening modeli, została zapewniona w standardowy sposób, który polega na udostępnieniu w Internecie wag sieci oraz metadanych zebranych w procesie treningu. Dane te zostały umieszczone w publicznym repozytorium utworzonym na platformie Harvard Dataverse[4].

6.2 Zestaw eksperymentalny w etapie I

W pierwszym etapie eksperymentów wykorzystano modele pre-trenowane, które zostały pobrane z Internetu. Zbadano wpływ kompresji na detekcję obiektów przez te modele na obrazach ze zbioru testowego. Obrazy te były poddawane kompresji w

[1] `https://github.com/tgandor/urban_oculus` (dostęp: 27 września 2022)
[2] `https://doi.org/10.7910/DVN/UPIKSF` (dostęp: 27 września 2022)
[3] `https://cocodataset.org/` (dostęp: 27 września 2022)
[4] `https://doi.org/10.7910/DVN/UHEP3C` (dostęp: 27 września 2022)

szerokim zakresie wartości parametru Q, który wpływa na stopień kompresji i jakość obrazu. Dla każdego modelu i każdej wartości parametru Q wyznaczono wartości miar skuteczności detekcji.

Zbiór testowy. Dostępnych publicznie zbiorów obrazów i metadanych (OiM) służących do testowania algorytmów detekcji obiektów jest wiele. Najważniejsze z nich (wymienione w pracy Liu [60]) w chwili obecnej to: PASCAL VOC [20], ImageNet [79] (ale nie 1000-klasowy zbiór obrazów do klasyfikacji, tylko 200-klasowy zbiór OiM do detekcji obiektów określany jako ImageNet Object Detection Challenge), Open Images [51], LVIS [31] oraz COCO (wcześniej nazywany Microsoft/MS COCO [59]). Do detekcji obiektów można użyć też zbiorów dedykowanych do segmentacji instancji, takich jak np. zbiór artykułów handlowych D2S [22]. Wiele innych zbiorów można znaleźć na platformie konkursów uczenia maszynowego Kaggle[5].

Do niniejszych badań wybrano walidacyjny podzbiór z konkursu COCO Detection Challenge, edycji 2017, który skrótowo określa się jako `val2017`. Dla konkretnego zbioru OiM, jak zbiór COCO, mogą istnieć jeszcze różne podziały (ang. *split*) na część uczącą i walidacyjną, której można również używać jako testowej. Dzięki temu możliwe jest porównywanie modeli między sobą, ponieważ wszyscy badacze używają tego samego podziału. I tak, podzbiór walidacyjny zbioru COCO w podziale z roku 2014 nazywa się `val2014`.

Zbiór ten zawiera 5 tys. obrazów, które są *nie-ikoniczne* [59], tj. przedstawiają obiekty w naturalnym, złożonym środowisku. Odróżnia to zbiór COCO od zbiorów do klasyfikacji obrazów, które często zawierają obrazy *ikoniczne*, to znaczy zawierające pojedynczy obiekt pewnej klasy, położony centralnie na obrazie, w uproszczonym kontekście.

Do obrazów dołączone są metadane ODW, które należą do 80 różnych klas. Klasy ODW nie są zbalansowane, najliczniej występują postacie ludzkie (klasa `person`) w liczbie 11 004, a kolejne klasy to samochód (`car`, 1932 obiekty) i krzesło (`chair`, 1791 obiektów). Klasa `car` nie obejmuje samochodów terenowych typu *pick-up*, które są razem z ciężarówkami w klasie `truck`. Tak czy owak, już między dwiema najliczniejszymi klasami jest ponad pięciokrotna dysproporcja. Dwie najmniej liczne zawierają po jedenaście i dziewięć obiektów (to tylko dziesięć możliwych wartości TPR: $\{0, \frac{1}{9}, \ldots, \frac{8}{9}, 1\}$). Rozkład częstości klas przypomina rozkład Zipfa, co sprawia trudności w treningu detektorów. Dla oceny detektora również stanowi on pewne wyzwanie, któremu może sprostać wyznaczanie miar skuteczności detekcji osobno dla każdej klasy ODW i uśrednianie wyniku.

Formatem, w którym zapisane są obrazy zbioru testowego, jest JPEG, a rozkład wartości parametru Q jest następujący:

- Q = 96 dla 3540 obrazów,
- Q = 90 dla 1414 obrazów,

[5] `https://www.kaggle.com/` (dostęp: 27 września 2022)

- Q = 80 dla pozostałych 46 obrazów.

Prawie wszystkie obrazy są kolorowe, za wyjątkiem 134 obrazów w skali szarości. Maksymalny rozmiar rastrowy to 640×640 pikseli (łącznie ok. 2 tys. obrazów zapisano w rozmiarach: 640×480, 640×428 i 480×640), czyli w większości są to obrazy o rozmiarze w zakresie 0.3 – 0.4 megapiksela.

Degradacja jakości poprzez kompresję. Krok ten został wykonany stukrotnie, dla każdej wartości $Q \in [1, 100]$. Przed każdą ponowną kompresją obrazy były przywracane do pierwotnej postaci tak, aby efekty ponownej kompresji nie kumulowały się. Wielokrotna kompresja obrazu z różnymi tablicami kwantyzacji przyczynia się do dodatkowych strat jakości, co potwierdzają prace z zakresu wykrywania wielokrotnej kompresji [10, 56]. Do przywrócenia oryginalnych obrazów wykorzystano polecenie:

```
$ unzip -o val2017.zip -d datasets/coco/
```

Opcja `-o` wymusza nadpisanie istniejących plików, a opcja `-d` wskazuje katalog docelowy, który został wybrany tak, aby pasował do konwencji położenia obrazów na dysku stosowanej w bibliotece Detectron2.

Do kompresji obrazów z zadaną wartością parametru Q użyto następującego polecenia:

```
$ mogrify -verbose -quality <Q> datasets/coco/val2017/*.jpg
```

Polecenie `mogrify` to program z pakietu ImageMagick[6], który modyfikuje obrazy *in situ*. Plik `val2017.zip` to oficjalne archiwum dostępne do pobrania na stronie zbioru obrazów COCO[7] (rozmiar ok. 800 MB). Opcja `-verbose` powoduje wyświetlenie informacji o każdej operacji (wczytanie obrazu oryginalnego i zapis obrazu po kompresji), co pozwala monitorować postęp kompresji oraz zmiany rozmiaru plików:

```
$ mogrify -verbose -quality 20 datasets/coco/val2017/*.jpg
000000000139.jpg JPEG 640x426 8-bit sRGB 161811B 0:00.005
000000000139.jpg JPEG 640x426 8-bit sRGB 16712B 0:00.010
000000000285.jpg JPEG 586x640 8-bit sRGB 335861B  0:00.008
000000000285.jpg JPEG 586x640 8-bit sRGB 40595B 0:00.008
(...)
```

Na każdy plik przypadają dwie linie: odczyt i zapis. W kolejnych kolumnach znajdują się: nazwa pliku, format, rozmiar rastrowy, głębia bitowa (8-bit), tryb kolorów (sRGB), rozmiar w bajtach i czas przetwarzania. Takie informacje są podane dla każdego z 5 tys. plików. Dwa pierwsze przykłady pokazują, że użycie kompresji z parametrem Q = 20 skutkuje redukcją rozmiaru o około rząd wielkości (z 161.8 kB do 16.7 kB

[6] `https://imagemagick.org/` (dostęp: 27 września 2022)

[7] `http://images.cocodataset.org/zips/val2017.zip` (dostęp: 27 września 2022)

oraz z 335.9 kB do 40.6 kB). Pierwsze dwa obrazy w zbiorze `val2017` mają numery 139 i 285 – numeracja nie jest ciągła, ponieważ identyfikator jest globalny w całym zbiorze COCO. Unikalne numery mają zarówno obrazy w części `val2017`, jak i `train2017`, oraz obrazy nieużyte w edycji 2017 COCO Detection Challenge.

Opisana operacja jest deterministyczna – wielokrotne wykonanie jej, a także użycie dowolnej innej wersji programu `mogrify`, w tym również na innym sprzęcie, wygeneruje te same wyniki.

Dane o obiektach, które również są dostępne na stronie zbioru COCO[8], nie wymagają żadnej modyfikacji, ponieważ położenie i rozmiar obiektów nie ulega zmianie na skutek degradacji jakości.

Miary jakości obrazów skompresowanych. Po degradacji za pomocą kompresji, zbiór obrazów został zbadany pod kątem pogorszenia jakości z użyciem obiektywnych miar MSE, PSNR i SSIM omówionych w rozdziale „Kompresja obrazów". Ostatnia z tych miar jest najlepsza pod względem zgodności z subiektywnymi miarami jakości obrazu i dlatego została szczegółowo przeanalizowana.

Parametr Q algorytmu JPEG nie determinuje stałej jakości dla obrazów skompresowanych, ale dla każdej jego wartości otrzymywany jest pewien rozkład pięciu tysięcy wartości SSIM (po jednej dla każdego obrazu z `val2017`), który można charakteryzować za pomocą statystyki opisowej stosując miary rozkładu takie jak:

- średnia arytmetyczna i odchylenie standardowe – tzw. miary klasyczne,
- minimum, maksimum, mediana, pierwszy (Q1) i trzeci (Q3) kwartyl – tzw. miary pozycyjne.

Miary te dla wybranych wartości parametru Q znajdują się w Tabeli 6.1.

Graficzną metodą przedstawiania miar pozycyjnych rozkładu jest wykres pudełkowy (inaczej: skrzynkowy, ang. *box plot*). Umożliwia on przedstawienie pięciu miar pozycyjnych w postaci prostokąta z linią wewnątrz i dwoma odchodzącymi od środków boków odcinkami. Dla pionowego wykresu pudełkowego:

- dolny i górny bok pokazują Q1 i Q3,
- linia wewnątrz prostokąta reprezentuje medianę,
- dolny i górny „wąs" to minimum i maksimum.

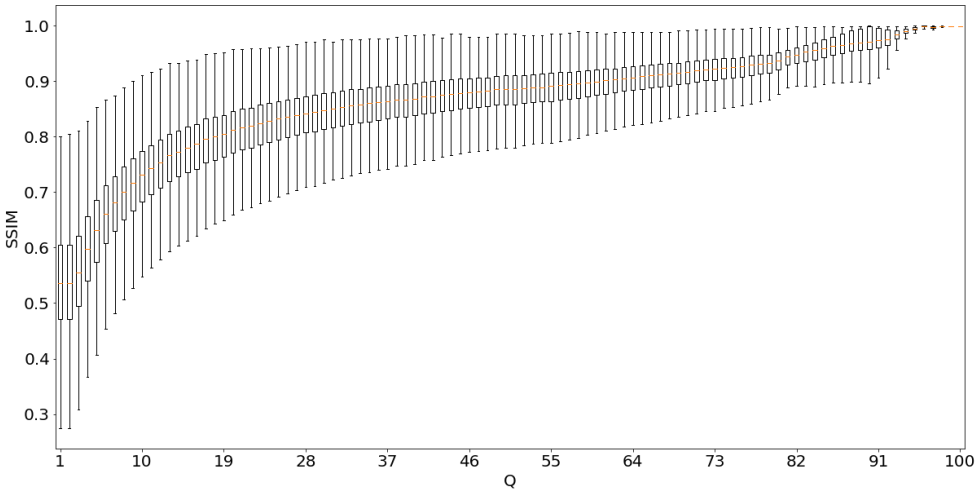
Wysokość prostokąta, tj. różnica Q3 − Q1, jest to tzw. rozstęp kwartylny (inaczej: ćwiartkowy). W ten sposób „pudełko" zawsze zawiera medianę i połowę obserwacji danego rozkładu. Wykresy rozkładów wartości SSIM dla każdej z wartości parametru Q znajdują się na Rys. 6.1.

Wykres na Rys. 6.1 można omawiać przedziałami w kolejności „od prawej do lewej", tzn. od najwyższej do najniższej wartości Q. W przedziale wartości parametru Q od 94

[8] `http://images.cocodataset.org/annotations/annotations_trainval2017.zip` (dostęp: 27 września 2022)

Tabela 6.1: **Statystyka opisowa SSIM dla wybranych wartości Q.** Uwzględniono $Q \in [5, 95]$ z krokiem 10.

Q	SSIM						
	średnia	odch. std.	min	Q1	mediana	Q3	max
5	0.6271	0.0898	0.1533	0.5741	0.6309	0.6860	0.9657
15	0.7738	0.0637	0.3593	0.7360	0.7802	0.8176	0.9694
25	0.8262	0.0546	0.3900	0.7945	0.8320	0.8643	0.9795
35	0.8549	0.0484	0.4191	0.8276	0.8600	0.8883	0.9760
45	0.8740	0.0436	0.4440	0.8498	0.8785	0.9039	0.9910
55	0.8868	0.0413	0.4674	0.8645	0.8915	0.9150	0.9931
65	0.9047	0.0358	0.5045	0.8865	0.9090	0.9287	0.9881
75	0.9211	0.0331	0.5768	0.9061	0.9256	0.9424	0.9972
85	0.9562	0.0249	0.7819	0.9432	0.9597	0.9757	0.9985
95	0.9947	0.0026	0.9808	0.9931	0.9947	0.9969	0.9996

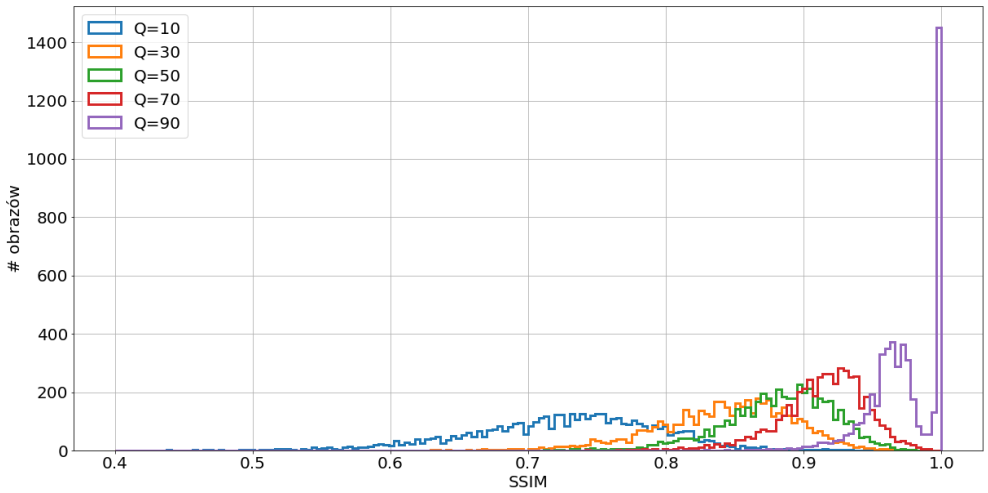


Rys. 6.1: **Rozkłady wartości SSIM dla różnych wartości parametru Q.** Wykresy pudełkowe miary SSIM skompresowanych obrazów względem oryginałów.

do 100 spadek jakości jest bardzo niski, w przybliżeniu można uznać taką kompresję za bezstratną. W przedziale średnich wartości parametru Q, od około 20 do około 80 mediana i kwartyle zmieniają się w przybliżeniu liniowo. Spadek jakości przyspiesza poniżej $Q = 30$ (mediana SSIM około 0.8), szczególnie poniżej $Q = 16$ (mediana SSIM ca. 0.75). Kompresja z parametrem Q mniejszym niż 5 daje w wyniku ponad połowę obrazów o SSIM niższym od 0.6.

Dokładniejszym od wykresu pudełkowego sposobem przedstawienia informacji o rozkładzie SSIM jest histogram. Wybrane rozkłady SSIM obrazów skompresowanych

w stosunku do oryginału zostały przedstawione z użyciem histogramów na Rys. 6.2.



Rys. 6.2: **Histogramy ilościowe wartości SSIM dla wybranych wartości Q.** Uwzględniono $Q \in [10, 90]$ z krokiem 20.

Za wyjątkiem $Q = 90$ (dla którego jest dominanta przy wartości $SSIM = 1$), są to w przybliżeniu symetryczne rozkłady, które ze spadkiem wartości parametru Q mają niższą średnią i są silniej spłaszczone (występuje słabsze skupienie pogorszenia jakości wokół średniej wartości SSIM).

Współczynnik kompresji obrazów. Innym aspektem degradacji obrazów, który został zmierzony, jest rozmiar danych po kompresji. Wzięty pod uwagę został rozmiar plików wyjściowych, co uwzględnia pewien narzut na format pliku (nagłówek, tablice kwantyzacji itp.), ale taka wielkość ma znaczenie praktyczne, bo właśnie w takiej postaci skompresowane dane są zapisywane. Zamiast **stopnia** kompresji (ang. *compression rate*), będącego ilorazem rozmiaru danych nieskompresowanych do skompresowanych (wartość większa od jeden w przypadku skutecznej kompresji), rozpatrywano jego odwrotność – **współczynnik** kompresji. Za rozmiar danych nieskompresowanych przyjęto „surowy" rozmiar obrazu, tj. iloczyn wysokości, szerokości i liczby kanałów obrazu.

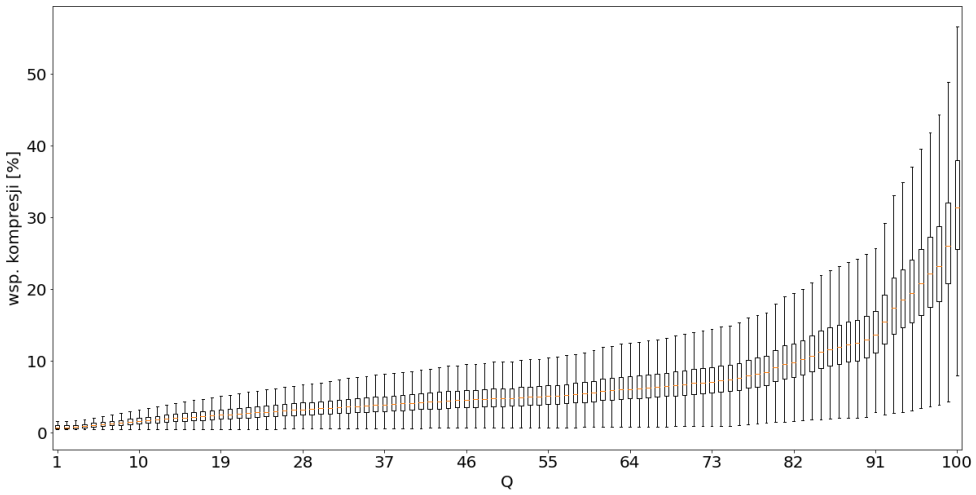
Współczynnik kompresji dla każdej wartości parametru Q ma inny rozkład. Względny rozmiar plików wyjściowych (wyrażony w procentach) został przedstawiony w Tabeli 6.2.

Rozkład względnych rozmiarów (w stosunku do surowego rozmiaru obrazu, przy trzech bajtach na piksel) w funkcji parametru jakości JPEG przedstawiono na Rys. 6.3.

Porównując wykresy na Rys. 6.1 i Rys. 6.3 można stwierdzić, że jest między nimi charakterystyczne podobieństwo, jednak rozkłady sprawiają wrażenie, jakby jeden z

Tabela 6.2: **Statystyka opisowa współczynnika kompresji dla wybranych wartości Q.** Uwzględniono Q $\in [5, 95]$ z krokiem 10.

Q	Współczynnik kompresji w %						
	średnia	odch. std.	min	Q1	mediana	Q3	max
5	1.20	1.18	0.44	0.80	1.02	1.32	68.80
15	2.32	1.48	0.48	1.62	2.09	2.70	69.59
25	3.27	1.82	0.51	2.29	2.98	3.85	70.25
35	4.12	2.14	0.57	2.90	3.77	4.89	70.84
45	4.91	2.45	0.65	3.48	4.51	5.84	71.37
55	5.55	2.69	0.69	3.94	5.10	6.55	71.86
65	6.71	3.12	0.82	4.84	6.20	7.97	72.52
75	7.96	3.59	0.97	5.75	7.37	9.42	73.55
85	12.00	4.86	1.87	8.95	11.23	14.18	76.02
95	20.47	7.64	3.04	15.35	19.50	24.12	83.83



Rys. 6.3: **Rozkłady współczynnika kompresji dla różnych wartości Q.** Wykresy pudełkowe stosunku rozmiaru skompresowanych obrazów do rozmiaru zdekompresowanych oryginałów.

nich powstał przez obrócenie drugiego o 180 stopni – wraz ze zmniejszaniem parametru Q od wartości 100 następuje gwałtowny spadek rozmiaru plików, ale rozrzut tego rozmiaru jest na początku duży, a następnie coraz mniejszy. Dla najniższych wartości Q rozmiar jest skupiony i ma niską medianę. Indeks podobieństwa strukturalnego również jest rosnący wraz ze wzrostem Q, ale odwrotnie – dla wysokich Q jest skupiony i wysoki, w środku przedziału możliwych wartości Q ma średnie skupienie i nieznaczne zmiany mediany, a dla najniższych występuje silny spadek i wzrost rozrzutu wartości

SSIM.

Obserwacje dotyczące miar jakości stanowią argument za dokonanym wyborem – analizowania zachowania modeli przede wszystkim w funkcji parametru Q, który jest pod kontrolą użytkownika algorytmu kompresji, a nie w oparciu o miary pogorszenia jakości, których określenie jest możliwe dopiero *a posteriori*.

Miary skuteczności detekcji. Po każdym cyklu odtworzenia oryginalnych obrazów i degradacji ich jakości dla kolejnej wartości parametru Q, zdegradowane obrazy stanowiły wejście do inferencji przez wszystkie badane modele. Informacje o wynikach detekcji obiektów (tj.: identyfikator obrazu, współrzędne prostokąta, indeks klasy i ufność dla każdego wykrytego obiektu) zostały zapisane do pliku w formacie JSON o nazwie `coco_instances_results.json`. Pliki te zostały zapisane w dwupoziomowej hierarchii folderów – nadrzędny folder dla modelu, wewnątrz podfoldery dla poszczególnych wartości parametru Q. Po utworzeniu plików archiwów zachowujących tę hierarchię wyniki zostały upublicznione w repozytoriach danych Harvard Dataverse odpowiednich dla każdego z etapów badań.

Na podstawie plików JSON zawierających WDO zostały wyznaczone dla modeli następujące miary skuteczności detekcji:

- TP: liczba poprawnych WDO (*true positive*),
- FP: liczba fałszywych WDO (*false positive*) – zwróconych przez detektor, ale nieposiadających odpowiadających im ODW w zbiorze testowym,
- EX: liczba WDO nadmiarowych (*extra*) – są to WDO znajdujące się w obszarach typu *crowd*, tj. kolekcjach obiektów tej samej klasy, w których nie ma wyróżnienia poszczególnych ODW. WDO typu EX mogą być albo poprawne, albo niepoprawne,
- PPV: precyzja wykrytych obiektów,

$$\mathrm{PPV} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP}), \tag{6.1}$$

- TPR: czułość detekcji obiektów,

$$\mathrm{TPR} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FN}), \tag{6.2}$$

- F1: miara F1 (ang. *F1-score, Dice index*), inaczej „współczynnik Sørensena” albo „indeks Czekanowskiego”, która jest średnią harmoniczną TPR i PPV,

$$\mathrm{F1} = 2 \cdot \mathrm{PPV} \cdot \mathrm{TPR}/(\mathrm{TPR} + \mathrm{PPV}) = 2\ \mathrm{TP}/(\mathrm{FN} + 2\ \mathrm{TP} + \mathrm{FP}), \tag{6.3}$$

- AP_{50}, AP_{75} – miary średniej precyzji (AP) dla pojedynczego progu nakładania WDO i ODW (tzw. próg IoU), wcześniej (np. w PASCAL VOC [20]) określane jako mAP, dla dwóch różnych progów $T_{IoU} \in \{0.5, 0.75\}$,

- AP: miara średniej precyzji uśredniona dla progów $T_{IoU} \in [0.5, 0.95]$ z krokiem 0.05 [59],
- $\mathrm{AP_s}$, $\mathrm{AP_m}$, $\mathrm{AP_l}$: miary średniej precyzji dla podzbiorów obiektów:
  - $\mathrm{AP_s}$ – małych (poniżej $32^2 = 1024$ pikseli powierzchni),
  - $\mathrm{AP_m}$ – średnich (od 1024 do $96^2 = 9216$ pikseli),
  - $\mathrm{AP_l}$ – dużych (powyżej 9216 pikseli).

  Przy wyznaczaniu tych miar nie są brane pod uwagę ODW o rozmiarze spoza zakresu, a WDO o niepasującym rozmiarze są pomijane (nie będą ani TP, ani FP, ani EX), nawet gdyby spełniały warunek progu T_{IoU} dla pewnego branego pod uwagę ODW.

Dla miar zależnych od T_c, tj. TP, FP, EX, PPV, TPR i F1, przyjęto wartość progu $T_{IoU} = 0.5$ (taki sam jak dla miary $\mathrm{AP_{50}}$).

Wyniki miar zostały wyznaczone za pomocą oficjalnej biblioteki COCO API[9] (wszystkie miary średniej precyzji) oraz za pomocą własnych narzędzi napisanych do tego celu przez autora rozprawy[10] (miary TP, FP, EX, TPR, PPV oraz F1). Oprócz tego miary AP, $\mathrm{AP_{50}}$ i $\mathrm{AP_{75}}$ zostały niezależnie wyznaczone z użyciem tych narzędzi, a następnie pozytywnie zweryfikowane względem wartości obliczonych z użyciem biblioteki COCO API.

Testowane modele. Do przebadania wybrano dziewięć modeli detekcji obiektów, które są dostępne w bibliotece Detectron2 [103]. Modelom zostały przypisane identyfikatory, używane w dalszej części tekstu do łatwej identyfikacji: R101, R101_C4, R101_DC5, R101_FPN, R50, R50_C4, R50_DC5, R50_FPN, X101. Ich zestawienie znajduje się w Tabeli 6.3. Kręgosłupy modeli z końcówkami C4 i DC5 nie zawierają FPN, tzn. nie generują piramidy cech głębokich, a końcówka DC5 oznacza użycie jako kręgosłupa sieci ResNet, w której część operacji konwolucji zastąpiono konwolucjami rozszerzonymi (ang. *dilated convolution*, DC).

Wagi połączeń sieci neuronowych dla modeli zostały pozyskane z udostępnionej w Internecie kolekcji *Detectron2 Model Zoo*[11] (dosł. „Zoo modeli projektu Detectron2"). Plik, w którym zapisane są wagi połączeń sieci, jest określany jako *snapshot* (dosł. „migawka" – nazwa ta pochodzi od „fotograficznego" utrwalenia stanu modelu w trakcie treningu).

Na stronie *Detectron2 Model Zoo* znajdują się pliki wag połączeń dla modeli detekcji obiektów zapisane w formacie odpowiednim dla biblioteki PyTorch. Trening tych

[9] `https://github.com/cocodataset/cocoapi` (dostęp: 27 września 2022)

[10] `https://github.com/tgandor/urban_oculus` (dostęp: 27 września 2022)

[11] `https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md` (dostęp: 27 września 2022)

Tabela 6.3: **Głębokie modele detekcji obiektów użyte w badaniu.**

Symbol	Opis
R101	RetinaNet [58], kręgosłup ResNet-101 [36] + FPN [57]
R101_C4	Faster R-CNN [76], kręgosłup ResNet-101 [36]
R101_DC5	Faster R-CNN [76], kręgosłup ResNet-101 [36] + DC [106]
R101_FPN	Faster R-CNN [76], kręgosłup ResNet-101 [36] + FPN [57]
R50	RetinaNet [58], kręgosłup ResNet-50 [36] + FPN [57]
R50_C4	Faster R-CNN [76], kręgosłup ResNet-50 [36]
R50_DC5	Faster R-CNN [76], kręgosłup ResNet-50 [36] + DC [106]
R50_FPN	Faster R-CNN [76], kręgosłup ResNet-50 [36] + FPN [57]
X101	Faster R-CNN [76], kręgosłup ResNeXt-101 [104] + FPN [57]

modeli został przeprowadzony przez Facebook AI Research za pomocą maszyn Big Basin[12], które są wyposażone w osiem akceleratorów Nvidia Tesla P100, o mocy obliczeniowej 10.6 TFLOPS każdy. W etapie I eksperymentów obliczeniowych wykorzystano modele pre-trenowane w harmonogramie oznaczonym w Detectron2 jako „3x", tj. 270 tys. iteracji przy 16 obrazach przetwarzanych w każdym kroku. Współczynnik uczenia (ang. *learning rate*) był zmniejszany dziesięciokrotnie po 210 tys. i 250 tys. iteracji.

Zestaw modeli wybrany do przeprowadzenia eksperymentów charakteryzuje się różnorodnością, ponieważ obejmuje on:

- detektory jednoetapowe (RetinaNet: R50, R101) i dwuetapowe (pozostałe),
- detektory niekorzystające z FPN (R50_C4, R101_C4, R50_DC5 i R101_DC5) oraz korzystające z FPN (pozostałe),
- kręgosłupy o różnej głębokości – ResNet-50 i ResNet-101 (według liczby w identyfikatorze),
- kręgosłup korzystający z konwolucji grupowanych (grupowanie kanałów w sieci ResNeXt-101, model X101), kręgosłupy korzystające z pełnych konwolucji we wszystkich warstwach (pozostałe),
- (wśród modeli bez FPN) zarówno modele z użyciem konwolucji rozszerzonych w końcowym (piątym) bloku warstw sieci ResNet (modele R50_DC5 i R101_DC5), jak i modele ze wszystkimi konwolucjami o rozmiarze jądra 3×3.

Wszystkie modele zostały wytrenowane na zbiorze `train2017`, to znaczy treningowym podzbiorze z konkursu COCO Detection Challenge, edycja 2017. Jest to zbiór treningowy, dla którego używany w tej pracy zbiór `val2017` jest zbiorem walidacyjnym. Dzięki temu modele mogą wykrywać obiekty wszystkich osiemdziesięciu klas, które znajdują się w użytym zbiorze testowym.

[12] `https://engineering.fb.com/2017/03/08/data-center-engineering/introducing-big-basin-our-next-generation-ai-hardware/` (dostęp: 27 września 2022)

6.3 Wyniki bazowe modeli pre-trenowanych

Jako punkt odniesienia wyznaczono miary skuteczności detekcji na niezmodyfikowanym zbiorze `val2017`. Przeprowadzono dwukrotnie detekcję obiektów, przyjmując różne progi ufności:

- $T_c = 0.5$ – neutralna wartość *a priori*, która oznacza wskazanie przez model, że dany WDO jest TP z wyższym prawdopodobieństwem niż FP,
- $T_c = 0.05$ – domyślna w bibliotece Detectron2 niska wartość progu, która ma na celu maksymalizację miar średniej precyzji.

Zliczanie WDO, precyzja, czułość i F1. Wyniki bazowe miar zależnych od wyboru progu ufności T_c zostały przedstawione w Tabeli 6.4 (próg $T_c = 0.5$) oraz w Tabeli 6.5 (próg $T_c = 0.05$).

Tabela 6.4: **Bazowe wyniki miar TPR, PPV, F1, TP, FP i EX dla $T_c = 0.5$.**

Model	PPV %	TPR %	F1 %	TP	FP	EX
R101	**81.1**	51.7	63.1	18769	<u>4360</u>	820
R101_C4	58.6	<u>67.1</u>	62.5	<u>24373</u>	17246	4463
R101_DC5	58.2	**68.0**	62.7	**24701**	17752	4504
R101_FPN	68.7	64.9	<u>66.8</u>	23593	10751	2605
R50	<u>80.7</u>	49.7	61.5	18043	**4320**	821
R50_C4	56.8	65.3	60.7	23733	18075	<u>4586</u>
R50_DC5	57.1	66.7	61.6	24244	18190	**4668**
R50_FPN	67.0	63.8	65.3	23170	11428	2763
X101	69.7	66.3	**67.9**	24073	10472	2534

Najlepszy wynik w kolumnie został **pogrubiony**, drugi najlepszy – <u>podkreślony</u>.
TPR, PPV, F1 – czułość, precyzja i F1.
TP, FP – liczba prawdziwych i fałszywych WDO.
EX – liczba WDO pomijanych w obliczeniach miar skuteczności detekcji.

Można zaobserwować różnorodność modeli w tym zestawieniu – dla każdej miary najlepszy jest inny model (z wyjątkiem miar TP i TPR, które są ściśle związane ze sobą i tym samym mają identyczny ranking). Wyniki bazowe miar zależnych od przyjętego progu ufności pokazują następujące własności badanych modeli:

- modele RetinaNet (R50, R101) zwracają przy tak przyjętym progu ufności znacznie mniej obiektów, ale za to ze znacznie wyższą precyzją,
- najwięcej obiektów spośród wszystkich modeli zwracają te modele Faster R-CNN, które nie zawierają FPN,

- pod względem miary F1, która stanowi pewne uogólnienie, dwoma najlepszymi modelami są kolejno X101 i R101_FPN, czyli modele dwuetapowe korzystające z FPN i głębokiego, 101-warstwowego kręgosłupa,
- najwięcej dodatkowych WDO (EX) zwracają modele dwuetapowe bez FPN, przy czym ciekawe jest to, że więcej „neutralnych" WDO zwracają modele z płytszym, 50-warstwowym kręgosłupem (R50_C4 i R50_DC5).

Wartości miar dla progu ufności $T_c = 0.05$, który jest sztucznie niski dla tych miar, znajdują się w Tabeli 6.5.

Tabela 6.5: **Bazowe wyniki miar TPR, PPV, F1, TP, FP i EX dla** $T_c = 0.05$**.**

Model	PPV %	TPR %	F1 %	TP	FP	EX
R101	8.4	**83.2**	15.2	**30238**	330518	**15319**
R101_C4	18.8	81.3	30.5	29531	127795	13591
R101_DC5	19.2	81.3	31.1	29530	124023	14152
R101_FPN	24.2	82.0	37.3	29786	93415	12744
R50	7.8	82.9	14.2	30112	358261	15188
R50_C4	17.3	79.8	28.5	29004	138201	13422
R50_DC5	18.4	80.3	29.9	29178	129690	14221
R50_FPN	21.5	81.5	34.0	29596	108119	13205
X101	**29.8**	81.3	**43.6**	29546	**69608**	11862

Najlepszy wynik w kolumnie został **pogrubiony**, drugi najlepszy – podkreślony.
TPR, PPV, F1 – czułość, precyzja i F1.
TP, FP – liczba prawdziwych i fałszywych WDO.
EX – liczba WDO pomijanych w obliczeniach miar skuteczności detekcji.

Dla tak niskiego progu ufności można poczynić nieco inne spostrzeżenia:

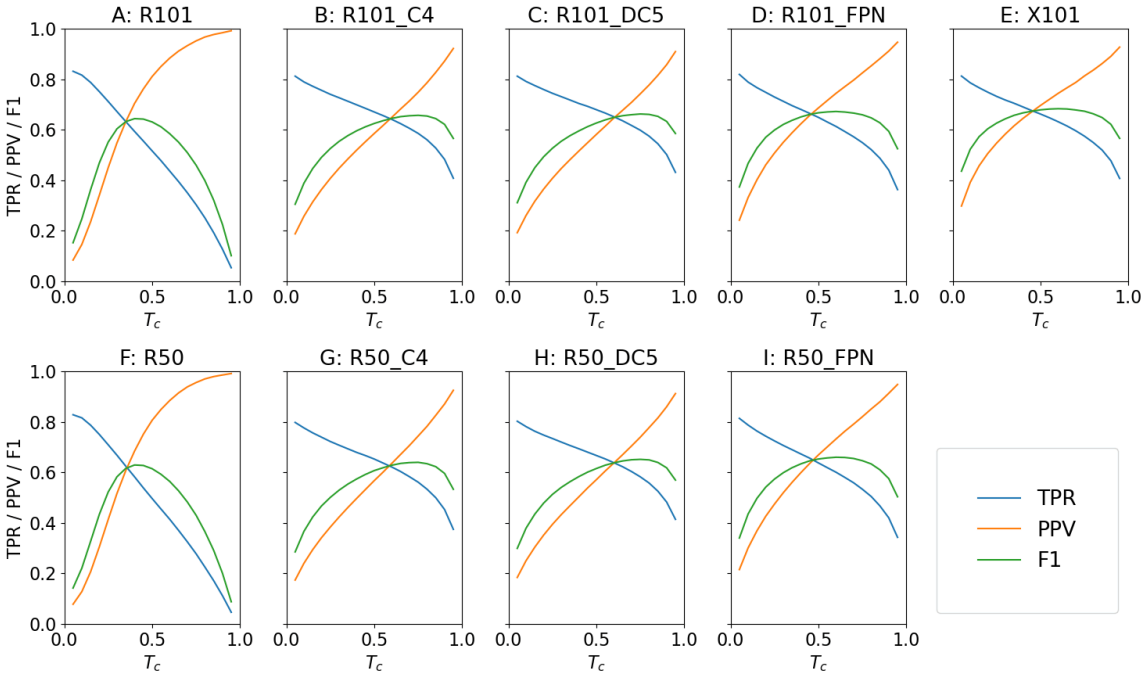
- końcowa czułość modeli jest dosyć zbliżona – wynosi około 80% – ale kolejność modeli jest zupełnie inna: tym razem najwięcej uzyskują modele jednoetapowe (83%, niezależnie od głębokości kręgosłupa), modele dwuetapowe z FPN są następne (81–82%), a na końcu są modele dwuetapowe bez FPN, i w ich przypadku ma znaczenie głębokość kręgosłupa: 101-warstwowe mają TPR na poziomie 81%, a 50-warstwowe – 80%,
- precyzja przy niskim progu ufności jest oczywiście niska, ale występuje dużo silniejsze zróżnicowanie niż w przypadku czułości: model X101 ma zdecydowanie najwyższą precyzję 30%, a kolejne miejsca w rankingu zajmują pozostałe modele dwuetapowe z FPN, a głębokość ich kręgosłupa ma pewne znaczenie (PPV wynosi, odpowiednio, ponad 24% i 21.5% dla R101_FPN i R50_FPN),
- precyzja modeli dwuetapowych bez FPN stawia tę grupę na trzecim miejscu rankingu i wynosi 17–19%, przy czym precyzyjniejsze są modele DC5 (ich przewaga

nad modelami C4 w czułości była marginalna), a w obu typach tych modeli głębszy kręgosłup daje PPV wyższe o około 1.5 punktu procentowego,

- zdecydowanie najniższa precyzja występuje dla modeli jednoetapowych, których PPV wynosi 8–8.5%, z niewielkim wpływem głębokości kręgosłupa,
- w przypadku F1 dominujący wpływ na wynik tej miary ma PPV, która jest związana z najlepszą (najniższą) liczbą FP; mimo to, na czele rankingu znajdują się te same modele co przy $T_c = 0.5$, tj. X101 i R101_FPN,
- najwięcej dodatkowych WDO (EX) zwracają sieci jednoetapowe, i dla tego niskiego progu ufności ich liczba zachowuje się podobnie jak TP.

Tak istotnie różne zachowanie modeli (ranking, względny stosunek miar) dla dwóch wartości progu ufności stanowiło sugestię, aby bardziej szczegółowo zbadać wartości miar skuteczności detekcji w szerokim zakresie progu T_c.

Bazowe miary skuteczności detekcji w funkcji progu ufności. Wykres zależności miar TPR, PPV i F1 od progu ufności T_c znajduje się na Rys. 6.4.



Rys. 6.4: **Wartości TPR, PPV i F1 dla bazowych modeli w funkcji T_c.** Precyzja i czułość są zależne od progu ufności T_c. F1 to ich średnia harmoniczna. Głębokości kręgosłupa: 101 (górny wiersz), 50 (dolny wiersz). (A), (F) – RetinaNet. (B), (C), (G), (H) – Faster R-CNN bez FPN. (D), (I), (E) – Faster R-CNN z FPN.

Na wykresie widoczne jest przede wszystkim podobieństwo wewnątrz grup modeli (podobne modele w tej samej kolumnie, różniące się głębokością kręgosłupa) – o ile

dokładne wartości miar różnią się o pojedyncze punkty procentowe, o tyle charakter zależności miar od progu ufności jest zachowany. Dla wszystkich modeli występuje kompromisowy, przeciwstawny wpływ progu ufności T_c na wartości miar TPR i PPV – TPR jest maksymalne dla $T_c = 0.05$ (najniższa zbadana wartość progu), a minimalne dla $T_c = 0.95$ (najwyższa zbadana wartość progu), natomiast PPV – odwrotnie. Zwracają uwagę następujące fakty:

- Modele jednoetapowe (R101 i R50, pierwsza kolumna wykresu) mają szerokie przedziały wartości TPR i PPV. Dla maksymalnej wartości T_c mają najwyższą wartość PPV i najniższą wartość TPR wśród wszystkich badanych modeli. Dla minimalnej wartości tego progu mają minimalną wartość PPV wśród wszystkich modeli, ale jednocześnie maksymalna wartość TPR, którą te modele wtedy uzyskują, jest zbliżona do wartości uzyskiwanej przez inne modele dla tego progu.

  Ze wzrostem progu czułość spada w sposób jednostajny (liniowo), a precyzja rośnie niejednostajnie – ze wzrostem progu rośnie najpierw szybko, a następnie łagodnie. Przebieg wartości F1 przypomina parabolę z maksimum w pobliżu wartości $T_c = 0.5$.

- Modele dwuetapowe Faster R-CNN (tj. wszystkie pozostałe) już od niskiej wartości progu T_c wykazują istotną precyzję (19–30%), ale dla maksymalnych wartości T_c ich czułość nie spada do zera. Oznacza to, że modele zwracają znaczną liczbę TP z wysokimi wartościami ufności (bliskimi jedności), a FP zwracanych z wysoką ufnością jest niewiele.

- Punkt, w którym precyzja i czułość osiągają równe wartości (przecięcie wszystkich trzech linii na wykresach), wypada w różnych miejscach dla różnych grup modeli: dla RetinaNet przy progu ok. 0.4, dla Faster R-CNN bez FPN ok. 0.6, a z FPN ok. 0.5 i nie jest to punkt maksymalnej wartości F1. Maksymalne F1 znajduje się „na prawo" (wyższy próg ufności) od tego „punktu potrójnego" ze względu na większy wzrost precyzji przy mniejszym spadku czułości.

- Przebieg krzywej F1 jest różny w zależności od grupy modeli: dla RetinaNet jest „stromy", z wąskim przedziałem wartości zbliżonych do maksimum, natomiast dla modeli dwuetapowych przebieg jest „łagodny", z szerokim przedziałem wartości bliskich maksymalnej. Przedział ten różni się nachyleniem – dla grup modeli C4 i DC5 jest ono największe (wyższe F1 dla wyższego T_c). Dla dwóch modeli z kręgosłupem ResNet + FPN (R101_FPN i R50_FPN) nachylenie jest bardzo nieznaczne, a wartość F1 dla modelu X101 (ResNeXt + FPN) ma w przybliżeniu *plateau* w zakresie od $T_c = 0.2$ do $T_c = 0.8$, które jest symetryczne względem poziomej osi $T_c = 0.5$.

Zaobserwowane różnice pokazują, że w zastosowaniu praktycznym wybór progu ufności dla detekcji obiektów jest ważny, szczególnie dla modeli, które są na ten próg bardzo wrażliwe (np. RetinaNet). W przybliżeniu stała wartość F1 w pewnym przedziale T_c nie oznacza, że wybór progu z tego przedziału nie ma znaczenia – oznacza

natomiast, że kompromis pomiędzy czułością i precyzją w tym przedziale jest „sprawiedliwy". W takiej sytuacji ostateczny wybór może być podyktowany tym, który rodzaj skuteczności detekcji jest ważniejszy w konkretnym zastosowaniu.

Można również ocenić arbitralne przyjęcie progu $T_c = 0.5$ do oceny miar opartych na liczbach TP i FP jako poprawne – wprawdzie maksymalizuje on F1 tylko dla modeli dwuetapowych z FPN, ale jest akceptowalny dla pozostałych modeli dwuetapowych (które maksimum F1 osiągają przy silniejszym „odcięciu" WDO o niskiej ufności, tj. wyższym T_c), a sieci przy tym progu mają nieznacznie zaniżoną czułość, w zamian za podwyższoną precyzję.

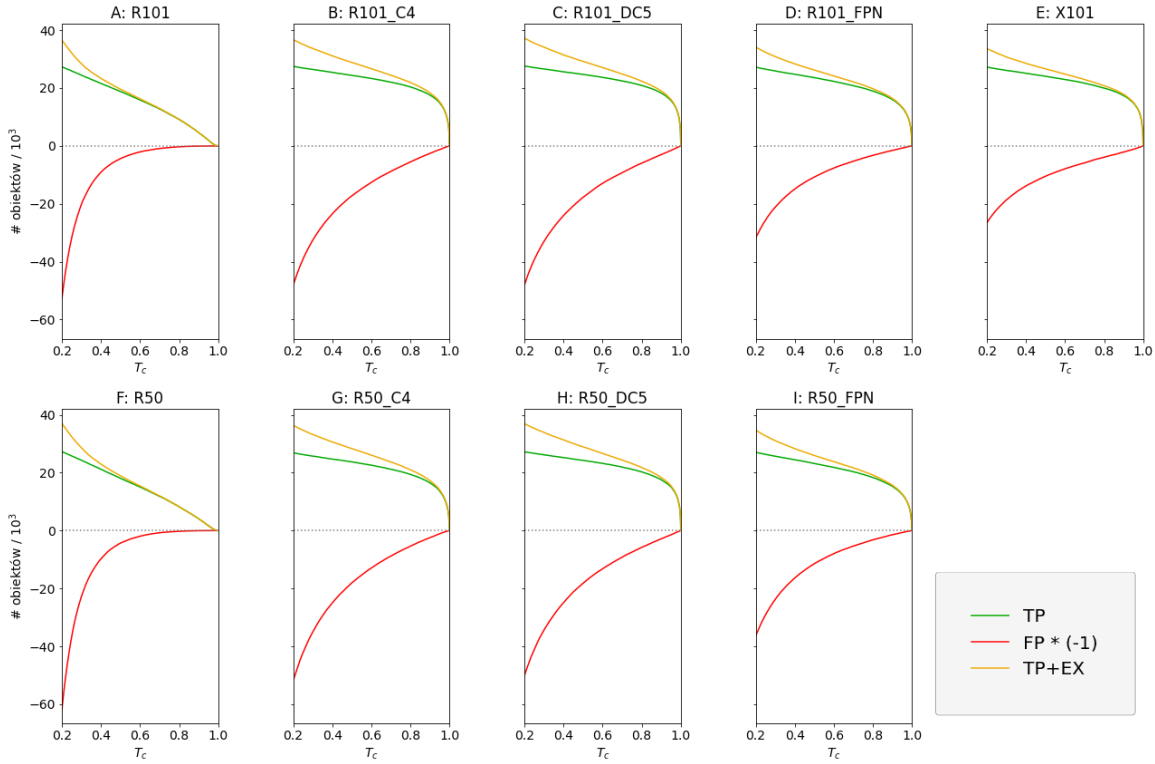
W tym miejscu można zapoznać się z bezwzględnymi liczbami poszczególnych rodzajów WDO, czyli miarami TP, FP i EX, jako funkcjami wartości T_c. Liczba ODW w użytym zbiorze OiM, COCO `val2017`, wynosi 36 335, natomiast obszarów zbiorczych (typu *crowd*) jest 446. Dodatkowa komplikacja bierze się z faktu, że obszary zbiorcze nie są pozbawione ODW, czego przykład można zobaczyć na Rys. 6.5.

Rys. 6.5: **Przykład obrazu z ODW obecnymi na obszarze zbiorczym.**
Na pierwszym planie znajdują się dwie osoby, ale większość górnej połowy zdjęcia jest oznaczona jako obszar zbiorczy dla klasy `person`. Niektóre rozmyte sylwetki osób w tym obszarze również są oznaczone jako ODW (podpisane prostokąty u góry po prawej). Źródło: `http://images.cocodataset.org/val2017/000000448263.jpg` (dostęp: 27 września 2022), licencja: CC BY-NC 2.0.

Dowolny WDO, którego prostokąt jest zawarty w takim obszarze zbiorczym, jeżeli nie zostanie dopasowany do żadnego ODW, staje się wynikiem nadmiarowym (EX) – nie jest liczony ani jako TP, ani jako FP. Oprócz tego, WDO, których prostokąty nie są zaznaczone dokładnie, przy odpowiednim podwyższeniu progu T_{IoU} (co ma miejsce przy obliczaniu miary AP), przestają być zliczane jako TP, ale liczba FP nie zwiększa się – takie WDO zostają zaliczone do EX (poza obszarem zbiorczym byłby to FP).

Wykresy miar zliczania WDO w funkcji progu ufności zostały pokazane na Rys. 6.6. Zastosowano pewien zabieg wizualny – jest to rozmieszczony po obu stronach osi poziomej wykres warstwowy (ang. *stacked plot*): zielona linia to liczba TP, a żółta to suma TP i EX, czyli łącznie wszystkich WDO, które nie są nieprawidłowe (a liczba EX to

pionowa odległość od linii zielonej do żółtej). FP, jako fałszywe detekcje zostały narysowane jako ujemne. W ten sposób pionowa odległość linii żółtej od czerwonej obrazuje liczbę wszystkich zwróconych WDO przy danym progu ufności.



Rys. 6.6: **Liczba obiektów wykrytych na oryginalnym zbiorze w funkcji T_c.** Przedstawiono liczby WDO poprawnych (TP), błędnych (FP, jako ujemne) i nadmiarowych (EX). Przyjęta granica $\min(T_c) = 0.2$ z powodu „eksplozji" FP poniżej tego progu. Głębokości kręgosłupa: 101 (górny wiersz), 50 (dolny wiersz). (A), (F) – RetinaNet; (B), (C), (G), (H) – Faster R-CNN bez FPN; (D), (I), (E) – Faster R-CNN z FPN.

Skala wszystkich wykresów na Rys. 6.6 jest jednakowa, stąd dobrze widoczne są wzajemne relacje pomiędzy poszczególnymi modelami. Należy analizować te wykresy „od prawej do lewej", tj. od maksymalnej wartości $T_c = 1$ do niższych wartości progu, które zwiększają liczbę WDO branych pod uwagę.

Ponownie potwierdza się podobieństwo wewnątrz rodziny modeli – modele RetinaNet w przedziale ufności $[0.6, 1]$ prawie nie zwracają fałszywych WDO, ale liczba TP (a zatem czułość) jest w tym przedziale niemalże liniowo zależna od progu T_c, przy czym jest to zależność ujemna.

Natomiast dwuetapowe modele Faster R-CNN przede wszystkim szybko zwiększają liczbę poprawnych WDO już w przedziale $[0.9, 1]$ przy bardzo niewielkiej liczbie FP. Dalsze obniżenie progu, w przedziale $[0.4, 0.9]$ wiąże się z liniowym przyrostem FP, oraz prawie liniowym przyrostem TP, który stopniowo wykazuje tendencję do przejścia w

linię poziomą (trend rosnący jest nadal widoczny dla sumy TP+EX). Poniżej progu $T_c = 0.4$ następuje przyspieszenie wzrostu FP, mniejsze niż w sieciach RetinaNet, ale powodujące znaczne wyprzedzenie liczby TP przez FP.

Górna część wykresu wszystkich siedmiu modeli dwuetapowych jest prawie identyczna (modele bez FPN w kręgosłupie mają nieco większą liczbę EX). Tym, co je odróżnia, jest przebieg linii reprezentującej FP – dla modelu X101 tendencja wzrostowa jest bardzo łagodna i wartości FP (jako bezwzględne wartości współrzędnych y punktów wykresu) nie przewyższają odpowiadających im wartości TP na długim odcinku poniżej $T_c = 0.4$. Dopiero dla $T_c = 0.05$, czego nie pokazano na wykresie, wartość FP jest dla modelu X101 dwukrotnie większa od TP.

Przeprowadzona analiza może stać się podstawą do wyciągnięcia wniosków ważnych dla inżynierów uczenia maszynowego, którzy będą potrzebowali wybrać model i ustalić dla niego próg ufności w konkretnych zastosowaniach praktycznych. Wnioski takie przedstawiono w rozdziale 8.

Miary średniej precyzji. Wyniki bazowe miar niezależnych od progu ufności T_c, czyli miar z rodziny AP, zostały zebrane w Tabeli 6.6. W przypadku prawie wszystkich miar dominuje model X101 – zwykle jest na pierwszym miejscu, a jedynie dla dużych obiektów (AP_l) zajmuje drugie miejsce w rankingu.

Tabela 6.6: **Bazowe wyniki miar AP dla obrazów oryginalnych,** $T_c = 0.05$**.**

Model	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s
R101	40.4	60.3	43.2	52.2	44.3	24.0
R101_C4	41.1	61.4	44.1	**55.9**	45.5	22.2
R101_DC5	40.6	61.7	43.9	54.1	45.1	22.9
R101_FPN	<u>42.0</u>	<u>62.5</u>	<u>45.9</u>	54.6	<u>45.6</u>	<u>25.2</u>
R50	38.7	58.0	41.5	50.3	42.3	23.3
R50_C4	38.4	58.7	41.3	53.1	42.7	20.7
R50_DC5	39.1	60.5	42.3	52.6	43.5	21.4
R50_FPN	40.2	61.0	43.8	52.0	43.5	24.2
X101	**43.0**	**63.7**	**46.9**	<u>54.9</u>	**46.1**	**27.2**

Najlepszy wynik w kolumnie został **pogrubiony**, drugi najlepszy – <u>podkreślony</u>.
AP_l, AP_m, AP_s – miary AP dla obiektów dużych, średnich i małych.

Wyniki w Tabeli 6.6 zostały uzyskane przy $T_c = 0.05$, co jest domyślnym progiem przy testowaniu modelu za pomocą biblioteki Detectron2. Potwierdzają to wartości głównej miary AP, które są równe wartościom w tabelach na stronie, na której modele zostały udostępnione[13].

Inaczej niż dla miar opartych na liczeniu TP i FP, wartość FP nie ma wpływu na miary z rodziny AP, dlatego obniżenie progu ufności nie prowadzi nigdy do pogorszenia

[13] `https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md` (dostęp: 27 września 2022)

wyniku tych miar. Tym niemniej, duża liczba fałszywych WDO obniża ich wartości pośrednio – obniżając cząstkową precyzję, która jest dodawana do sumy przy kolejnym progu czułości.

Dla wykonania modeli z progiem $T_c = 0.5$ również wyznaczono miary z rodziny AP, które umieszczono w Tabeli 6.7.

Tabela 6.7: **Bazowe wyniki miar AP dla obrazów oryginalnych,** $T_c = 0.5$**.**

Model	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s
R101	33.6	47.0	37.2	46.3	37.5	15.3
R101_C4	<u>38.5</u>	56.3	41.9	**53.6**	42.8	19.1
R101_DC5	38.3	<u>56.8</u>	42.0	52.1	<u>42.8</u>	19.4
R101_FPN	38.4	55.5	<u>42.6</u>	51.2	42.1	<u>20.8</u>
R50	31.6	44.3	35.2	44.3	34.8	14.1
R50_C4	35.9	53.6	39.3	51.0	39.8	17.6
R50_DC5	36.8	55.7	40.5	50.5	41.4	18.2
R50_FPN	36.7	54.1	40.7	49.4	40.1	19.1
X101	**39.6**	**57.0**	**43.9**	<u>52.1</u>	**42.9**	**22.6**

Najlepszy wynik w kolumnie został **pogrubiony**, drugi najlepszy – <u>podkreślony</u>.
AP_l, AP_m, AP_s – miara AP dla obiektów dużych, średnich i małych.

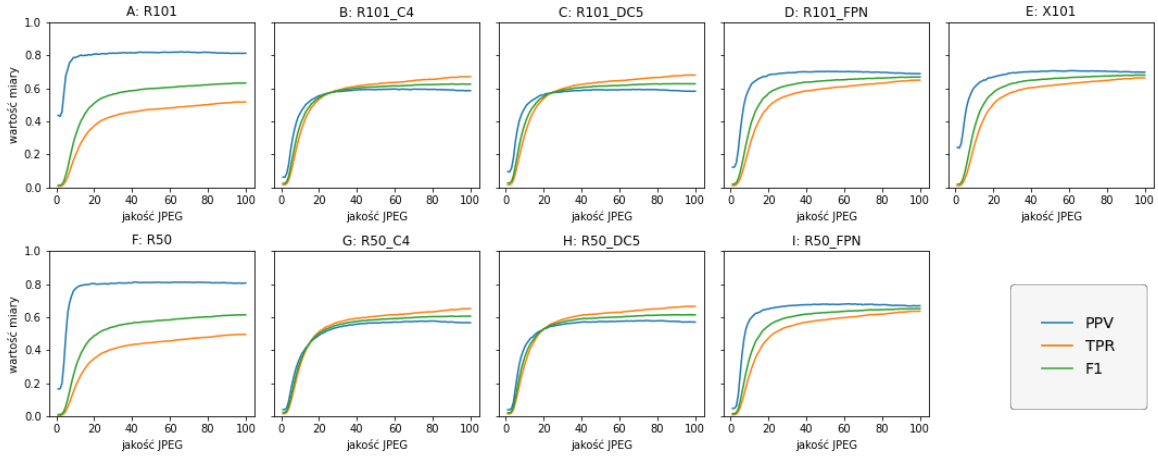
Obliczenie miary AP z takim progiem ufności nie jest w pełni zgodne z ideą tej miary, ponieważ wszystkie TP o ufności mniejszej zostają „odcięte". Porównując Tabele 6.6 i 6.7 można zauważyć, że przez to pominięcie części WDO najwięcej tracą modele R50 i R101 (około 7 punktów procentowych AP), następnie modele z FPN (ok. 3.5 p.p.), a najmniej modele typu C4 i DC5 (ok. 2.5 p.p.). Daje to pewien obraz proporcji liczby poprawnych WDO, które te modele zwracają z ufnością mniejszą od 0.5.

6.4 Skuteczność detekcji w funkcji parametru Q

Po wyznaczeniu i zgromadzeniu miar skuteczności detekcji dla wszystkich wartości parametru Q można przeanalizować zmiany każdej z nich jako funkcję wartości tego parametru. Takie podejście pozwala na wizualną ocenę zachowania tej funkcji na wykresach, a także np. na obliczanie jej dyskretnej pochodnej, rozumianej jako różnica bezpośrednio po sobie następujących elementów ciągu.

Precyzja, czułość i F1. Wykresy miar zależnych od wyboru progu ufności, wyznaczone dla $T_c = 0.5$ zostały przedstawione na Rys. 6.7.

Dla wszystkich modeli reprezentowanych na Rys. 6.7 zauważalna jest przede wszystkim niemal stała wartość precyzji (PPV) w szerokim zakresie Q. W konsekwencji okazuje się, że za spadek miary F1, którą w porównaniu do PPV i TPR można określić jako „ogólniejszą", odpowiedzialny jest spadek czułości detekcji (TPR).



Rys. 6.7: **Wartości TPR, PPV i F1 w funkcji parametru Q.**
Miary zostały wyznaczone dla $T_c = 0.5$. Głębokości kręgosłupa: 101 (górny wiersz), 50 (dolny wiersz). (A), (F) – RetinaNet. (B), (C), (G), (H) – Faster R-CNN bez FPN. (D), (I), (E) – Faster R-CNN z FPN.

Kształt krzywych wszystkich trzech miar zależy od rodziny modeli – modele z tej samej grupy wykazują duże podobieństwa, ale pomiędzy grupami można zauważyć znaczące różnice (relacje między TPR i PPV, dynamika spadku miar itp.).

Dwuetapowe detektory bez FPN (R50_C4, R101_C4, R50_DC5 i R101_DC5) mają następującą charakterystykę:

- mają najbardziej zbliżone wartości TPR i PPV w całym przedziale wartości Q (ponieważ F1 jest ich średnią harmoniczną, to dla tych modeli wszystkie trzy miary mają zbliżony przebieg),
- czułość (TPR) jest bardzo wysoka i dla Q = 100 przyjmuje wartość większą niż PPV; łagodnie opada do wartości równej PPV w pobliżu Q = 20,
- poniżej Q = 20 występuje gwałtowny spadek obydwu miar, najpierw TPR, a następnie PPV (szczególnie dla Q < 16),
- precyzja (PPV) jest bardzo stabilna, ale najniższa wśród badanych rodzin modeli – początkowe wartości to 58% w modelach z siecią ResNet-101 i 57% z modelem ResNet-50, które przy Q = 25 spadają do 57% w ResNet-101 (zaledwie 1 punkt procentowy spadku PPV) i 52–54% dla ResNet-50 (3–5 punktów procentowych spadku).

Detektory jednoetapowe RetinaNet (R50 i R101, obydwa zawierają FPN) mają następującą charakterystykę:

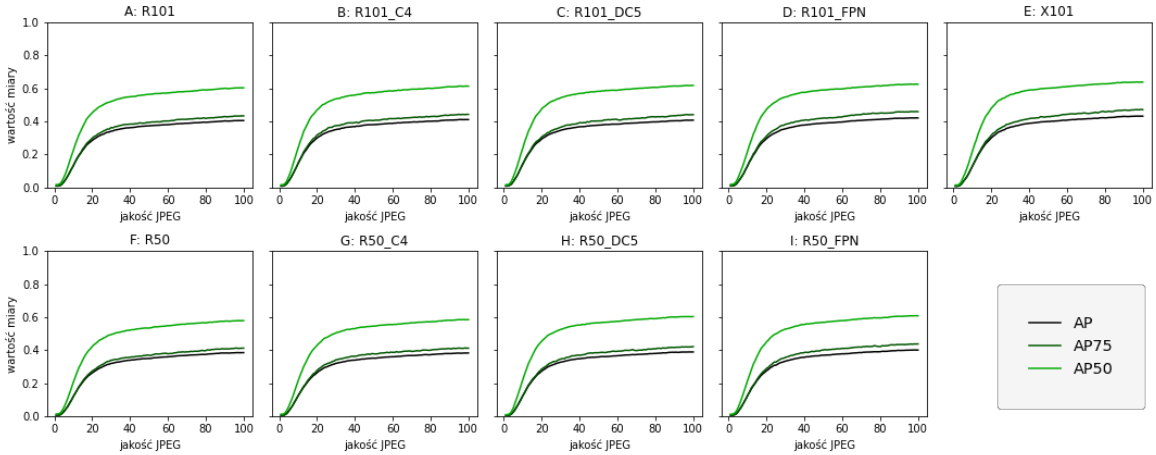
- mają najwyższą wśród wszystkich rodzin modeli precyzję (PPV $\approx$ 80% w przedziale $30 \leqslant Q \leqslant 100$,

- czułość (TPR) jest najniższa wśród badanych modeli i szybko spada wraz ze spadkiem Q.

Detektory dwuetapowe Faster R-CNN z FPN (R50_FPN, R101_FPN, X101) mają następującą charakterystykę:

- stanowią połączenie zalet pozostałych dwóch grup,
- precyzję (PPV) mają wyższą niż czułość (TPR),
- PPV rośnie w sposób nieznaczny wraz ze spadkiem wartości parametru Q,
- wartości TPR i PPV są zbliżone, ale nie tak bardzo jak w pozostałych detektorach dwuetapowych.

Miary średniej precyzji. Wykres miar średniej precyzji (obliczonych dla standardowego $T_c = 0.05$) w funkcji parametru jakości Q przedstawiono na Rys. 6.8.



Rys. 6.8: **Wartości AP, AP_{50} i AP_{75} w funkcji parametru Q.**
Głębokości kręgosłupa: 101 (górny wiersz), 50 (dolny wiersz). (A), (F) – RetinaNet. (B), (C), (G), (H) – Faster R-CNN bez FPN. (D), (I), (E) – Faster R-CNN z FPN.

Kształt krzywych AP dla praktycznie wszystkich modeli wygląda identycznie. Jest to w przybliżeniu monotoniczna funkcja – występują miejscowe zakłócenia, spadki wartości w granicach 0.1–0.12 punktu procentowego AP przy wzroście Q o jeden, przy czym dla miary AP zdarzają się one rzadziej niż dla składowych AP_{50} i AP_{75}.

W kolejnych punktach sformułowano szereg szczegółowych obserwacji:

- W przedziale 95–100 wartości parametru Q (zależnie od modelu, lewy brzeg tego przedziału może być mniejszy lub większy o 1) występują praktycznie stałe wartości miar AP (brak spadku). Wartości miar w granicach 0.2 p.p. od maksimum każdej z miar są osiągane przez modele dla wartości Q począwszy od 88.

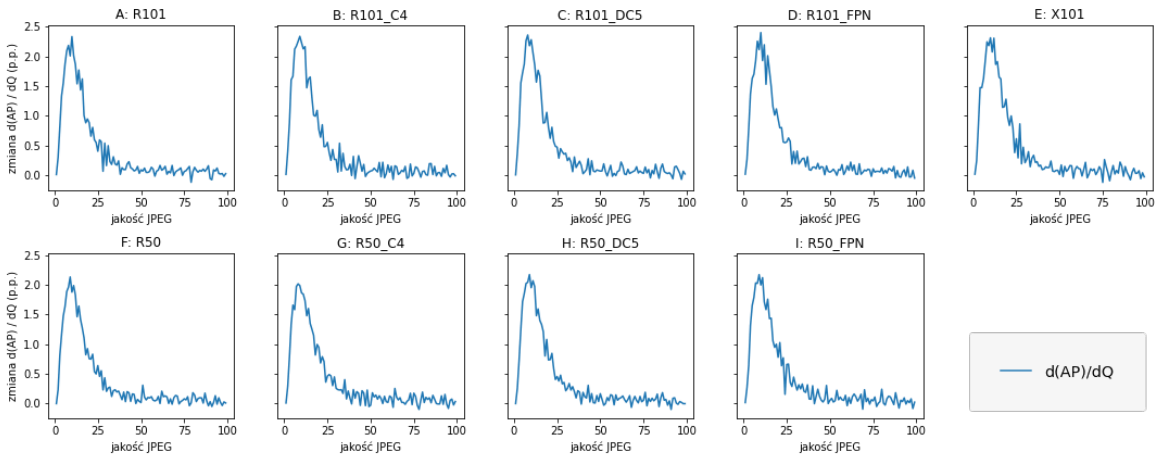
- W przedziale niższych wartości Q (25–87) następuje równomierny spadek wartości miar w sposób łagodny, poniżej 0.9 p.p. w każdym kroku (jest to odcinek wykresu o „łagodnym" zboczu).

- W kolejnym przedziale wartości Q (5–24) następuje szybki spadek wartości miary AP, w przedziale 5–17 wszystkie modele tracą powyżej 1 p.p. na jednostkę Q, a w przedziale 7–15 zmiana wartości miary przekracza 2 p.p. dla niektórych modeli (jest to odcinek wykresu o „stromym" zboczu).

- Najniższe kilka wartości Q (1–4) stanowią „wypłaszczenie" na wykresie, a wszystkie miary średniej precyzji mają w tym przedziale wartość zbliżoną do zera.

- AP $\approx$ AP_{75} – miara AP i jej składowa AP_{75} są bardzo zbliżone dla wszystkich modeli i dla różnych wartości Q, różnica mieści się w granicach 3 punktów procentowych.

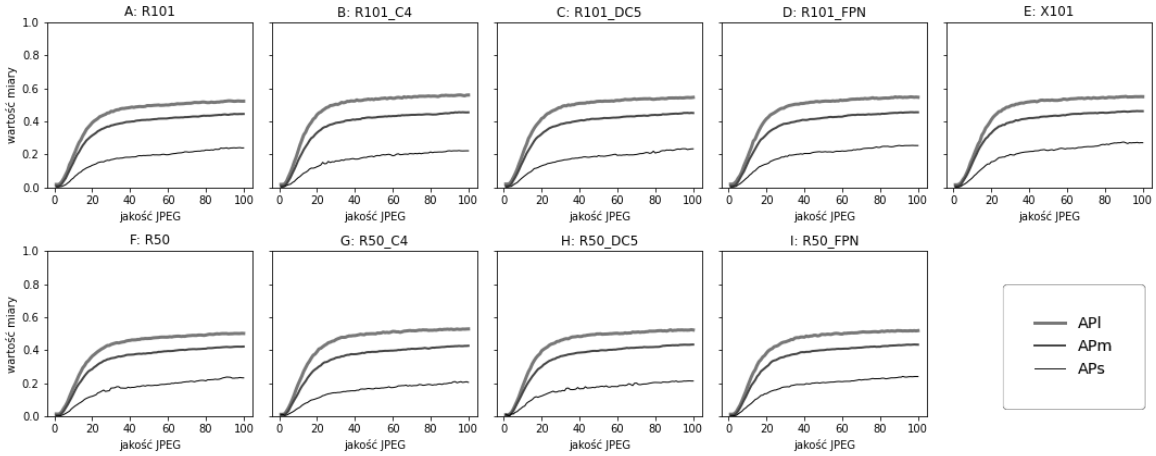
Dokładniejszy obraz zmienności miary AP daje wykres jej pochodnej dyskretnej, tj. różnicy sąsiednich elementów ciągu wartości miary, który jest przedstawiony na Rys. 6.9.



Rys. 6.9: **Dyskretna pochodna średniej precyzji względem parametru Q.** Różnica wartości AP dla sąsiednich Q pokazuje, ile punktów procentowych AP jest tracone przy przejściu na wartość Q mniejszą o 1. Głębokości kręgosłupa: 101 (górny wiersz), 50 (dolny wiersz). (A), (F) – RetinaNet. (B), (C), (G), (H) – Faster R-CNN bez FPN. (D), (I), (E) – Faster R-CNN z FPN.

Obraz różnic wartości miary AP dla sąsiednich wartości parametru Q przede wszystkim ujawnia nieregularne zmiany wartości skuteczności detekcji przy, ale w określonych granicach – skuteczność detekcji spada o nieco inną wartość przy każdej zmianie parametru kompresji. Dla wszystkich modeli widoczne jest gwałtowne przyspieszenie tych zmian poniżej wartości Q = 25, a spowolnienie tempa spadku poniżej wartości Q ok. 10–11 wynika z faktu, że w tym przedziale (1–10) wartość miary AP jest już bardzo bliska zeru (a zatem już tylko niewielka wartość miary przy takiej kompresji „pozostała do stracenia").

Dodatkowe informacje, w stosunku do miar średniej precyzji dla wszystkich ODW, niosą ze sobą miary AP_l, AP_m i AP_s. Wykres zmiany miar średniej precyzji dla ODW podzielonych na grupy według rozmiaru przedstawiono na Rys. 6.10.

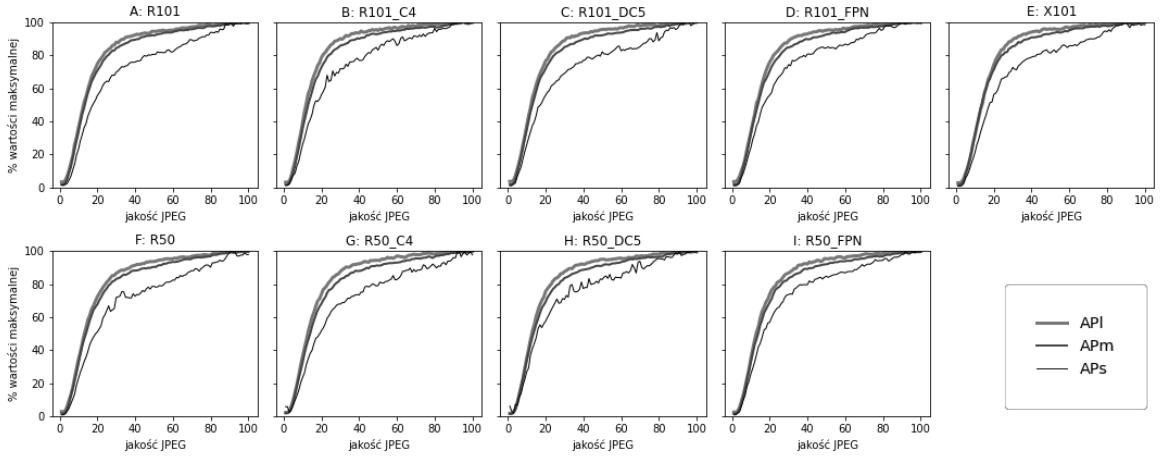


Rys. 6.10: **Wartości AP_l, AP_m i AP_s w funkcji parametru Q.**
Średnia precyzja dla ODW o powierzchni: AP_s – do 1024 px, AP_m – do 9216 px, AP_l – pozostałych. Większa grubość linii symbolicznie wskazuje większy rozmiar obiektów. Głębokości kręgosłupa: 101 (górny wiersz), 50 (dolny wiersz). (A), (F) – RetinaNet. (B), (C), (G), (H) – Faster R-CNN bez FPN. (D), (I), (E) – Faster R-CNN z FPN.

Na wykresie miar średniej precyzji powiązanych z rozmiarem obiektów widać wyraźną tendencję – obiekty duże i średnie są wykrywane dużo lepiej od obiektów małych (poniżej 1024 pikseli powierzchni). W szerokim zakresie („łagodny" odcinek wykresu, Q od około 25 do 88) wartości AP_l, AP_m i AP_s utrzymują w przybliżeniu stałe odległości: AP_l i AP_m dzieli około 7–9 punktów procentowych, a odstęp między AP_m i AP_s jest większy i wynosi około 20 p.p. Poniżej wartości parametru Q = 25 wszystkie trzy miary zmierzają do wartości zero. Linia miary AP_s jest, szczególnie dla modeli R101_C4, R50 i R50_DC5, bardziej „postrzępiona" – lokalnie nie jest monotoniczna. Przewaga modeli dwuetapowych z końcówką DC5 i C4 dla obiektów dużych nie jest w tej skali widoczna, ponieważ w rzeczywistości jest bardzo niewielka.

Jeszcze lepiej można prześledzić zachowanie wartości miar średniej precyzji powiązanych z rozmiarem obiektów, kiedy podzieli się każdy element ciągu przez jego maksymalną wartość w całym przedziale (zazwyczaj jest to wartość dla Q = 100). Pozwala

to wizualnie stwierdzić, czy zmiany każdej z miar są proporcjonalne. Wykresy takich „znormalizowanych" miar przedstawiono na Rys. 6.11.



Rys. 6.11: **Względne wartości AP_l, AP_m i AP_s w funkcji parametru Q.** Na wykresie jest średnia precyzja w podzbiorach obiektów dużych, średnich i małych, podzielona przez maksymalną wartość osiąganą przez każdą z miar z osobna. Większa grubość linii symbolicznie wskazuje większy rozmiar obiektów. Głębokości kręgosłupa: 101 (górny wiersz), 50 (dolny wiersz). (A), (F) – RetinaNet. (B), (C), (G), (H) – Faster R-CNN bez FPN. (D), (I), (E) – Faster R-CNN z FPN.

Na wykresie z Rys. 6.11 widać, że miary AP_l i AP_m są bardzo zbliżone i w funkcji parametru Q ich linie prawie się pokrywają. Zgodnie z intuicją, nieco bardziej pogarsza się wynik dla obiektów średnich, ale w przypadku niektórych modeli (R101, R101_C4, R101_FPN i X101) względne wartości AP_l i AP_m pokrywają się począwszy od wartości Q w przybliżeniu równej 60. Pozostałe modele mają zauważalną w tej skali różnicę już przy wyższych wartościach Q, ale odległość zawsze pozostaje mała, szczególnie dla modeli X101 i obu sieci RetinaNet, R50 i R101.

Inna sytuacja zachodzi dla miary AP_s. W wartościach względnych jeszcze mocniej widoczny jest niejednostajny przebieg wartości miary, przy czym efekt ten zależy od grupy modeli. Najbardziej „gładkie" są przebiegi dla modeli Faster R-CNN z FPN (zwłaszcza R101_FPN). Dla sieci RetinaNet zmienność miary AP_s jest umiarkowana, ale należy zwrócić uwagę na różnicę w zależności od głębokości kręgosłupa – R101 ma bardziej równomierny przebieg niż R50. Najbardziej „postrzępione" są wykresy dla „klasycznych" modeli Faster R-CNN (tj. bez FPN).

Względne wartości AP_s zaczynają odbiegać od pary AP_l i AP_m w przybliżeniu dla wartości $Q \leqslant 80$ (dla niektórych modeli nawet wyższej).

Przebieg linii AP_s jest inny dla każdego z modeli – lokalne maksima i minima nie występują w tych samych miejscach, co wskazuje na to, że nie wynikają one ze specyfiki kompresji, lecz dla poszczególnych modeli pojawiają się w nieregularny sposób.

Oprócz tego, w przedziale „łagodnego zbocza" (Q o wartościach 25–90), nachylenia, tj. spadki wartości miary przy zmianie wartości parametru Q na mniejszą o 1, dla linii

AP_l i AP_m są większe niż nachylenia linii AP_s. Dla $Q \leqslant 25$, tj. w przedziale „szybkiego spadku", to linia AP_s ma łagodniejsze nachylenie (utrzymując wcześniej zaproponowaną konwencję komentarza, można stwierdzić, że ma ona tam mniej „do stracenia"), a miary dla obiektów większych mają mocniejszy spadki (więcej „do stracenia").

Wynika z tego, że w przedziale „łagodnego spadku" miary AP (widocznym na Rys. 6.8) to zmiana miary AP_s ma dominujący wpływ na spadek ogólnego wyniku miary AP. Natomiast w przedziale niskich jakości, tj. wartości Q w zakresie 1–25, wszystkie miary pogarszają się do zera, spadki są znaczne, a ciąg wartości – monotoniczny (bez maksimów lokalnych).

Komplet wynikowych wartości miar w funkcji parametru Q został zamieszczony w tabelach w Dodatku B.

6.5 Zestaw eksperymentalny w etapie II

W drugim etapie realizacji celów rozprawy eksperymenty polegały na treningu modeli detekcji obiektów. Należało zbadać możliwości poprawy wyników poprzez poddanie silnej kompresji obrazów w zbiorze uczącym. Zestaw badanych modeli w tym etapie eksperymentów został ograniczony ze względu na znaczące wymagania obliczeniowe treningu modeli, wielokrotnie większe niż w etapie I, w którym eksperymenty obejmowały jedynie inferencję.

Badane architektury. Badaniu poddano osiem modeli wytrenowanych przez autora rozprawy. Modele te można podzielić na dwie grupy według struktury sieci neuronowej, z której model korzysta. Dla wspólnych cech (np. głębokość sieci, rodzaj zastosowanego kręgosłupa, budowa głowicy detekcji) grupy modeli przyjęto określenie *architektura*. Różnice w każdej grupie dotyczą wag połączeń w sieci, które są wynikiem procesu treningu. Architektura w połączeniu z wagami stanowi model, analogicznie jak sprzęt w połączeniu z oprogramowaniem stanowi system informatyczny.

Obydwie architektury, które zostały wybrane do zbadania, są oparte na ekstraktorze cech ResNet-50 z FPN. Należą do nich:

- jednoetapowy RetinaNet (oznaczony symbolem R50),
- dwuetapowy Faster R-CNN (oznaczony symbolem F50).

Konfiguracja sprzętowa treningu. Sprzęt użyty do treningu modeli składał się z dwóch stanowisk o różnej wydajności:

- wysoka wydajność (WW): procesor Intel Core i9-7900X o częstotliwości 3.3 GHz, pamięć RAM o rozmiarze 128 GB, dwa akceleratory GPU oparte na układzie NVIDIA GV100 (Tesla V100S oraz Titan V), odpowiednio, 32 GB i 12 GB pamięci, moc obliczeniowa 16.35 TFLOPS każdy,

- niska wydajność (NW): procesor Intel Core i7-6700K o częstotliwości 4 GHz, pamięć RAM o rozmiarze 16 GB, dwa akceleratory GPU oparte na układzie NVIDIA GP104 (GeForce 1070 oraz GeForce 1070Ti), 8 GB pamięci każdy, moc obliczeniowa, odpowiednio, 6.46 TFLOPS i 8.19 TFLOPS.

Trening modeli. Dla obydwu architektur zastosowano każde z pięciu podejść treningu opisanych w Tabeli 6.8.

Tabela 6.8: **Sposoby treningu modeli w etapie II eksperymentów.**

Oznaczenie	Opis sposobu treningu modelu
STD	Standardowy trening, oryginalny zbiór COCO `train2017`. Stanowisko: WW. Inicjalne wagi głowicy: losowe. Inicjalne wagi kręgosłupa: MSRA ResNet-50
Q20	Trening na zbiorze COCO `train2017` w jakości Q = 20. Stanowisko: WW. Inicjalne wagi głowicy: losowe. Inicjalne wagi kręgosłupa: MSRA ResNet-50
T20	Transfer uczenia na zbiorze COCO `train2017` w jakości Q = 20. Stanowisko: NW. Inicjalne wagi kręgosłupa i głowicy: *Detectron2 Model Zoo*. (wagi modelu wytrenowanego na oryginalnym zbiorze COCO `val2017`)
Q40	Analogicznie jak w Q20, ale dla jakości Q = 40.
T40	Analogicznie jak w T20, ale dla jakości Q = 40.

Harmonogramem treningu był harmonogram „x1" z biblioteki Detectron2 obejmujący 90 tys. kroków uczenia. Modele badane w etapie I eksperymentów były pretrenowane w harmonogramie „x3", który zawiera 270 tys. iteracji i ma wydłużony okres zmniejszania współczynnika uczenia w stosunku do harmonogramu „x1". W harmonogramie „x1" współczynnik uczenia jest zmniejszany dziesięciokrotnie po 60 tys. i ponownie po 80 tys. iteracji. Zbiorem treningowym, podobnie jak dla modeli pretrenowanych, był COCO `train2017`.

Dodatkowe modele pre-trenowane. Oprócz wytrenowanych modeli, zbadano również miary skuteczności dla modeli z końcówką „D2", które również pochodzą z repozytorium pre-trenowanych modeli Detectron2, ale F50-D2 jest innym modelem niż R50_FPN z etapu I eksperymentów, a R50-D2 nie jest modelem R50 z etapu I. Architektura obydwu modeli nie ulega zmianie, ale wagi połączeń są inne, ponieważ zostały one wytrenowane w harmonogramie „x1" zamiast „x3". Celem zbadania modeli pretrenowanych było zapewnienie wyników bazowych dla modeli z końcówkami „T20" i

„T40". Dla tych modeli punktem wyjścia przed dostrajaniem był właśnie odpowiedni model pre-trenowany.

Miary skuteczności detekcji Na podstawie wyników etapu I ograniczono liczbę wartości parametru Q, dla których wyznaczono miary skuteczności detekcji. Mimo pewnych wahań, wartości miar w funkcji Q są w przybliżeniu monotoniczne (najbardziej „stabilna" jest wartość miary AP, a najmniej – miary AP_s). Dzięki temu można pominąć niektóre wartości Q stosując równomierne próbkowanie i zakładając, że dla punktów pośrednich wartość Q może być interpolowana liniowo. Do zbadania wybrano 14 równomiernie próbkowanych wartości parametru Q, od 5 do 96 z krokiem 7. Obie skrajne wartości Q są reprezentatywne dla obszarów „wypłaszczenia" na wykresach wartości miar (poniżej 5 nie następuje dalsze pogorszenie, a począwszy od wartości 96 wyniki nie odbiegają od bazowych rezultatów).

Dla miar zależnych od progu czułości, zastosowane wartości obydwu progów wynoszą $T_c = 0.5$ i $T_{IoU} = 0.5$.

6.6 Wyniki dla modeli wytrenowanych

Szczegółowe wartości miar skuteczności detekcji dla modeli wytrenowanych przez autora w etapie II eksperymentów zostały zamieszczone w Dodatku C. W tym podrozdziale skupiono się na trzech miarach skuteczności detekcji: PPV, TPR i F1. Przedstawiono podsumowania miar dla dwóch wartości parametru Q – 96 i 33. Ta druga wartość znajduje się pomiędzy wartościami parametru Q, które zostały zastosowane przy degradacji zbioru uczącego.

W tabelach i na wykresach modele podzielono na dwie grupy:

- STD, Q20 i Q40 – modele trenowane z użyciem stanowiska WW od typowej inicjalizacji wag połączeń,
- D2, T20 i T40 – model pre-trenowany i dwa modele wytrenowane drogą transferu uczenia od inicjalizacji wagami połączeń tego modelu, z użyciem stanowiska NW.

Taki podział symuluje dwa odmienne rzeczywiste scenariusze. Pierwszy to trening od podstaw na bardziej wydajnym sprzęcie, który oferuje możliwości takie jak wprowadzanie i usuwanie wykrywanych klas obiektów. Drugi scenariusz to sytuacja, w której nie jest dostępny wydajny sprzęt, więc do wyboru pozostaje albo skorzystanie z modelu pre-trenowanego, albo dostrajanie go do obrazów skompresowanych poprzez transfer uczenia.

Precyzja i czułość. Wyniki miar zależnych od wyboru progu ufności T_c uzyskane dla wartości parametru Q = 96 zostały przedstawione w Tabeli 6.9.

Dla wysokiej jakości obrazu najwyższą czułość w grupie modeli trenowanych (Q20, Q40, STD) mają sieci Faster R-CNN, w kolejności malejącej jakości obrazu w zbiorze

Tabela 6.9: **Wyniki miar TPR, PPV, F1, TP, FP i EX przy** Q = 96**.**

Model	PPV %	TPR %	F1 %	TP	FP	EX
F50-Q20	62.4	56.3	59.2	20473	12359	**2870**
F50-Q40	63.8	57.5	60.4	20876	11864	2773
F50-STD	65.8	**58.2**	**61.8**	**21155**	11017	2691
R50-Q20	82.1	42.7	56.2	15512	3373	809
R50-Q40	82.8	43.1	56.7	15646	3245	721
R50-STD	**84.4**	42.2	56.3	15327	**2833**	669

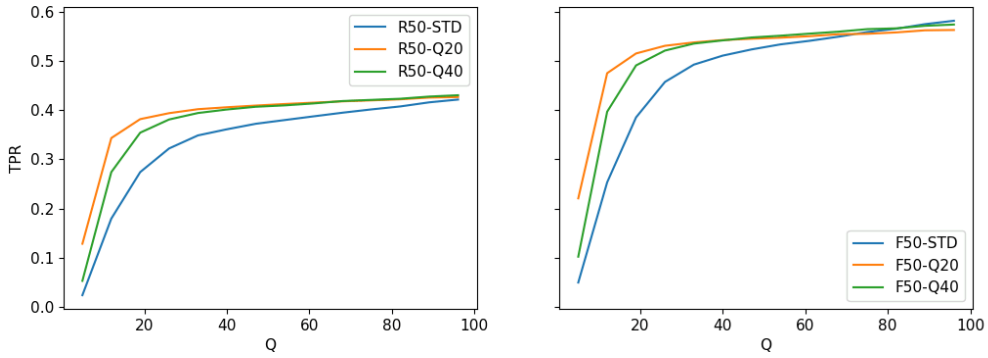
Model	PPV %	TPR %	F1 %	TP	FP	EX
F50-D2	66.6	**61.8**	**64.1**	**22450**	11236	2649
F50-T20	63.3	58.9	61.0	21408	12408	2966
F50-T40	63.3	60.4	61.8	21930	12722	**2975**
R50-D2	83.7	45.8	59.2	16645	3253	720
R50-T20	83.6	43.8	57.4	15898	**3115**	826
R50-T40	**83.7**	44.6	58.2	16213	3156	816

Najlepszy wynik w grupie został **pogrubiony**, drugi najlepszy – podkreślony.

treningowym: F50-STD, F50-Q40, F50-Q20 – różnice pomiędzy kolejnymi modelami wynoszą niewiele ponad 1 punkt procentowy.

Wśród trenowanych modeli RetinaNet nie zachodzi taka kolejność – najwyższą czułość ma model R50-Q40, a modele R50-STD i R50-Q20 mają niemalże równą wartość TPR, niższą od TPR modelu R50-Q40 jedynie o pół p.p.

Miara TPR w funkcji wartości parametru Q dla modeli trenowanych została przedstawiona na wykresie na Rys. 6.12.

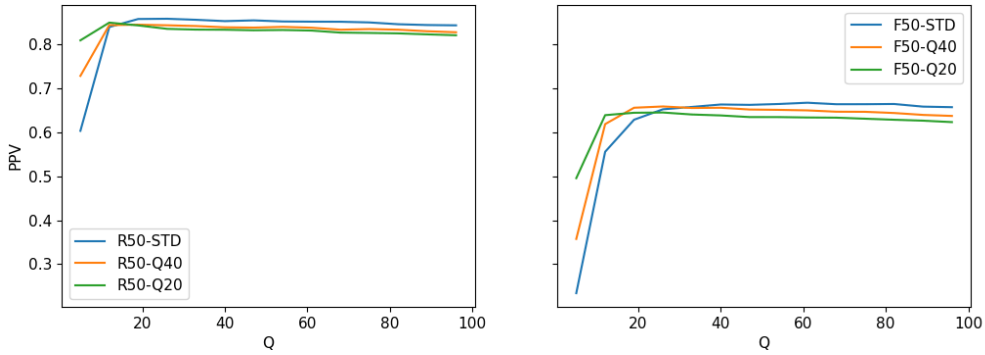


Rys. 6.12: **Wykresy czułości modeli STD, Q20 i Q40 w funkcji parametru Q.** Po lewej modele dwuetapowe Faster R-CNN, po prawej – jednoetapowe RetinaNet.

Dla precyzji kolejność malejącej jakości obrazów treningowych jest zachowana zarówno dla RetinaNet, jak i Faster R-CNN, obydwie architektury mają pogorszenie o 2

p.p. przy przejściu z modelu STD do Q40, różnica między modelami Q40 a Q20 jest mniejsza – dla F50-Q20 wynosi 1.4 p.p., a dla R50-Q20 wynosi 0.7 p.p.

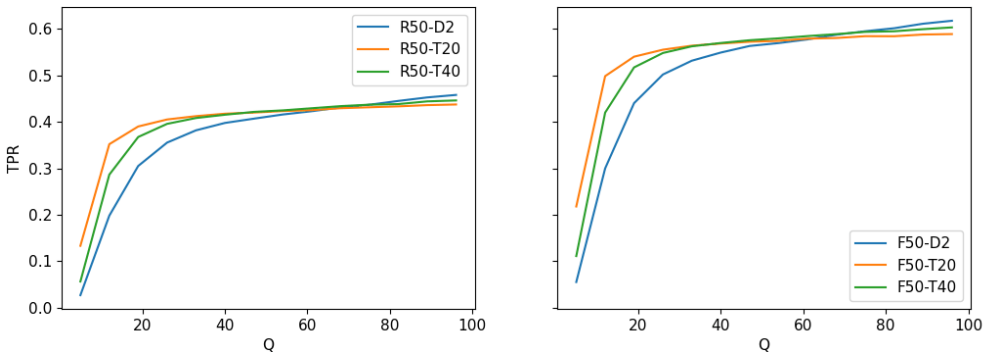
Miara PPV w funkcji wartości parametru Q dla modeli trenowanych została przedstawiona na wykresie na Rys. 6.13.



Rys. 6.13: **Wykresy precyzji modeli STD, Q20 i Q40 w funkcji parametru Q.** Po lewej modele dwuetapowe Faster R-CNN, po prawej – jednoetapowe RetinaNet.

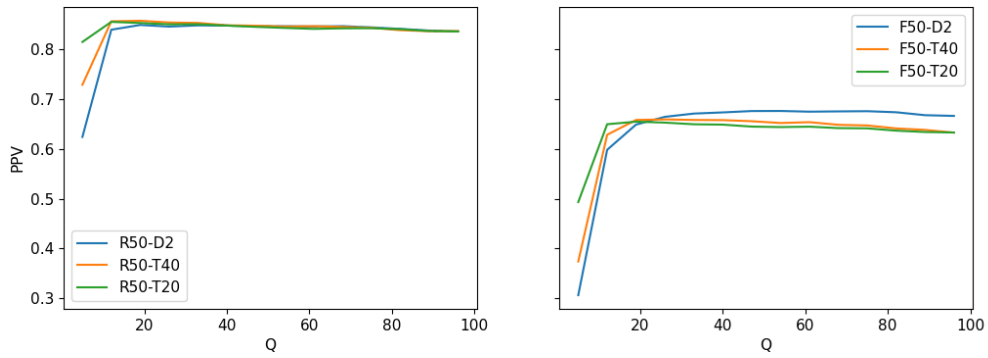
Modele dostrajane również mają małe różnice różnice w czułości dla słabej kompresji przy Q = 96, a kolejność malejącej jakości obrazu w zbiorze treningowym – F50-D2, F50-T40, F50-T20 (Faster R-CNN), podobnie jak R50-D2, R50-T40, R50-T20 (RetinaNet) – również jest zachowana. Różnice pomiędzy kolejnymi modelami wynoszą niewiele ponad (architektura F50) lub dokładnie (architektura R50) 1 punkt procentowy.

Miara TPR w funkcji wartości parametru Q dla modeli dostrajanych została przedstawiona na wykresie na Rys. 6.14.



Rys. 6.14: **Wykresy czułości modeli D2, T40 i T20 w funkcji parametru Q.** Po lewej modele dwuetapowe Faster R-CNN, po prawej – jednoetapowe RetinaNet.

Miara PPV w funkcji wartości parametru Q dla modeli dostrajanych została przedstawiona na wykresie na Rys. 6.15.



Rys. 6.15: **Wykresy precyzji modeli D2, T40 i T20 w funkcji parametru Q.** Po lewej modele dwuetapowe Faster R-CNN, po prawej – jednoetapowe RetinaNet.

Wyniki miar zależnych od wyboru progu ufności T_c uzyskane dla wartości parametru Q = 33 zostały przedstawione w Tabeli 6.10.

Tabela 6.10: **Wyniki miar TPR, PPV, F1, TP, FP i EX przy** Q = 33**.**

Model	PPV %	TPR %	F1 %	TP	FP	EX
F50-Q20	64.1	**53.8**	58.5	**19553**	10956	**2346**
F50-Q40	65.6	53.6	**59.0**	19466	10220	2179
F50-STD	65.8	49.3	56.4	17911	9306	1903
R50-Q20	83.4	40.2	54.3	14614	2903	636
R50-Q40	84.2	39.4	53.7	14331	2681	527
R50-STD	**85.7**	34.9	49.6	12676	**2122**	424

Model	PPV %	TPR %	F1 %	TP	FP	EX
F50-D2	67.1	53.2	59.3	19325	9469	1937
F50-T20	65.0	**56.4**	60.4	**20499**	11056	**2447**
F50-T40	65.8	56.3	**60.7**	20450	10615	2351
R50-D2	84.8	38.2	52.7	13874	**2478**	447
R50-T20	85.1	41.2	55.5	14984	2629	642
R50-T40	**85.3**	40.8	55.2	14838	2549	606

Najlepszy wynik w grupie został **pogrubiony**, drugi najlepszy – podkreślony.

Miary skuteczności detekcji dla silnej kompresji Q = 33 pokazują dominację modeli na zbiorze uczącym o zdegradowanej jakości obrazu. Różnica w przypadku precyzji jest niewielka.

Średnia precyzja. Miary średniej precyzji dla Q = 96 zostały przedstawione w Tabeli 6.11.

Dla modeli trenowanych dominują modele trenowane na standardowym zbiorze, z

Tabela 6.11: **Wyniki miar AP przy** Q = 96**.**

Model	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s
F50-Q20	31.7	51.4	33.7	41.4	33.9	17.8
F50-Q40	33.2	52.9	35.7	43.7	35.5	18.7
F50-STD	**34.3**	**54.6**	**37.3**	**44.1**	37.1	19.2
R50-Q20	32.4	50.3	34.4	42.2	35.8	17.5
R50-Q40	33.6	51.8	36.0	43.0	37.4	18.9
R50-STD	34.3	52.5	36.6	44.1	**38.0**	**19.6**

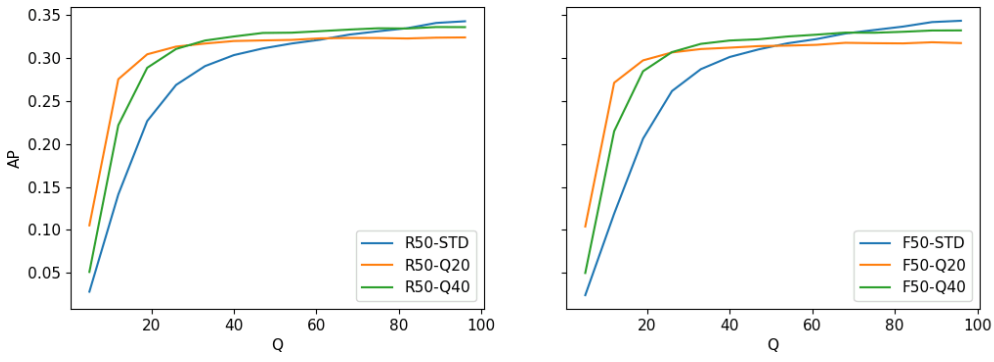
Model	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s
F50-D2	**37.9**	**58.8**	**41.1**	**49.2**	41.1	22.6
F50-T20	34.5	54.5	36.9	45.1	37.3	20.2
F50-T40	35.6	55.8	38.4	46.4	38.3	20.8
R50-D2	37.4	56.7	40.2	48.3	**41.5**	**23.0**
R50-T20	34.6	53.1	37.1	45.0	38.7	19.4
R50-T40	35.5	54.5	37.9	45.5	39.1	21.7

Najlepszy wynik w grupie został **pogrubiony**, drugi najlepszy – podkreślony.

nieznacznymi wyjątkami – model F50-Q40 jest lepszy od R50-STD o 0.4 p.p w mierze AP_{50}, a model R50-Q40 jest lepszy od F50-STD w mierze AP_m o 0.3 p.p. Różnice dla słabej kompresji pomiędzy kolejnymi modelami w rankingu są w granicach 2 p.p.

Podobnie jest dla modeli dostrajanych, za wyjątkiem faktu, że nie występują żadne wyjątki – zawsze na pierwszym i drugim miejscu jest model typu D2 (bazowy).

Miara AP w funkcji wartości parametru Q dla modeli trenowanych została przedstawiona na wykresie na Rys. 6.16.

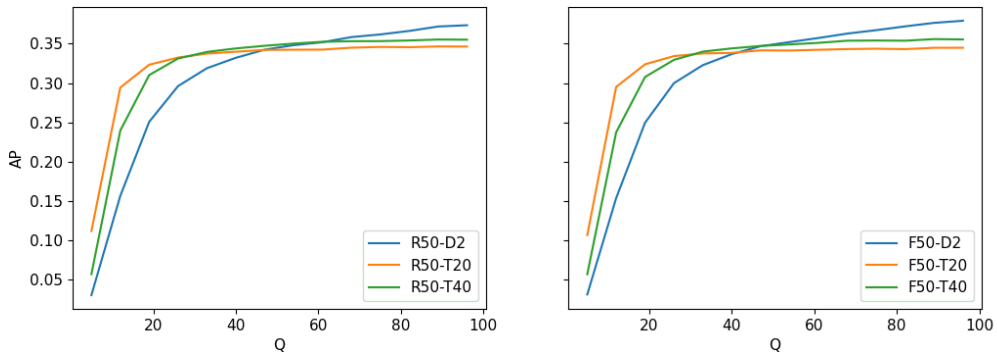


Rys. 6.16: **Wykresy miary AP modeli STD, Q20 i Q40 w funkcji parametru Q.**
Po lewej modele jednoetapowe RetinaNet, po prawej – dwuetapowe Faster R-CNN.

Wykresy miary AP dla trenowanych modeli potwierdzają Tezę 2 – dla modeli jed-

noetapowych i dwuetapowych nie można zaobserwować istotnych różnic. Modele trenowane na obrazach poddanych silnej kompresji Q = 20 mają dla Q ∈ [30, 96] praktycznie poziomy przebieg miary AP w funkcji Q.

Miara AP w funkcji wartości parametru Q dla modeli dostrajanych została przedstawiona na wykresie na Rys. 6.17.



Rys. 6.17: **Wykresy miary AP modeli D2, T40 i T20 w funkcji parametru Q.** Po lewej modele jednoetapowe RetinaNet, po prawej – dwuetapowe Faster R-CNN.

Miary średniej precyzji dla Q = 33 zostały przedstawione w Tabeli 6.12.

Tabela 6.12: **Wyniki miar AP przy** Q = 33**.**

Model	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s
F50-Q20	31.0	50.3	33.0	41.3	33.0	16.2
F50-Q40	31.7	**50.9**	33.5	**43.1**	33.6	16.4
F50-STD	28.7	46.8	30.3	38.8	31.1	14.3
R50-Q20	31.7	49.2	33.5	42.6	34.8	17.7
R50-Q40	**32.0**	49.8	**34.0**	42.5	**35.5**	**18.0**
R50-STD	29.1	45.7	30.5	39.0	32.1	15.7
Model	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s
F50-D2	32.3	51.8	34.7	43.9	35.3	16.4
F50-T20	33.8	53.5	36.1	45.1	36.7	18.1
F50-T40	**34.0**	**53.7**	**36.5**	**45.3**	36.7	18.5
R50-D2	31.9	49.9	33.5	43.1	35.4	16.3
R50-T20	33.7	52.2	36.0	44.4	**37.5**	18.8
R50-T40	34.0	52.5	36.0	44.9	37.2	**19.7**

Najlepszy wynik w grupie został **pogrubiony**, drugi najlepszy – podkreślony.

W tabeli dla Q = 33, podobnie jak dla miar TPR i PPV, dominują modele trenowane na obrazach poddanych średniej i silnej kompresji.

Wyniki uzyskane w etapie I posłużą do uzasadnienia Tez 1 i 2 rozprawy, natomiast rezultaty etapu II zostaną wykorzystane dla potwierdzenia Tezy 3.

Rozdział 7

Omówienie rezultatów i wnioski

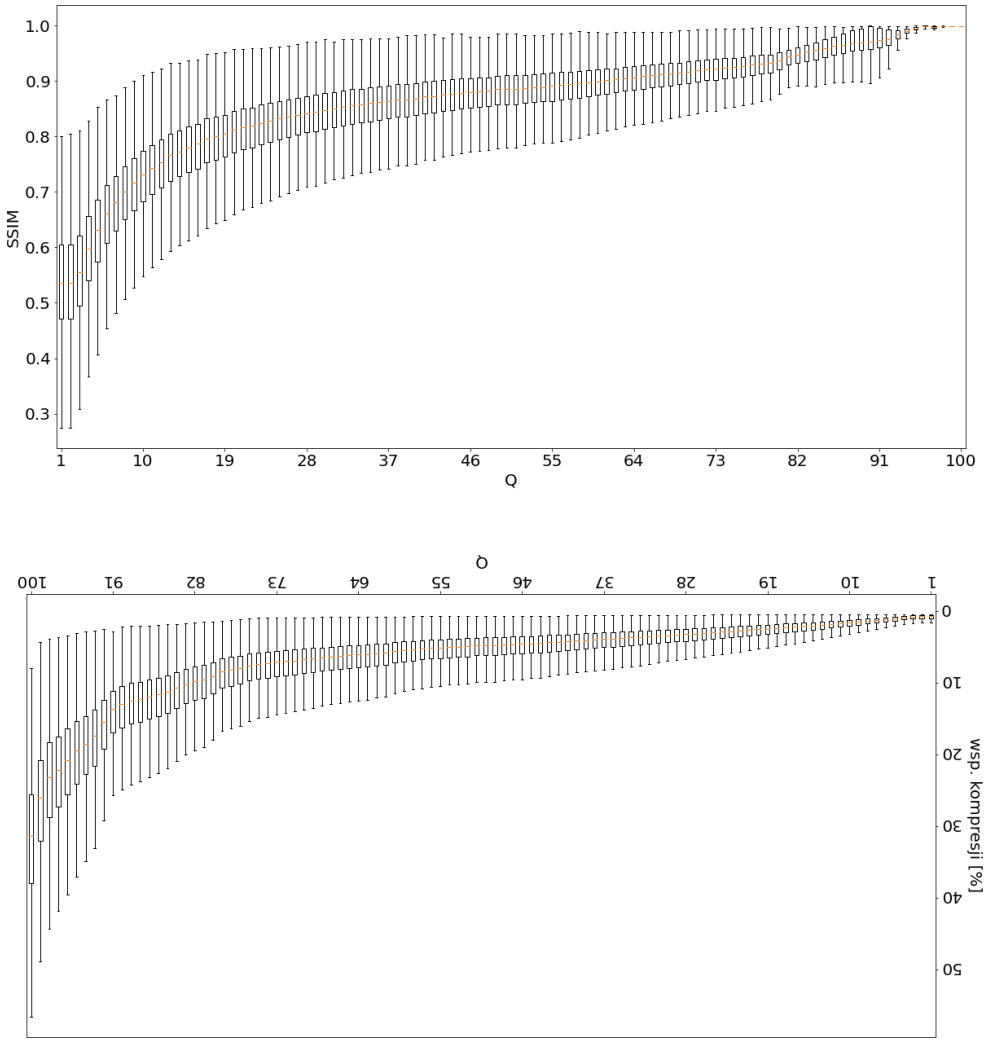
Niniejszy rozdział zawiera dyskusję i wnioski sformułowane na podstawie wyników zaprezentowanych w rozdziale 6. Część z nich ma charakter praktycznych zaleceń, które mogą być pomocne w inżynierii systemów monitoringu wideo opartych na modelach detekcji obiektów, w których obrazy są poddawane stratnej kompresji.

Oprócz tego, w tym rozdziale znajdują się odniesienia do literatury. Inaczej niż dla przeglądu prac w rozdziale 2, nie są to uprzednie prace na podobny temat, ale publikacje, które pozwolą poszerzyć zakres badań zapoczątkowanych w niniejszej pracy. Niektóre z przytoczonych prac zawierają odkrycia i obserwacje, które potwierdzają lub są zbieżne z tymi, które sformułowano na podstawie badań wykonanych w niniejszej pracy.

7.1 Wnioski i wskazania praktyczne

Pierwsza obserwacja dotyczy działania kompresji JPEG i wynikowej jakości obrazu mierzonej za pomocą podobieństwa strukturalnego (SSIM). Żeby ją poczynić, należy zestawić ze sobą wykresy z Rys. 6.1 i Rys. 6.3. Można dostrzec, że oba wykresy są bardzo podobne do siebie, o ile jeden z nich obrócimy o 180°. Zostało to przedstawione na Rys. 7.1.

Patrząc w kierunku malejących wartości Q, oznacza to, że najpierw gwałtownie spada rozmiar danych wyjściowych (rośnie stopień kompresji) przy prawie stałej jakości, następnie obydwie wartości spadają łagodnie przez większość przedziału wartości parametru Q, a następnie, przy najniższych wartościach Q, gwałtownie spada jakość, przy prawie stałym rozmiarze (braku poprawy stopnia kompresji). Wniosek praktyczny jest taki, że stosowanie skrajnych wartości parametru Q nie ma sensu – za wysokie wartości to duży rozmiar plików wyjściowych, a za niskie to katastrofalny spadek jakości, w obu przypadkach z nikłymi korzyściami. Wybór wartości w środkowym, szerokim przedziale (ok. 20–90), może być podyktowany wymaganiami konkretnego zastosowania. Na przykładzie zbioru COCO, który używa głównie wartości Q = 96, widać, że praktyka w budowaniu zbiorów OiM do detekcji obiektów była zgodna z tą obserwacją, a uzyskane tu wyniki dodatkowo potwierdzają jej słuszność.



Rys. 7.1: **Podobieństwo wykresów pudełkowych SSIM i współczynnika kompresji.**
Rysunek przedstawia obrócony wykres współczynnika kompresji (Rys. 6.3) pod wykresem SSIM (Rys. 6.1).

Wyniki uzyskane w etapie I eksperymentów pozwalają sformułować następujące wskazania praktyczne dla projektantów i operatorów systemów monitoringu stosujących detekcję obiektów z użyciem sieci głębokich:

- Bezwzględnie nie należy stosować parametru Q o wartościach poniżej 20, a niewskazanie jest użycie Q niższego niż 35. Dopuszczalna wartość zależy od rodzaju obiektu (większe obiekty, np. samochody, mogą być skutecznie wykrywane przy niższym Q).

- Jeżeli celem detekcji jest osiągnięcie wysokiej precyzji, ale dopuszczalne są niewykryte obiekty (FN), to można wybrać dowolną wartość Q z przedziału 35–100. Wprawdzie, w przypadku niektórych modeli, zaobserwowano poprawę precyzji przy wartościach parametru Q w przedziale 60–80 w stosunku do przedziału 80–100, ale była ona bardzo niewielka i wiązała się ze spadkiem czułości. Do praktycznych rozważań można więc przyjąć, że wartość precyzji będzie stała. Z drugiej strony, zmiana wartości parametru Q np. z 90 na 100 nie przyniesie korzyści w postaci zwiększenia precyzji, a będzie się wiązała z podwyższonymi kosztami (rozmiar plików).

- Jeżeli celem detekcji jest osiągnięcie wysokiej czułości, to wymagania względem jakości obrazu są wyższe. Zalecane wartości parametru Q są w przedziale 70–94.

Na podstawie wykresów z Rys. 6.8 można poczynić drobne spostrzeżenie o charakterze technicznym. Zwraca uwagę fakt, że dla wszystkich badanych modeli AP $\approx$ AP_{75}. Miara AP jest średnią arytmetyczną dziesięciu miar, od AP_{50} do AP_{95}, w miara AP_{75} prawie na środkową z nich. Wyznaczenie jest wartości jest dziesięciokrotnie szybsze niż AP, dlatego w pewnych sytuacjach (walidacja dużej liczby modeli, zgrubna ocena skuteczności detekcji) warto ją stosować. Kiedy zajdzie potrzeba porównania opracowywanego modelu z wynikami opublikowanymi w literaturze, można wyznaczyć pełną miarę AP.

Wyniki etapu II znacznie poszerzają informację na temat badanych architektur w stosunku do wyników etapu I. Przede wszystkim zbadano już nie pojedyncze modele, odnajdując podobieństwa ich charakterystyki na podstawie przynależności do określonych grup, ale przeprowadzono eksperymenty na dwóch architekturach, obserwując efekty poddania ich zróżnicowanym procesom treningu.

Dzięki temu można było zaobserwować, że miara PPV nie tylko jest w dużym stopniu odporna na stratną kompresję obrazów wejściowych. Zaskakujące jest to, że jest ona odporna również na kompresję obrazów treningowych. Na wykresach z Rys. 6.13 widać obie te cechy – poziome ułożenie linii pokazuje znikomy wpływ kompresji obrazów wejściowych, a bliskość linii (za wyjątkiem skranie niskich wartości Q) pokazuje mały wpływ kompresji zastosowanej na zbiorze treningowym. Szczególnie jaskrawo widoczne jest to na Rys. 6.15, tj. dla modeli dostrajanych. Uprzednio wytrenowana precyzja dla architektury RetinaNet jest całkowicie odporna na transfer uczenia na silnie skompresowanych obrazach – wszystkie trzy linie niemal się nakładają. Stanowi to uzupełnienie Tezy 1 niniejszej rozprawy.

Wpływ kompresji obrazu na TPR w modelach trenowanych w sposób standardowy (w tym modele pre-trenowane) jest odmienny od wpływu jaki kompresja obrazu ma na TPR modeli trenowanych lub dostrajanych z użyciem obrazów silnie skompresowanych. Dla tych ostatnich, powyżej pewnej wartości parametru Q występuje pozioma linia (podobnie jak dla miary AP w przypadku tych modeli). W modelach pre-trenowanych i trenowanych standardowo występował w przedziale $Q \in [30, 96]$ jednostajny spadek czułości, który rzutował na wynik miary AP. Modele trenowane na obrazach o średniej kompresji wykazują podobieństwo do obu pozostałych grup.

7.2 Ulepszenia treningu detektorów

Problemy wiążące się z treningiem modeli detekcji zostały omówione w rozdziale 4. W tym podrozdziale przytoczono informacje z publikacji, które należy uwzględnić w dalszych badaniach nad treningiem detekcji obiektów, skutecznej w warunkach kompresji stratnej.

Oksuz [66] omawia problemy niezrównoważenia obiekt-tło (rzadkie występowanie obiektów) i obiekt-obiekt (liczby wystąpień klas w zbiorze uczącym znacząco odbiegające od siebie), opisane w rozdziale 4. Oprócz tego rozważa problem braku zrównoważenia w odniesieniu do następujących właściwości:

- zbiór uczący: skala (rozmiary) obiektów, położenie w różnym obszarze obrazu, nakładanie się prostokątów obiektów;
- kręgosłup: głębokość (poziom abstrakcji) warstw sieci, z których cechy przekazywane są do głowicy detekcji;
- proces trenowania: relacje między składnikami funkcji straty – strata klasyfikacji i strata lokalizacji.

Szczególną wartością tej pracy jest dołączone do niej publiczne repozytorium GitHub[1], które jest aktualizowane na bieżąco. Aktualizacje dotyczą podejść rozwiązujących poszczególne problemy, a oprócz nazwy każdego podejścia jest, tam, gdzie to możliwe, odnośnik do związanej z nim publikacji.

Jedną z prac, która dotyczy przeciwdziałania skutkom braku zrównoważenia obiekt-obiekt oraz zapewnienia zrównoważenia składników funkcji straty, jest praca Chena i in. [11]. Oprócz wprowadzenia klasyfikacji z użyciem SVM pomiędzy etapami dwuetapowej detekcji obiektów, autorzy proponują własną funkcję straty, która ma możliwość wzmocnienia składnika klasyfikacji względem składnika lokalizacji.

Ciekawy pomysł został przedstawiony w pracy [37]: trening całościowy (ang. *end-to-end*) modelu detekcji obiektów z użyciem funkcji straty opartej o miarę mAP (miara średniej precyzji, z ustalonym progiem T_{IoU} nakładania się prostokątów ODW i WDO). Autorzy wskazują niekonsekwencję polegającą na tym, że w czasie testowania detektorów, do oceny używana jest miara mAP, która operuje na wyniku detekcji poddanym usuwaniu obserwacji nie-maksymalnych, podczas gdy w czasie treningu funkcja straty nie uwzględnia tej operacji. Próba zastosowania mAP z NMS bezpośrednio jako funkcji straty napotka jednak na pewien problem, ponieważ taka funkcja jest obszarami stała (ang. *piecewise constant*). To natomiast jest bardzo niekorzystne z punktu widzenia treningu modelu, który opiera się na gradiencie funkcji straty, a ten w obszarach stałych jest równy zero, a na granicach takich obszarów – nieokreślony.

Aby przezwyciężyć te trudności, autorzy proponują dwie metody obliczania pochodnych cząstkowych funkcji obszarami stałej (próbkowanie otoczenia i średnia obwiedni liniowych). W celu zapewnienia stabilności treningu opartego na mAP wymagane jest

[1] `https://github.com/kemaloksuz/ObjectDetectionImbalance` (dostęp: 27 września 2022)

znaczne zwiększenie rozmiaru wsadu pojedynczej iteracji (ang. *batch size*), ponieważ miara mAP jest liczona na całym zbiorze obiektów, z uwzględnieniem ich porządku względem wartości ufności. Co prawda, uzyskane wyniki nie przewyższają rezultatów standardowego treningu sieci Fast R-CNN, ale nie są również od nich gorsze. Artykuł stanowi zwięzłe źródło informacji o mierze mAP, sieciach Fast R-CNN i metodzie eliminacji obserwacji NMS.

Praca Bergmanna i in. [6] opisuje trening do detekcji defektów (np. uszkodzenia tkanin, paneli fotowoltaicznych i dróg). Problem detekcji defektów może być rozwiązywany z użyciem algorytmów detekcji obiektów, np. w pracy Du [17]. W tym przypadku jednak postąpiono inaczej – zastosowano segmentację obrazu (klasyfikację pikseli), jednak nie wprost – na początku obraz jest przetwarzany przez autoenkoder (model, który dokonuje kompresji, a następnie dekompresji obrazu). Model ten jest trenowany na obrazach poprawnych, pozbawionych defektów. Po podaniu na wejście takiego modelu obrazu z defektem, obszar tego defektu jest odtwarzany mniej dokładnie – w skrajnym przypadku autoenkoder może „naprawić" obraz, usuwając defekt[2]. Zazwyczaj jednak defekt jest widoczny w pewien sposób na obrazie wyjściowym, ale jest on zrekonstruowany mniej dokładnie w stosunku do nieuszkodzonych części obrazu. W tym miejscu następuje użycie referencyjnej miary jakości obrazu – albo globalnie, do klasyfikacji – spadek jakości rekonstrukcji wskazuje, że obraz zawiera defekt – albo lokalnie – wtedy możliwa jest segmentacja obszarów, w których zaobserwowano spadek jakości. W najprostszym przypadku stosowano do tego miarę MSE, która również może być funkcją straty przy treningu autoenkodera. Autorzy pracy [6] dostosowali bardziej skomplikowaną miarę SSIM do użycia jako funkcji straty przy treningu, a następnie przetestowali skuteczność segmentacji defektów z użyciem różnych miar w roli funkcji straty w treningu oraz jako miary jakości podczas wykonania modelu. Wyniki pokazały zdecydowaną przewagę wariantu, w którym indeks podobieństwa strukturalnego jest zastosowany w obydwu etapach.

Trening detektorów obiektów w praktyce jest kosztowny nie tylko ze względu na wymagania obliczeniowe, ale również ze względu na konieczność stworzenia zbioru OiM do uczenia modelu. Najbardziej pracochłonne jest przy tym rysowanie prostokątów ODW, co zazwyczaj wymaga dla każdego obiektu co najmniej dwóch precyzyjnych kliknięć urządzeniem wskazującym, albo jednego przeciągnięcia kursora po przekątnej obiektu, z ewentualnymi poprawkami rozmiaru i lokalizacji. Jest to znacznie bardziej czasochłonne niż klasyfikacja obrazu, ale przygotowanie zbioru OiM dla problemu segmentacji instancji jest oczywiście jeszcze kosztowniejszym przedsięwzięciem. Papadopoulos i in. [68] opisują metodę uproszczenia tej procedury z użyciem techniki uczenia słabo nadzorowanego (ang. *weakly-supervised learning*), która polega na wielokrotnym trenowaniu modelu, równolegle z budowaniem zbioru OiM.

Na początku potrzebna jest niewielka liczba poprawnych adnotacji ODW, które służą do wytrenowania pierwszego cyklu modelu. Model wykonany na poszerzonym zbiorze obrazów zwraca WDO, które są kandydatami na ODW. W tym miejscu sto-

[2]To pozwala na uboczne zastosowanie takiego modelu np. do poprawy jakości fotografii poprzez usuwanie artefaktów wizualnych (ang. *inpainting*).

sowana jest weryfikacja przez człowieka, ale zostaje ona sprowadzona do problemu klasyfikacji binarnej – odpowiedzi, czy kandydat ODW jest poprawny (wówczas staje się częścią zbioru OiM), czy nie (wówczas również jest zapamiętywany jako przykład negatywny, zawężający przestrzeń poszukiwań nowych ODW). Autorzy rozszerzają jednak klasyfikację prostokąta kandydata ODW na kilka innych przypadków: *container* – wszystkie ODW na obrazie mają być zawarte w tym prostokącie, *part* – zaznaczona jest tylko część prawidłowego ODW, *mixed* – usunąć wszystkie propozycje ODW, które nie zawierają tego prostokąta lub nie są przez niego zawarte (tj. zostawić te, których granice nie przecinają się z nim) oraz *missed* – wszystkie propozycje ODW z niezerowym IoU względem danego prostokąta mają zostać usunięte.

W artykule przedstawiono wyniki doświadczeń, m.in. na zbiorze PASCAL VOC, w których porównano efekt końcowy przygotowania zbioru OiM przy ręcznym i przy zautomatyzowanym tworzeniu ODW.

Kolejną praca, która jest pomocna przy trenowaniu detekcji obiektów, zawiera opis przygotowania zbioru uczącego i treningu segmentacji instancji opublikowany przez Follmanna i in. [23]. Dotyczy ona wykrywania stosunkowo niewielkich obiektów (artykuły handlowe dostępne w supermarketach). Opisana metoda również nie wymaga zaznaczania ODW w zbiorze, ponieważ korzysta z fizycznych urządzeń wspierających akwizycję obrazu – przemysłowych kamer wizyjnych, oświetlaczy i stolika obrotowego z neutralnym, łatwym w segmentacji (metodami klasycznego widzenia maszynowego) tłem. Dzięki stolikowi możliwe jest uzyskanie zdjęć obiektów każdej klasy w różnych ustawieniach fizycznych i kierunkach oświetlenia. Następnie zbiór OiM jest poddany augmentacji, tj. rozszerzeniu za pomocą metod przetwarzania obrazów – np. obiekty są umieszczane na zaszumionym tle, albo jako bezpośrednio stykające się z sobą (w celu dostarczenia przypadków uczących z naciskiem na separację ODW). Wynikiem tej procedury jest model segmentacji instancji (zarazem detekcji obiektów), który dobrze wykrywa obiekty na obrazie w naturalnym środowisku. Pokazano w ten sposób zalety zastosowania właściwych środków technologicznych już na etapie akwizycji obrazu do stworzenia zbioru uczącego.

Użycie detekcji obiektów w praktycznym zastosowaniu, takim jak monitoring wideo, może wymagać trenowania i „dotrenowywania" (uczenie transferowe) modeli detekcji, czy to z uwagi na wymagania klienta (wykrywanie szczególnych klas osób, np. pracowników sklepu i klientów – osobno), czy też ze względu na uwarunkowania techniczne (np. użycie stratnej kompresji, co pokazują badania w niniejszej pracy). Podsumowując zalecenia praktyczne odnoszące się do treningu modeli, można wymienić następujące punkty:

- Należy zbadać dostępny zbiór OiM pod kątem zbalansowania i rozważyć zastosowanie technik przeciwdziałania niezbalansowaniu zbioru uczącego.
- Jeżeli zbiór uczący jest tworzony od podstaw, opłaca się zastosować uczenie maszynowe już na etapie adnotowania obrazów, z pomocniczym udziałem człowieka (ang. *human in the loop*), zamiast tworzenia wszystkich adnotacji ręcznie.
- Dla nowych klas ODW, których model powinien się nauczyć, możliwa jest au-

gmentacja zbioru uczącego przez sztucznie wygenerowane obrazy, w których umieszczone są obiekty pozyskane z innych ujęć, np. z użyciem tzw. „blue box"-u (ang. *chroma key compositing*).

- Jeżeli celem treningu jest optymalizacja pewnej miary skuteczności, np. średniej precyzji, ale również dowolnej innej, można uwzględnić tę miarę w funkcji straty treningu detektora obiektów (z koniecznymi zmianami, jeżeli miara nie jest „przyjazna gradientom").

7.3 Ocena detekcji obiektów

Miara średniej precyzji stanowi proste kryterium do porównywania między sobą modeli detekcji obiektów, ale nie jest pozbawiona ograniczeń, na które zwracają uwagę autorzy publikacji dotyczących oceny detekcji obiektów. Niektórzy wskazują problemy, np. Redmon [74] pokazuje jaskrawe przykłady dwóch detekcji, które są równoważne w sensie miary AP, ale znacząco różne w praktyce. Inne prace oprócz krytyki samej miary AP proponują rozwiązania i własne miary jakości detekcji obiektów.

Jednym z takich artykułów jest praca Oksuza i in. [67], którzy proponują miarę LRP (ang. *localization-recall-precision error*). Praca zawiera krytykę miary AP, również z przykładem – tym razem trzech – detektorów o tym samym wyniku miary AP, oraz z wykresami ich (arbitralnych, stworzonych na potrzeby przykładu) krzywych precyzja-czułość. Pierwszy detektor ma „pionowy" przebieg krzywej, tzn. ma $\mathrm{PPV} = 1$ dla $\mathrm{TPR} \leqslant 0.5$, a następnie $\mathrm{PPV} = 0$, tzn. wykrywana jest połowa ODW, ze stuprocentową precyzją. Drugi detektor ma poziomy przebieg precyzji na poziomie 0.5, czyli ma stuprocentową czułość – wykrywa wszystkie ODW, ale średnio co drugi WDO jest fałszywy. W trzecim detektorze jest „ukośny" przebieg krzywej precyzja-czułość, to znaczy precyzja od $\mathrm{TPR} = 0$ do $\mathrm{TPR} = 1$ liniowo spada od jedności do zera (ten przykład jest najmniej realistyczny, ponieważ przy niezerowym TPR nie można uzyskać zerowego PPV, co najwyżej asymptotyczne zbliżanie się do zerowej precyzji; z drugiej strony rzeczywiste detektory, takie jak RetinaNet, mają bardzo zbliżone zachowanie do tego przebiegu – obniżanie progu T_c powoduje „eksplozję" FP i tym samym zmniejszenie PPV dla coraz mniejszych przyrostów TPR). Każdy z tych trzech detektorów jest zdecydowanie różny od pozostałych i być może nadaje się do innych zastosowań praktycznych niż pozostałe, ale łączy je jedno – taki sam wynik miary AP.

Pozostałe argumenty przeciw miarom AP to, po pierwsze, ich „niewrażliwość" na próg ufności T_c (co zostało dokładniej opisane w rozdziale 4). Chodzi o to, że ufności WDO służą wyłącznie do ustalenia porządku przetwarzania WDO przy obliczaniu miar AP. Po drugie, nie ma możliwości ustalenia żadnego „optymalnego" progu T_c na podstawie AP. Ostatnim wskazanym problemem jest interpolacja wartości czułości dla klas o małej liczbie ODW (np. dla 9 elementów z próbkowaniem czułości co 1%).

W pracy [67] znajduje się również odniesienie do miary F1 i wskazanie publikacji, które opisują dobór progu ufności poprzez jej maksymalizację. Jednym ze wskazanych ograniczeń miary błędu opartej na F1, $E = 1 - F1$, jest brak spełniania przez nią nie-

równości trójkąta (nie jest to więc metryka, lecz semi-metryka[3]). Ten problem z miarą F1 jest podkreślany m.in. w raporcie Powersa [70], który dotyczy F1 jako narzędzia oceny klasyfikacji – sama wartość F1 może być równa dla klasyfikatorów o różnych właściwościach (krytyka ta jest podobna do krytyki AP przez Oksuza i in.).

Proponowana w pracy [67] miara LRP jest miarą błędu – najlepszą możliwą wartością jest zero, dla doskonałej detekcji, a najgorszą wartością jest jeden. Aby obliczyć LRP, należy przyjąć próg nakładania się prostokątów ODW i WDO T_{IoU}, podobnie jak w przypadku miar AP. Inaczej niż w przypadku AP, należy przyjąć też próg ufności T_c, który pozwala wyznaczyć liczby TP, FP i FN, a tym samym również wartości TPR i PPV. Miara LRP jest ważoną średnią trzech elementów, a wagami są liczby: TP (dla błędu lokalizacji, LRP_{TP}), FN (dla błędu czułości, LRP_{FN}) i FP (dla błędu precyzji, LRP_{FP}).

$$LRP = \frac{TP \cdot LRP_{TP} + FN \cdot LRP_{FN} + FP \cdot LRP_{FP}}{TP + FP + FN}. \tag{7.1}$$

Błąd czułości to:

$$LRP_{FN} = 1 - TPR = \frac{FN}{TP + FN}. \tag{7.2}$$

Błąd precyzji to:

$$LRP_{FP} = 1 - PPV = \frac{FP}{TP + FP}. \tag{7.3}$$

Błąd lokalizacji jest liczony tylko dla poprawnych WDO (tylko dla nich istnieje ODW względem którego można go wyznaczyć). WDO będą miały IoU równe co najmniej T_{IoU}, zatem będą leżały w przedziale $[T_{IoU}, 1]$. Aby znormalizować błąd lokalizacji do przedziału $[0, 1]$, dla LRP_{TP} zastosowano dzielenie przez $1 - T_{IoU}$:

$$LRP_{TP} = \frac{1}{TP} \sum_{i=1}^{TP} \frac{1 - IoU_i}{1 - T_{IoU}}, \tag{7.4}$$

gdzie IoU_i oznacza indeks Jaccarda prostokątów WDO i ODW dla i-tego poprawnego WDO. Liczba $1 - LRP_{TP}$ stanowi średnie IoU poprawnych WDO z ich przypisanymi ODW.

Autorzy przedstawiają zalety tej miary, jak również podejście, dzięki któremu można zastąpić za jej pomocą miarę AP. Polega to na znalezieniu dla każdej klasy indywidualnego progu ufności, który minimalizuje dla niej wartość LRP i ta minimalna wartość dla klasy c jest oznaczana jako $oLRP_c$ (ang. *optimal localization-recall-precision error*). Wartością używaną jako finalna miara oceny jest moLRP (ang. *mean optimal localization-recall-precision error*).

Ubocznym produktem są progi ufności dla poszczególnych klas – które minimalizują LRP. W ten sposób użytkownik detektora otrzymuje wskazówkę jaki próg ufności

[3]tj. funkcja o wartościach nieujemnych, symetryczna względem kolejności argumentów i zwracająca 0 dla równych argumentów.

wybrać i w dodatku jest to wartość zależna od klasy obiektu[4].

Autorzy przedstawiają zalety miary i w dodatkowych materiałach udowadniają, że LRP jest metryką, tj. zachowuje nierówność trójkąta oraz zamieszczają wyniki miary moLRP dla detektorów obiektów RetinaNet [58], Faster R-CNN [77] oraz SSD [61].

Innym kierunkiem rozwoju oceny detekcji obiektów jest zastąpienie miary nakładania WDO i ODW – IoU. Jedną z propozycji, jest zastąpienie tej wielkości poprzez IoU konturu (ang. *boundary IoU*), zaproponowane w pracy Chenga i in. [12]. Takie podejście do określania zgodności masek ODW pozwala na dokładniejsze ocenianie segmentacji obiektu (zbiór OiM LVIS [31], który korzysta ze zdjęć zbioru COCO, ale dostarcza inne metadane, korzysta właśnie z IoU konturu) i trening lepszych detektorów obiektów z funkcją segmentacji instancji.

W niniejszej pracy zastosowano zarówno miary średniej precyzji, ze świadomością ich ograniczeń, jak i inne miary zależne od progu ufności, ale brane pod uwagę przy podejmowaniu decyzji o zastosowaniu modelu w praktyce (są one łatwe w interpretacji, zrozumiałe dla człowieka, używając popularnego terminu: „interpretowalne"). Oprócz tego, analizowano zmienność czułości i precyzji w funkcji progu ufności, co odpowiada badaniu krzywej precyzja-czułość, ale z dodatkową informacją o wartości progu T_c, dla której miary TPR i PPV osiągają dane wartości. W ten sposób uzyskano informacje wykraczające poza wartość średniej precyzji, np. dla zbliżonych wartością AP modeli RetinaNet i Faster R-CNN zachowanie miar było różne – w ten sposób niezależnie pokazano to, na co wskazują autorzy prac przytoczonych w tej sekcji.

Poszerzeniem badań zrealizowanych w ramach tej pracy może być zbadanie wpływu kompresji stratnej na konkurencyjne względem miar AP miary skuteczności detekcji obiektów.

Niniejsza praca wprawdzie nie dostarcza odpowiedzi na pytanie Goodfellowa – czy modele uczenia głębokiego posiadają prawdziwą percepcję wizualną, a jeżeli tak, to w jakiej relacji jest ona do ludzkiej percepcji wizualnej? Natomiast przeprowadzone w niej badania pozwalają potwierdzić postawione Tezy 1–3, dostarczając interesujących, ale przede wszystkim przydatnych w praktyce, informacji o stosowaniu i trenowaniu głębokich modeli detekcji w warunkach stratnej kompresji obrazu.

[4]Implementacje rzadko umożliwiają osobny wybór T_c dla poszczególnych klas. W praktyce, np. w bibliotece Detectron2 użytej w części empirycznej tej pracy, trzeba by było użyć minimalnego progu i filtrować WDO w dodatkowym kroku przetwarzania po detekcji.

Rozdział 8

Podsumowanie

Przedstawiona rozprawa jest poświęcona problemowi wpływu kompresji stratnej na detekcję obiektów wykorzystującą uczenie głębokie. W celu szczegółowego przebadania tego wpływu zebrano niezbędną wiedzę dostępną w literaturze, która dotyczy działania detektorów obiektów opartych na głębokich konwolucyjnych sieciach neuronowych, oraz wybrano algorytm kompresji JPEG jako reprezentatywny przykład kompresji stratnej. Zgromadzone informacje zostały włączone do teoretycznej części niniejszej rozprawy. Oprócz tego, na ich podstawie zostały zaprojektowane dwa etapy eksperymentów:

- w etapie I analizowano miary skuteczności detekcji obiektów reprezentatywnej grupy dziewięciu pre-trenowanych modeli detekcji na obszernym zbiorze zróżnicowanych obrazów COCO `val2017` (5 tys. obrazów),
- w etapie II przeprowadzono treningi ośmiu modeli podzielonych na dwie architektury i zbadano ich skuteczność detekcji analogicznie jak w etapie I.

W toku eksperymentów obliczeniowych obrazy ze zbioru `val2017` zostały poddane kompresji algorytmem JPEG dla wszystkich możliwych wartości parametru jakości Q, tj. liczb całkowitych od 1 do 100. Na podstawie przeprowadzonych eksperymentów scharakteryzowano zachowanie modeli detekcji i potwierdzono tezy rozprawy:

- Teza 1: Kompresja JPEG nie powoduje spadku precyzji detekcji dla parametru Q z przedziału $[30, 100]$. Przyczyną spadku skuteczności detekcji jest w tym przypadku pogorszenie czułości detekcji.
- Teza 2: Spadek skuteczności detekcji obiektów dla głębokich modeli, mierzony za pomocą miar AP, jest podobny zarówno w modelach jednoetapowych, jak i dwuetapowych, dla parametru Q z przedziału $[30, 100]$.
- Teza 3: Trening detektorów obiektów z użyciem obrazów silnie (Q=20) lub średnio (Q=40) skompresowanych poprawia skuteczność detekcji na obrazach poddanych silnej kompresji, dla parametru Q z przedziału $[5, 30)$. Źródłem poprawy jest zwiększenie czułości detekcji dla tych obrazów.

Potwierdzenie Tezy 1 przynoszą wykresy miar czułości i precyzji jako funkcji parametru Q (Rys. 6.7), a także wartości miar PPV i TPR, które zostały przeanalizowane i zestawione w tabelach dostępnych w załączniku B. Dodatkowym potwierdzeniem są wykresy z II etapu eksperymentów, Rys. 6.13 i 6.12

Uzyskane wyniki pokazują proporcjonalny spadek skuteczności we wszystkich rozpatrywanych modelach, co potwierdza Tezę 2. Dodatkowym potwierdzeniem jest wykres różnicy sąsiednich wartości miary AP (Rys. 6.9). Świadczy o tym również wzajemne podobieństwo wykresów dla poszczególnych modeli, które jest widoczne na wykresach miar z rodziny AP (Rys. 6.8 i Rys. 6.10).

Wyniki pierwszego etapu realizacji celów rozprawy zostały opublikowane jako artykuł [91] w czasopiśmie MDPI Sensors, którego *Impact Factor* wynosi 3.847.

Poszerzone wyniki etapu II realizacji celów niniejszej rozprawy stanowią potwierdzenie Tezy 3. Dane zamieszczone w Dodatku C oraz Tabelach 6.9 i 6.10, oraz Rysunkach 6.13, 6.12, 6.16 i 6.17 demonstrują osiągniętą poprawę miar skuteczności detekcji dla obrazów poddanych nie tylko silnej (Q w przedziale $[5, 30)$ – co postuluje Teza 3), ale również średniej kompresji. Modele trenowane na obrazach poddanych kompresji z parametrem $Q = 40$ osiągały wartości miary AP lepsze lub równe wynikom bazowym, dla niektórych modeli, nawet w przedziale $Q \in [5, 75]$.

Na podstawie części wyników etapu II eksperymentów przygotowano rozdział [90], który został przyjęty do publikacji w monografii „Artificial Intelligence and Data Processing" obejmującej dokonania pracowników Politechniki Śląskiej w zakresie Priorytetowego Obszaru Badawczego „Sztuczna Inteligencja i Przetwarzanie Danych" (POB-2).

Dodatkowymi osiągnięciami związanymi z niniejszą rozprawą są:

- Prezentacja wczesnych wyników na warsztatach dla doktorantów Polsko-Japońskiej Akademii Technik Komputerowych WDSiT 2018 jako referat „Object Detection for Metropolitan Video Surveillance Using a Single Deep Model", Kazimierz Dolny, 2018.
- Rozszerzony abstrakt i wideo-prezentacja „Object detection using deep learning under lossy image compression", warsztaty EEML 2020, Kraków, 2020.
- Opublikowane repozytorium danych etapu I eksperymentów, `https://doi.org/10.7910/DVN/UPIKSF` (dostęp: 27 września 2022).
- Opublikowane repozytorium danych etapu II eksperymentów, `https://doi.org/10.7910/DVN/UHEP3C` (dostęp: 27 września 2022).
- Udostępniony na platformie GitHub kod źródłowy oprogramowania użytego do badan: `https://github.com/tgandor/urban_oculus/` (dostęp: 27 września 2022).

Uzyskana wiedza wytycza dodatkowe kierunki, na które można rozszerzyć przyszłe badania:

- wpływ innych algorytmów kompresji na detekcję obiektów, np. zyskującego popularność algorytmu WebP[1], który również korzysta z transformacji DCT,

[1] `https://developers.google.com/speed/webp/docs/webp_study` (dostęp: 27 września 2022)

- użycie dodatkowych miar skuteczności detekcji, które niosą informacje np. o zmianach dokładności lokalizacji obiektów,
- trening detektorów obiektów na większej liczbie zbiorów uczących i testowanie na innych zbiorach testowych, z uwzględnieniem postępów w dziedzinie treningu detektorów obiektów,
- zbadanie większej liczby architektur i modeli detekcji.

Potrzeba badań w zakresie przedstawionym w rozprawie wynikała z realnego problemu zidentyfikowanego we wdrażanym systemie CityEye, służącym do inteligentnej analizy sekwencji wideo pochodzących z kamer monitoringu miejskiego. Uzyskana w ten sposób wiedza znajdzie zastosowanie w tym systemie.

Bibliografia

[1] Nasir Ahmed, T. Natarajan, Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974. 7, 59

[2] Yancheng Bai, Yongqiang Zhang, Mingli Ding, Bernard Ghanem. Sod-mtgan: Small object detection via multi-task generative adversarial network. *Proceedings of the European Conference on Computer Vision (ECCV)*, strony 206–221, 2018. 11, 31

[3] Saikat Basu, Manohar Karki, Sangram Ganguly, Robert DiBiano, Supratik Mukhopadhyay, Shreekant Gayaka, Rajgopal Kannan, Ramakrishna Nemani. Learning sparse feature representations using probabilistic quadtrees and deep belief nets. *Neural Processing Letters*, 45(3):855–867, 2017. 12

[4] Sebastiano Battiato, Massimo Mancuso, Angelo Bosco, Mirko Guarnera. Psychovisual and statistical optimization of quantization tables for dct compression engines. *Proceedings 11th International Conference on Image Analysis and Processing*, strony 602–606. IEEE, 2001. 7

[5] Yoshua Bengio. *Learning deep architectures for AI*. Now Publishers Inc, 2009. 12

[6] Paul Bergmann, Sindy Löwe, Michael Fauser, David Sattlegger, Carsten Steger. Improving unsupervised defect segmentation by applying structural similarity to autoencoders. *arXiv preprint arXiv:1807.02011*, 2018. 70, 109

[7] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 4, 20, 33, 36, 37

[8] Margaret A. Boden. *GOFAI*, strony 89–107. Cambridge University Press, 2014. 18

[9] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014. 14

[10] Chunhua Chen, Yun Q Shi, Wei Su. A machine learning based scheme for double jpeg compression detection. *2008 19th International Conference on Pattern Recognition*, strony 1–4. IEEE, 2008. 75

[11] Yu Peng Chen, Ying Li, Gang Wang. An enhanced region proposal network for object detection using deep learning method. *PLOS ONE*, 13(9):1–26, 09 2018. 44, 108

[12] Bowen Cheng, Ross Girshick, Piotr Dollár, Alexander C Berg, Alexander Kirillov. Boundary iou: Improving object-centric image segmentation evaluation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, strony 15334–15342, 2021. 113

[13] Matt Crane. Questionable answers in question answering research: Reproducibility and variability of published results. *Transactions of the Association for Computational Linguistics*, 6:241–252, 2018. 73

[14] Navneet Dalal, Bill Triggs. Histograms of oriented gradients for human detection. *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, wolumen 1, strony 886–893. IEEE, 2005. 13, 32

[15] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E Kounavis, Duen Horng Chau. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. *arXiv preprint arXiv:1705.02900*, 2017. 16

[16] Samuel Dodge, Lina Karam. Understanding how image quality affects deep neural networks. *2016 eighth international conference on quality of multimedia experience (QoMEX)*, strony 1–6. IEEE, June 2016. 13

[17] Fu-Jun Du, Shuang-Jian Jiao. Improvement of lightweight convolutional neural network model based on yolo algorithm and its research in pavement defect detection. *Sensors*, 22(9):3537, 2022. 109

[18] Ferda Ernawan, Nur Azman Abu, Nanna Suryana. Tmt quantization table generation based on psychovisual threshold for image compression. *2013 International Conference of Information and Communication Technology (ICoICT)*, strony 202–207. IEEE, 2013. 7

[19] Ferda Ernawan, Nur Azman Abu, Nanna Suryana. An adaptive jpeg image compression using psychovisual model. *Advanced Science Letters*, 20(1):26–31, 2014. 7

[20] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 6, 26, 51, 54, 55, 74, 80

[21] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009. 13, 32

[22] Patrick Follmann, Tobias Bottger, Philipp Hartinger, Rebecca Konig, Markus Ulrich. Mvtec d2s: densely segmented supermarket dataset. *Proceedings of the European conference on computer vision (ECCV)*, strony 569–585, 2018. 74

[23] Patrick Follmann, Bertram Drost, Tobias Böttger. Acquire, augment, segment and enjoy: Weakly supervised instance segmentation of supermarket products. *German Conference on Pattern Recognition*, strony 363–376. Springer, 2018. 110

[24] Cheng-Yang Fu, Mykhailo Shvets, Alexander C Berg. Retinamask: Learning to predict masks improves state-of-the-art single-shot detection for free. *arXiv preprint arXiv:1901.03353*, 2019. 5, 45

[25] Elizabeth Gibney. Google ai algorithm masters ancient game of go. *Nature News*, 529(7587):445, 2016. 27

[26] Ross Girshick. Fast r-cnn. *Proceedings of the IEEE international conference on computer vision*, strony 1440–1448, 2015. 5, 20, 33, 43

[27] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, strony 580–587, 2014. 4, 13, 40, 42

[28] Xavier Glorot, Yoshua Bengio. Xavier initialization. *Journal of Machine Learning Research, 2010b. ISSN*, 15324435, 2010. 25

[29] Ian Goodfellow, Yoshua Bengio, Aaron Courville. *Deep learning*. MIT press, 2016. 3, 18, 19, 20, 27

[30] Ian J Goodfellow, Jonathon Shlens, Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 16

[31] Agrim Gupta, Piotr Dollar, Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, strony 5356–5364, 2019. 74, 113

[32] Matthew Hartley, Tjelvar SG Olsson. dtoolai: Reproducibility for deep learning. *Patterns*, 1(5):100073, 2020. 73

[33] Kaiming He, Ross Girshick, Piotr Dollár. Rethinking imagenet pre-training. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, strony 4918–4927, 2019. 27

[34] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*, strony 2961–2969, 2017. 5, 33, 45

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015. 43

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, strony 770–778, 2016. 26, 36, 82

[37] Paul Henderson, Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. *Asian conference on computer vision*, strony 198–213. Springer, 2016. 52, 108

[38] Geoffrey E Hinton, Simon Osindero, Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 12

[39] Brandon Houghton, Stephanie Milani, Nicholay Topin, William Guss, Katja Hofmann, Diego Perez-Liebana, Manuela Veloso, Ruslan Salakhutdinov. Guaranteeing reproducibility in deep learning competitions. *arXiv preprint arXiv:2005.06041*, 2020. 73

[40] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. 36

[41] Gary B Huang, Marwan Mattar, Tamara Berg, Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008. 12

[42] Xiaojun Jia, Xingxing Wei, Xiaochun Cao, Hassan Foroosh. Comdefend: An efficient image compression model to defend adversarial examples. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, strony 6084–6092, 2019. 17

[43] Lina J Karam, Tong Zhu. Quality labeled faces in the wild (qlfw): a database for studying face recognition in real-world environments. *Human Vision and Electronic Imaging XX*, wolumen 9394, strona 93940B. International Society for Optics and Photonics, 2015. 12, 13

[44] Andrej Karpathy. *Connecting images and natural language*. Praca doktorska, Stanford University, 2016. 2

[45] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, Li Fei-Fei. Large-scale video classification with convolutional neural networks. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, strony 1725–1732, 2014. 4

[46] Peter R Killeen. Mathematical principles of reinforcement. *Behavioral and brain sciences*, 17(1):105–135, 1994. 27

[47] Kye-Hyeon Kim, Sanghoon Hong, Byungseok Roh, Yeongjae Cheon, Minje Park. Pvanet: Deep but lightweight neural networks for real-time object detection. *arXiv preprint arXiv:1608.08021*, 2016. 21

[48] Davis E King. Max-margin object detection. *arXiv preprint arXiv:1502.00046*, 2015. 31, 32, 47

[49] Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 4, 14, 20

[50] Siddharth Krishna Kumar. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*, 2017. 25

[51] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, i in. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, 2020. 74

[52] Michail G Lagoudakis, Ronald Parr. Reinforcement learning as classification: Leveraging modern classifiers. *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, strony 424–431, 2003. 28

[53] Svetlana Lazebnik, Cordelia Schmid, Jean Ponce, i in. Spatial pyramid matching. *Object Categorization: Computer and Human Vision Perspectives*, 3(4), 2009. 43

[54] Yann LeCun, Yoshua Bengio, Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 18

[55] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 12

[56] Jixian Li, Wei Lu, Jian Weng, Yijun Mao, Guoqiang Li. Double jpeg compression detection based on block statistics. *Multimedia Tools and Applications*, 77(24):31895–31910, 2018. 75

[57] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie. Feature pyramid networks for object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, strony 2117–2125, 2017. 33, 34, 82

[58] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár. Focal loss for dense object detection. *Proceedings of the IEEE international conference on computer vision*, strony 2980–2988, 2017. IV, 4, 33, 36, 37, 38, 46, 82, 113

[59] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, C Lawrence Zitnick. Microsoft coco: Common objects in context. *European conference on computer vision*, strony 740–755. Springer, 2014. 5, 26, 55, 74, 81

[60] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, Matti Pietikäinen. Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128(2):261–318, Feb 2020. 32, 74

[61] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C Berg. Ssd: Single shot multibox detector. *European conference on computer vision*, strony 21–37. Springer, 2016. 4, 33, 37, 41, 43, 113

[62] Jonathan Long, Evan Shelhamer, Trevor Darrell. Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, strony 3431–3440, 2015. 35

[63] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 4

[64] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 27

[65] David E Moriarty, Alan C Schultz, John J Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999. 28

[66] Kemal Oksuz, Baris Can Cam, Sinan Kalkan, Emre Akbas. Imbalance problems in object detection: A review. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 30, 31, 46, 108

[67] Kemal Oksuz, Baris Can Cam, Emre Akbas, Sinan Kalkan. Localization recall precision (lrp): A new performance metric for object detection. *Proceedings of the European Conference on Computer Vision (ECCV)*, strony 504–519, 2018. 111, 112

[68] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, Vittorio Ferrari. We don't need no bounding-boxes: Training object class detectors using only human verification. *Proceedings of the IEEE conference on computer vision and pattern recognition*, strony 854–863, 2016. 109

[69] Deepak Pathak, Evan Shelhamer, Jonathan Long, Trevor Darrell. Fully convolutional multi-class multiple instance learning. *arXiv preprint arXiv:1412.7144*, 2014. 35

[70] David MW Powers. What the f-measure doesn't measure: Features, flaws, fallacies and fixes. *arXiv preprint arXiv:1503.06410*, 2015. 112

[71] Majid Rabbani, Rajan Joshi. An overview of the jpeg 2000 still image compression standard. *Signal processing: Image communication*, 17(1):3–48, 2002. 63, 68

[72] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, strony 779–788, 2016. IV, 4, 20, 26, 33, 36, 37, 41, 43

[73] Joseph Redmon, Ali Farhadi. Yolo9000: better, faster, stronger. *Proceedings of the IEEE conference on computer vision and pattern recognition*, strony 7263–7271, 2017. IV, 4, 20, 26, 30, 33, 35, 37, 38, 39, 40

[74] Joseph Redmon, Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. IV, 4, 26, 33, 37, 38, 39, 45, 47, 51, 111

[75] Chuan-Xian Ren, Dao-Qing Dai, Hong Yan. Coupled kernel embedding for low-resolution face image recognition. *IEEE Transactions on Image Processing*, 21(8):3770–3783, 2012. 12

[76] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, strony 91–99, 2015. 5, 20, 26, 33, 38, 44, 82

[77] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 20, 26, 44, 113

[78] Félix Renard, Soulaimane Guedria, Noel De Palma, Nicolas Vuillerme. Variability and reproducibility in deep learning for medical image segmentation. *Scientific Reports*, 10(1):1–16, 2020. 73

[79] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 14, 25, 30, 74

[80] Markus Scholz, Harald Niesch, Olaf Steffen, Baerbel Ernst, Markus Loeffler, Evelin Witruk, Hans Schwarz. Impact of chess training on mathematics performance and concentration ability of children with learning disabilities. *International Journal of Special Education*, 23(3):138–148, 2008. 24

[81] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks, 2014. 36, 39

[82] L Shapiro, G Stockman. *Computer Vision*. Prentice-Hall, New Jersey, USA, 2001. 2, 31

[83] Karen Simonyan, Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 13, 14, 36, 43

[84] Roman Starosolski. „przegląd metod bezstratnej kompresji obrazów medycznych". *Studia Informatica*, 2(58):49–66, 2004. 58

[85] Waqas Sultani, Chen Chen, Mubarak Shah. Real-world anomaly detection in surveillance videos. *Proceedings of the IEEE conference on computer vision and pattern recognition*, strony 6479–6488, 2018. 4

[86] Richard S Sutton, Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 27

[87] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, strony 1–9, 2015. 15

[88] Mingxing Tan, Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning*, strony 6105–6114. PMLR, 2019. 36

[89] JianWen Tao, Wenjun Hu, Shiting Wen. Multi-source adaptation joint kernel sparse representation for visual classification. *Neural Networks*, 76:135–151, 2016. 12

[90] **Tomasz Gandor**, Jakub Nalepa. Training deep convolutional object detectors for images affected by lossy compression. Rozdział w monografii „Artificial Intelligence and Data Processing", obejmującej dokonania pracowników Politechniki Śląskiej w zakresie Priorytetowego Obszaru Badawczego „Sztuczna Inteligencja i Przetwarzanie Danych" (POB-2). 2021. (przyjęty do publikacji). 115

[91] **Tomasz Gandor**, Jakub Nalepa. First gradually, then suddenly: Understanding the impact of image compression on object detection using deep learning. *Sensors*, 22(3):1104, 2022. 115

[92] Miroslav Uhrina, Juraj Bienik, Tomas Mizdos. Chroma subsampling influence on the perceived video quality for compressed sequences in high resolutions. *Advances in Electrical and Electronic Engineering*, 15(4):692–700, 2017. 63

[93] Shimon Ullman, Liav Assif, Ethan Fetaya, Daniel Harari. Atoms of recognition in human and computer vision. *Proceedings of the National Academy of Sciences*, 113(10):2744–2749, 2016. 13

[94] Ivan Valiela. *Doing science: Design, analysis, and communication of scientific research*. Oxford University Press, 2009. 73

[95] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, Kate Saenko. Sequence to sequence-video to text. *Proceedings of the IEEE international conference on computer vision*, strony 4534–4542, 2015. 21

[96] Gaurav Vijayvargiya, Sanjay Silakari, Rajeev Pandey. A survey: various techniques of image compression. *arXiv preprint arXiv:1311.6877*, 2013. 58

[97] Paul Viola, Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004. 32

[98] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992. 7, 58, 59

[99] Yi-Qing Wang. An analysis of the viola-jones face detection algorithm. *Image Processing On Line*, 4:128–148, 2014. 32

[100] Zhou Wang, Alan C Bovik. A universal image quality index. *IEEE signal processing letters*, 9(3):81–84, 2002. 69, 70

[101] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 70

[102] Yoni Wilkenfeld. Can chess survive artificial intelligence? *The New Atlantis*, strony 37–45, 2019. 27

[103] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2` (dostęp: 27 września 2022), 2019. 27, 31, 81

[104] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He. Aggregated residual transformations for deep neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, strony 1492–1500, 2017. 36, 82

[105] Zhaoxia Yin, Hua Wang, Jie Wang, Jin Tang, Wenzhong Wang. Defense against adversarial attacks by low-level image transformations. *International Journal of Intelligent Systems*, 35(10):1453–1466, 2020. 17

[106] Fisher Yu, Vladlen Koltun. Multi-scale context aggregation by dilated convolutions, 2016. 36, 82

[107] Han Zhang, Ian Goodfellow, Dimitris Metaxas, Augustus Odena. Self-attention generative adversarial networks. *International conference on machine learning*, strony 7354–7363. PMLR, 2019. 4

[108] Xizhou Zhu, Han Hu, Stephen Lin, Jifeng Dai. Deformable convnets v2: More deformable, better results. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, strony 9308–9316, 2019. 36

[109] G Zipf. Human behavior and the principle of least effort. addisonwesley, new york. 1949. 48

[110] Jacob Ziv, Abraham Lempel. A universal algorithm for sequential data compression. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 23(3):337–343, 1977. 6

[111] Wilman WW Zou, Pong C Yuen. Very low resolution face recognition problem. *IEEE Transactions on image processing*, 21(1):327–340, 2011. 11

Dodatek A

Wykaz skrótowców i symboli

Skrótowce. W pracy zastosowano następujące skrótowce:

- AI (ang. *artificial intelligence*) – sztuczna inteligencja,
- ANN (ang. *artificial neural networks*) – sztuczne sieci neuronowe,
- AP (ang. *average precision*) – średnia precyzja,
- CE (ang. *cross-entropy*) – entropia krzyżowa,
- CNN (ang. *convolutional neural networks*) – sieci konwolucyjne,
- DC (ang. *dilated convolution*) – konwolucja rozszerzona,
- DCT (ang. *discrete cosine transform*) – dyskretna transformacja cosinusowa,
- DFS (ang. *depth first search*) – przeszukanie (grafu) wgłąb,
- DPI (ang. *dots per inch*) – (liczba) punktów na cal,
- DPM (ang. *deformable part-based model, deformable parts model*) – deformowalny model wieloczęściowy,
- FAIR – jednostka badawcza Facebook AI Research,
- FCN (ang. *fully convolutional networks*) – sieci w pełni konwolucyjne,
- FL (ang. *focal loss*) – skoncentrowana funkcja straty,
- FN (ang. *false negative*) – wynik fałszywy negatywny (bład klasyfikacji lub brak detekcji),
- FP (ang. *false positive*) – wynik fałszywy pozytywny (bład klasyfikacji lub błędna detekcja),

- FPN (ang. *feature pyramid network*) – sieć konwolucyjna zwracająca wielorozdzielczą piramidę cech głębokich,
- HoG (ang. *histogram of oriented gradients*) – histogram zorientowanych gradientów,
- IoT (ang. *Internet of Things*) – Internet rzeczy,
- IoU (ang. *intersection over union*) – współczynnik podobieństwa Jaccarda dla prostokątów, dosł. „część wspólna przez sumę mnogościową”,
- IQR (ang. *inter-quartile range*) – rozstęp kwartylny,
- JFIF (ang. *JPEG file interchange format*) – format plików z kompresją JPEG,
- JPEG (ang. *Joint Photograpic Expert Group*) – algorytm stratnej kompresji obrazów, nazwany skrótowcem grupy roboczej, która go opracowała,
- kNN (ang. *k nearest neighbors*) – k najbliższych sąsiadów,
- mAP (ang. *mean average precision*) – średnia precyzja (zwykle dla progu IoU=0.5),
- MCU (ang. *minimal coding unit*) – minimalna jednostka kodowania (w algorytmie JPEG),
- ML (ang. *machine learning*) – uczenie maszynowe,
- MSE (ang. *mean squared error*) – średni błąd kwadratowy,
- MSRA – jednostka badawcza Microsoft Research Lab Asia,
- NMS (ang. *non-maximum supression*) – usuwanie obserwacji nie-maksymalnych,
- ODW – obiekt do wykrycia,
- OiM (zbiór OiM) – zbiór obrazów i metadanych ODW do detekcji obiektów,
- PCA (ang. *principal component analysis*) – analiza głównych składowych,
- PPV (ang. *positive predictive value, precision*) – precyzja,
- PSNR (ang. *peak signal to noise ratio*) – szczytowy stosunek sygnału do szumu,
- QT (ang. *quantization table*) – tablica kwantyzacji,
- RL (ang. *reinforcement learning*) – uczenie ze wzmocnieniem,
- RLE (ang. *run-length encoding*) – kodowanie długości serii,
- RMSE (ang. *root mean squared error*) – pierwiastek średniego błędu kwadratowego,

- RNN (ang. *recursive neural network*) – rekurencyjna sztuczna sieć neuronowa,
- RPN (ang. *region proposal network*) – sieć propozycji regionów,
- SSIM (ang. *Structural SIMilarity/structual similarity index metric*) – indeks podobieństwa strukturalnego obrazów,
- SVD (ang. *singular value decomposition*) – rozkład macierzy wg wartości osobliwych,
- SVM (ang. *support vector machines*) – maszyny wektorów nośnych,
- TN (ang. *true negative*) – wynik prawdziwy negatywny (tylko poprawna klasyfikacja binarna),
- TP (ang. *true positive*) – wynik prawdziwy pozytywny (poprawna klasyfikacja lub detekcja),
- TPR (ang. *true positive rate, sensitivity*) – czułość,
- UIQI (ang. *universal image quality index*) – uniwersalny indeks jakości obrazu,
- WDO – wynik detekcji obiektów.

Symbole i oznaczenia. Stosowane w tekście symbole należy rozumieć jako:

T_c – próg ufności detekcji,

T_{IoU} – próg IoU zgodności lokalizacji wyników detekcji obiektów względem obiektów do wykrycia,

σ – odchylenie standardowe,

$f(x, y)$ – jasność piksela o współrzędnych (x, y) w obrazie wejściowym,

$F(u, v)$ – wartość współczynnika o współrzędnych (u, v) w obrazie wynikowym DCT.

Dodatek B

Miary skuteczności modeli pre-trenowanych

Tabele zawarte w niniejszym dodatku zawierają wartości miar skuteczności dla zbadanych pre-trenowanych modeli detekcji obiektów. Opisy poszczególnych modeli są zawarte w rozdziale 6.

Miary niezależne od progu ufności T_c (AP, AP_{50}, AP_{75}, AP_l, AP_m, AP_s) zostały obliczone dla WDO o minimalnej ufności $T_c = 0.05$, wynikającej ze względów technicznych. Miary zależne od progu ufności (TP, FP, EX, TPR, PPV) zostały podane dla wartości progów $T_c = 0.5$ i $T_{IoU} = 0.5$.

Miary bezwzględne będące liczbami naturalnymi (TP, FP, EX) zostały przedstawione dokładnie. Pozostałe miary będące wielkościami ułamkowymi z przedziału $[0, 1]$ zostały podane w procentach i zaokrąglone do dziesiątej części procenta.

Zakres wartości parametru Q dla modeli F50-D2 i R50-D2 wynosi od 5 do 96 włącznie, z krokiem 7 (łącznie 14 wartości). Dla pozostałych modeli zakresem wartości parametru Q jest przedział 1–100, ale ze względu na oszczędność miejsca, a także niewielkie zmiany sąsiednich wartości, w tabelach uwzględnione zostały tylko nieparzyste wartości Q.

B.1 Model R101

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
1	0.9	1.6	0.8	1.7	0.7	0.3	43.5	0.6	226	293	5
3	1.2	2.2	1.0	2.2	1.1	0.4	45.8	0.9	334	396	12
5	3.3	6.0	3.2	5.4	3.9	1.1	67.6	3.9	1434	687	39
7	6.7	11.8	6.6	10.0	7.9	2.5	75.5	9.6	3477	1126	97
9	11.0	19.0	11.0	15.9	12.6	4.7	78.6	16.2	5894	1606	152
11	15.3	25.7	15.4	21.5	17.3	6.8	79.0	21.8	7915	2108	236
13	19.2	31.8	19.5	27.0	21.3	8.7	80.1	26.9	9764	2419	288
15	22.5	36.7	23.3	31.1	25.4	10.4	79.9	30.7	11138	2797	331

17	25.6	41.3	26.4	35.2	28.5	11.7	80.3	33.8	12278	3007	397
19	27.4	43.9	28.6	37.7	30.5	12.7	80.4	36.0	13084	3195	436
21	29.3	46.3	30.8	40.1	32.2	13.7	80.8	38.1	13860	3295	474
23	30.7	48.2	32.2	41.7	34.0	14.9	80.7	39.8	14475	3466	496
25	31.9	49.5	33.4	43.2	35.3	15.4	80.9	41.0	14900	3511	513
27	32.9	50.7	34.6	44.2	36.2	16.0	81.0	42.0	15266	3585	533
29	33.5	51.5	35.2	45.1	36.8	16.3	81.1	42.7	15513	3607	571
31	34.2	52.3	36.0	45.8	37.6	16.9	81.1	43.4	15779	3668	560
33	34.9	53.3	36.8	46.9	38.3	17.6	81.4	44.2	16062	3674	572
35	35.4	53.8	37.5	47.5	38.7	17.7	81.6	44.6	16214	3663	583
37	35.7	54.3	38.0	47.8	39.2	18.1	81.4	45.0	16348	3738	588
39	36.0	54.7	38.1	48.1	39.6	18.2	81.6	45.5	16542	3736	616
41	36.2	55.0	38.3	48.6	39.9	18.4	81.4	45.8	16657	3797	611
43	36.5	55.1	38.5	48.7	40.3	18.5	81.7	46.1	16734	3760	621
45	36.8	55.6	39.0	48.9	40.5	19.2	81.8	46.5	16879	3752	636
47	37.0	55.9	38.8	48.8	40.8	19.3	81.8	46.8	17016	3798	646
49	37.2	56.2	39.3	49.5	40.9	19.3	81.7	47.1	17124	3837	643
51	37.3	56.4	39.4	49.4	41.0	19.4	81.6	47.2	17154	3860	647
53	37.5	56.7	39.5	49.7	41.3	19.6	81.5	47.4	17221	3906	655
55	37.6	56.8	39.6	49.8	41.6	19.6	81.7	47.6	17290	3880	651
57	37.8	57.0	39.9	49.7	41.6	20.0	81.8	47.8	17372	3853	660
59	37.9	57.0	40.1	49.9	41.6	19.7	81.7	48.0	17430	3914	657
61	38.0	57.3	40.2	49.9	41.9	20.0	81.7	48.1	17479	3907	698
63	38.3	57.5	40.4	50.2	42.2	20.4	81.9	48.3	17553	3880	721
65	38.5	57.8	41.0	50.6	42.2	20.7	81.9	48.7	17689	3900	697
67	38.6	57.9	41.2	50.6	42.3	20.9	81.8	48.8	17714	3929	712
69	38.8	58.0	41.2	50.8	42.5	21.5	81.9	49.0	17804	3946	703
71	38.8	58.1	41.4	50.9	42.6	21.3	81.8	49.2	17863	3975	722
73	39.0	58.2	41.4	51.1	42.8	21.5	81.8	49.4	17934	3990	716
75	39.1	58.4	41.7	51.4	42.9	21.8	81.6	49.6	18006	4063	709
77	39.2	58.8	41.6	51.4	43.0	21.9	81.7	49.8	18082	4058	718
79	39.5	59.0	42.1	51.6	43.4	22.2	81.7	49.9	18126	4050	735
81	39.4	59.0	41.9	51.3	43.4	22.5	81.7	50.2	18253	4088	759
83	39.6	59.2	42.1	51.6	43.4	22.7	81.5	50.4	18324	4162	765
85	39.7	59.3	42.3	51.4	43.7	23.0	81.3	50.8	18441	4250	788
87	39.9	59.6	42.5	51.6	43.9	23.0	81.3	50.8	18472	4241	804
89	40.1	59.8	42.7	51.9	44.0	23.6	81.2	51.1	18554	4292	802
91	40.2	59.9	42.7	52.2	44.0	23.9	81.3	51.1	18584	4282	802
93	40.2	59.9	42.9	52.3	44.2	23.8	81.1	51.5	18709	4351	804
95	40.4	60.1	43.2	52.1	44.4	23.9	81.1	51.7	18772	4365	824
97	40.4	60.2	43.2	52.3	44.3	23.9	81.1	51.6	18755	4360	808
99	40.4	60.2	43.3	52.1	44.4	23.8	81.1	51.6	18762	4367	817

B.2 Model R101_C4

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
1	0.8	1.5	0.8	1.7	0.7	0.3	6.2	1.5	544	8284	87
3	1.2	2.2	1.2	2.1	1.1	0.5	8.5	2.3	831	8966	125
5	3.7	6.3	3.6	6.4	4.0	1.6	22.2	8.2	2979	10416	615
7	7.5	12.7	7.6	11.9	8.5	2.9	34.7	18.0	6557	12335	1347
9	11.9	20.1	11.9	18.3	13.3	4.7	42.5	28.0	10175	13743	1784
11	16.5	27.3	16.9	25.5	18.4	6.5	46.8	35.7	12959	14711	2178
13	20.8	33.9	21.6	31.8	22.8	8.6	50.1	42.0	15265	15223	2500
15	23.9	38.5	24.9	36.0	26.3	9.8	52.0	46.3	16806	15528	2695
17	26.8	42.8	28.0	39.4	29.8	11.6	53.5	49.7	18044	15673	2849
19	28.8	45.4	30.1	42.2	31.9	11.9	54.8	52.1	18934	15593	3047
21	30.7	47.7	32.4	45.0	34.0	12.9	55.6	54.1	19656	15714	3194
23	32.1	50.0	33.9	46.6	35.9	15.1	56.5	55.8	20258	15598	3230
25	33.1	51.2	35.1	48.2	36.7	14.4	57.1	56.8	20634	15478	3253
27	34.0	52.3	36.3	49.5	37.9	15.0	57.6	57.7	20974	15420	3340
29	34.7	53.3	37.1	49.9	38.6	15.4	58.0	58.6	21298	15404	3377
31	35.2	53.8	37.5	50.1	39.2	15.9	58.0	59.2	21510	15564	3412
33	35.8	54.5	38.0	51.4	39.6	16.6	58.3	59.8	21736	15538	3507
35	36.2	55.0	38.6	51.5	40.3	16.9	58.5	60.3	21895	15555	3560
37	36.5	55.5	39.0	52.0	40.6	16.9	58.6	60.7	22059	15576	3590
39	36.8	55.8	38.9	52.2	41.1	17.4	58.9	61.0	22179	15448	3601
41	37.1	56.2	39.4	52.9	41.2	17.1	59.1	61.4	22322	15439	3620
43	37.1	56.4	39.7	52.5	41.3	17.6	59.2	61.8	22448	15493	3688
45	37.7	57.1	40.2	53.0	41.9	18.1	59.3	62.0	22536	15498	3703
47	37.9	57.3	40.5	53.4	42.3	18.5	59.2	62.3	22630	15579	3760
49	37.9	57.3	40.5	53.4	42.1	18.8	59.1	62.5	22715	15700	3748
51	38.0	57.4	40.7	53.6	42.3	19.0	59.2	62.6	22752	15653	3754
53	38.2	57.7	40.7	53.5	42.7	19.1	59.2	63.0	22891	15754	3811
55	38.3	57.9	41.2	53.9	42.8	19.7	59.3	63.1	22917	15717	3853
57	38.5	58.1	41.1	53.6	42.9	19.8	59.4	63.4	23028	15768	3841
59	38.7	58.4	41.6	54.2	43.0	19.7	59.6	63.5	23074	15661	3909
61	38.9	58.5	41.6	53.9	43.1	19.3	59.5	63.6	23109	15736	3855
63	39.1	58.8	41.9	54.4	43.4	20.3	59.3	63.7	23154	15881	3953
65	39.2	58.9	41.7	54.1	43.7	19.9	59.4	63.9	23216	15873	3960
67	39.3	59.1	42.0	54.4	43.7	20.1	59.4	64.3	23352	15982	3970
69	39.4	59.3	42.1	54.6	43.8	20.4	59.3	64.5	23418	16080	3991
71	39.6	59.5	42.4	54.9	43.9	20.2	59.5	64.8	23559	16066	4010
73	39.5	59.4	42.4	54.8	44.0	20.3	59.4	64.9	23578	16120	4064
75	39.7	59.5	42.5	55.1	44.0	20.3	59.4	65.2	23690	16220	4086
77	39.9	59.8	42.6	55.1	44.2	20.8	59.3	65.2	23703	16270	4152
79	40.0	60.0	42.7	55.3	44.2	20.8	59.3	65.4	23753	16300	4149

81	40.1	59.9	42.9	55.2	44.1	21.0	59.4	65.5	23797	16269	4169
83	40.1	60.1	42.8	55.1	44.1	21.3	59.1	65.7	23888	16518	4228
85	40.3	60.3	43.1	55.3	44.5	21.4	59.1	66.0	23963	16608	4289
87	40.5	60.6	43.5	55.5	44.7	21.5	59.0	66.2	24070	16693	4313
89	40.6	60.9	43.5	55.6	44.8	22.0	59.0	66.5	24148	16793	4371
91	40.7	60.9	43.8	55.6	45.2	22.0	59.0	66.7	24226	16836	4415
93	40.8	60.9	43.9	55.6	45.4	22.1	58.6	66.7	24223	17109	4393
95	41.0	61.3	44.2	55.9	45.5	22.2	58.6	67.0	24333	17165	4448
97	41.0	61.1	44.2	55.6	45.5	22.0	58.5	66.9	24323	17256	4447
99	41.0	61.3	44.2	55.8	45.3	22.1	58.6	67.0	24347	17232	4436

B.3 Model R101_DC5

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
1	0.9	1.6	0.8	1.9	0.8	0.2	9.5	1.5	546	5220	112
3	1.2	2.2	1.2	2.3	1.1	0.4	11.7	2.2	806	6100	172
5	3.6	6.4	3.5	6.0	4.2	1.5	27.5	8.4	3053	8061	705
7	7.2	12.8	7.0	10.9	8.3	2.9	40.0	18.3	6646	9951	1477
9	11.8	20.4	12.0	17.0	13.3	5.1	46.1	27.8	10103	11808	1854
11	16.3	27.7	16.5	23.6	18.0	7.0	49.9	35.7	12965	13016	2219
13	20.2	34.1	20.7	29.6	22.4	8.7	51.9	42.1	15287	14146	2560
15	23.5	39.0	24.6	33.8	26.1	10.2	53.4	46.4	16861	14712	2698
17	26.5	43.4	27.6	37.5	29.6	11.6	54.8	50.1	18221	15027	2914
19	28.2	45.9	29.3	40.1	31.4	12.6	55.8	52.6	19111	15131	3117
21	30.1	48.6	31.4	42.3	33.4	13.5	56.7	54.9	19959	15264	3208
23	31.5	50.4	33.1	44.6	35.0	14.3	56.9	56.5	20542	15549	3298
25	32.6	52.0	34.3	45.9	36.3	14.8	57.3	57.7	20979	15626	3363
27	33.3	52.8	35.2	46.9	37.0	15.2	57.5	58.5	21272	15754	3437
29	34.2	53.7	36.0	48.2	37.7	15.9	57.7	59.4	21578	15826	3507
31	34.9	54.6	36.8	49.1	38.5	16.4	58.2	60.1	21833	15671	3523
33	35.4	55.3	37.6	49.3	39.0	16.7	58.2	60.7	22061	15874	3566
35	35.7	55.7	37.7	49.6	39.2	17.1	58.4	61.4	22299	15899	3673
37	36.1	56.2	38.2	50.0	39.8	17.5	58.6	61.8	22449	15868	3714
39	36.5	56.7	38.7	50.7	40.2	17.7	58.6	62.2	22598	15935	3711
41	36.7	56.9	39.0	50.9	40.7	18.0	58.8	62.5	22696	15916	3715
43	36.8	57.3	39.1	51.0	40.7	18.4	58.8	62.8	22821	15976	3791
45	37.1	57.6	39.4	51.2	41.0	18.2	58.9	62.9	22855	15979	3763
47	37.4	57.8	40.1	51.5	41.5	18.6	59.2	63.4	23046	15910	3824
49	37.6	58.1	40.1	52.0	41.5	18.6	59.2	63.7	23141	15973	3781
51	37.7	58.2	40.2	52.1	41.6	18.9	59.1	63.7	23132	16035	3814
53	38.0	58.4	40.8	52.2	42.0	18.8	58.9	63.9	23230	16223	3901
55	38.0	58.6	41.0	52.2	41.9	19.2	59.0	64.3	23362	16243	3904

57	38.2	58.8	41.1	52.3	42.1	19.6	59.1	64.5	23431	16210	3975
59	38.2	58.7	41.2	52.5	42.2	20.0	59.0	64.5	23418	16253	3965
61	38.3	58.9	40.7	52.5	42.3	19.5	59.2	64.7	23497	16226	3938
63	38.6	59.2	41.2	52.9	42.7	19.6	59.1	64.7	23515	16253	3958
65	38.8	59.4	41.3	53.1	42.9	19.8	59.2	65.1	23640	16317	4057
67	38.9	59.5	41.4	53.1	43.0	19.6	59.3	65.3	23736	16286	4062
69	38.9	59.8	41.6	53.6	42.9	19.7	59.2	65.5	23783	16413	4083
71	39.1	59.9	41.7	53.4	43.3	20.1	59.2	65.8	23906	16492	4140
73	39.2	60.0	42.1	53.5	43.3	20.4	59.1	65.9	23951	16602	4181
75	39.3	60.2	42.1	53.6	43.4	21.0	59.3	66.2	24038	16531	4151
77	39.5	60.3	42.5	53.7	43.6	20.6	59.1	66.4	24115	16663	4172
79	39.6	60.6	42.7	53.3	43.8	21.9	59.2	66.4	24143	16662	4194
81	39.7	60.5	42.8	53.6	43.7	21.2	58.9	66.5	24176	16854	4244
83	39.9	60.8	42.7	53.8	43.9	22.0	59.0	66.8	24289	16913	4314
85	39.9	60.8	42.6	53.7	44.2	22.2	58.9	67.0	24344	17015	4348
87	40.2	61.2	43.0	54.1	44.3	22.5	58.6	67.3	24448	17261	4370
89	40.3	61.3	43.3	54.1	44.5	22.3	58.6	67.5	24531	17307	4403
91	40.3	61.3	43.4	54.2	44.6	22.9	58.5	67.5	24526	17377	4409
93	40.3	61.4	43.5	54.2	44.7	23.0	58.3	67.7	24590	17594	4503
95	40.5	61.5	43.8	54.1	45.0	23.4	58.3	68.0	24699	17690	4474
97	40.7	61.7	44.0	54.4	45.1	22.9	58.2	68.0	24696	17720	4475
99	40.6	61.7	44.0	54.3	45.1	23.3	58.2	68.0	24690	17760	4496

B.4 Model R101_FPN

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
1	0.9	1.7	0.8	1.9	0.8	0.2	12.1	1.1	392	2837	29
3	1.2	2.2	1.1	2.5	1.0	0.4	15.0	1.6	570	3242	44
5	3.4	5.9	3.4	5.5	3.4	1.5	33.4	5.3	1918	3817	219
7	6.7	11.6	6.7	9.6	7.6	3.1	49.9	12.3	4479	4490	548
9	10.9	18.4	11.1	15.6	12.1	5.2	57.9	20.8	7553	5490	846
11	15.4	25.7	15.8	21.4	17.2	7.4	62.6	29.0	10543	6309	1042
13	19.5	32.2	20.2	28.0	21.2	9.4	64.4	35.6	12948	7151	1249
15	23.0	37.7	23.9	32.0	25.3	11.3	66.0	40.4	14683	7571	1388
17	26.3	42.5	27.6	36.7	28.9	12.8	67.1	44.7	16236	7976	1504
19	28.4	45.7	29.8	40.2	30.8	13.8	67.3	47.3	17176	8347	1558
21	30.5	48.5	32.2	42.5	33.2	14.6	68.3	49.9	18124	8411	1689
23	32.1	50.4	34.0	44.4	34.9	16.1	68.5	51.7	18772	8617	1782
25	33.2	51.8	35.2	45.8	36.0	17.0	68.8	53.1	19309	8765	1779
27	34.4	53.4	36.9	47.3	37.4	17.7	69.1	54.2	19697	8825	1825
29	35.1	54.4	37.5	47.9	38.4	18.3	69.2	55.1	20014	8911	1899
31	35.7	54.9	38.4	49.1	38.7	18.7	69.4	55.9	20309	8967	1883

33	36.3	55.9	39.2	49.7	39.3	19.4	69.5	56.6	20559	9023	1930
35	36.8	56.4	39.7	50.3	39.7	19.7	69.7	57.2	20795	9029	1980
37	37.2	56.9	40.0	50.3	40.2	20.0	69.9	57.7	20954	9038	2015
39	37.6	57.5	40.5	50.7	40.8	20.1	69.9	58.3	21181	9106	2005
41	37.8	57.6	40.8	51.2	40.7	20.5	69.8	58.5	21247	9182	2034
43	38.1	58.1	40.9	51.5	41.3	20.5	69.9	58.7	21319	9187	2074
45	38.4	58.4	41.1	51.6	41.3	21.2	70.0	59.0	21453	9175	2094
47	38.6	58.6	41.6	52.0	41.6	21.5	70.0	59.3	21558	9236	2109
49	38.8	58.9	41.8	52.0	41.9	21.4	70.1	59.7	21704	9268	2147
51	38.9	59.0	42.0	52.2	42.2	21.6	70.2	60.0	21791	9243	2126
53	39.1	59.2	42.1	52.1	42.5	21.5	70.0	60.2	21878	9361	2178
55	39.1	59.3	42.1	52.4	42.5	21.2	70.1	60.5	21971	9393	2208
57	39.3	59.6	42.3	52.4	42.6	21.9	70.0	60.7	22050	9467	2196
59	39.5	59.5	42.5	52.8	42.7	21.8	70.0	60.8	22098	9463	2208
61	39.7	59.7	43.0	52.7	43.1	21.9	70.1	61.0	22157	9462	2216
63	40.0	60.1	43.3	52.9	43.6	22.2	70.1	61.4	22312	9534	2253
65	40.2	60.3	43.7	53.3	43.8	22.3	70.0	61.5	22359	9599	2242
67	40.3	60.3	43.9	53.4	43.8	22.7	69.9	61.7	22429	9647	2269
69	40.5	60.6	43.9	53.4	43.9	22.8	69.9	62.1	22564	9700	2293
71	40.6	60.8	44.4	53.5	44.0	23.4	69.9	62.2	22612	9744	2319
73	40.8	60.9	44.3	53.6	44.1	23.7	69.9	62.5	22727	9791	2347
75	40.8	61.0	44.5	53.9	44.1	23.7	69.6	62.5	22703	9901	2359
77	41.0	61.3	44.8	53.9	44.2	24.0	69.7	62.8	22822	9899	2396
79	41.1	61.5	44.8	53.7	44.4	24.1	69.6	63.0	22883	10011	2400
81	41.3	61.4	45.0	53.9	44.6	24.7	69.6	63.1	22929	10036	2449
83	41.4	61.7	44.9	54.1	44.6	24.8	69.4	63.4	23026	10138	2465
85	41.5	61.9	45.0	54.1	44.9	24.6	69.3	63.7	23152	10264	2511
87	41.7	61.9	45.2	54.3	45.0	24.6	69.1	64.0	23245	10374	2553
89	41.8	62.2	45.7	54.4	45.0	25.1	69.2	64.2	23333	10380	2547
91	41.8	62.1	45.6	54.6	45.3	25.2	69.1	64.4	23396	10440	2568
93	41.8	62.3	45.6	54.5	45.3	25.3	68.9	64.6	23471	10613	2595
95	41.9	62.4	45.6	54.6	45.4	25.3	68.9	64.8	23529	10622	2603
97	42.0	62.4	45.9	54.7	45.5	25.4	68.9	64.8	23552	10651	2587
99	42.0	62.5	45.9	54.6	45.5	25.3	68.8	64.9	23574	10691	2593

B.5 Model X101

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
1	0.7	1.4	0.7	1.5	0.6	0.2	24.2	1.0	356	1114	23
3	1.0	1.8	0.9	1.9	0.9	0.3	26.4	1.4	525	1463	38
5	3.3	5.7	3.2	4.7	3.6	1.5	42.8	5.3	1934	2588	230
7	6.4	11.2	6.1	8.7	7.3	2.9	53.5	12.4	4517	3924	505

9	10.5	18.1	10.6	14.8	12.1	4.8	58.9	21.2	7706	5375	788
11	15.0	25.6	15.2	20.4	17.3	7.1	61.8	29.4	10666	6602	989
13	19.4	32.3	20.0	26.0	22.1	9.2	63.8	36.3	13201	7504	1174
15	23.2	38.0	24.1	31.5	26.1	11.2	65.0	41.4	15043	8109	1324
17	26.4	42.9	27.8	35.9	29.5	13.2	66.0	45.9	16673	8584	1436
19	28.7	46.1	30.1	39.0	31.9	14.1	66.4	48.8	17719	8953	1505
21	31.0	49.2	32.7	42.5	34.2	15.6	67.1	51.6	18732	9178	1619
23	32.8	51.8	34.6	44.7	36.0	17.1	67.7	53.5	19436	9279	1663
25	34.0	53.1	35.9	46.0	37.2	18.1	68.2	54.8	19914	9269	1736
27	34.9	54.1	37.1	47.6	37.8	18.4	68.5	56.0	20357	9368	1786
29	35.9	55.4	38.2	48.6	39.0	19.3	69.0	57.2	20774	9328	1881
31	36.6	56.3	39.0	49.7	39.5	19.6	69.4	57.9	21031	9285	1899
33	37.3	57.2	39.7	50.3	40.6	20.2	69.6	58.7	21332	9321	1914
35	37.7	57.6	40.4	50.9	40.8	20.8	69.6	59.2	21511	9376	1992
37	38.2	58.2	40.9	51.0	41.3	21.2	69.8	59.7	21698	9381	1991
39	38.6	58.7	41.3	51.6	41.7	21.5	69.9	60.2	21884	9423	2008
41	38.9	59.0	41.8	51.9	41.9	21.7	70.0	60.5	21965	9400	2038
43	39.1	59.1	41.9	52.1	42.1	22.1	70.0	60.7	22060	9457	2062
45	39.4	59.4	42.1	52.2	42.2	22.2	70.2	60.9	22145	9409	2093
47	39.7	59.8	42.7	52.8	42.8	22.7	70.3	61.2	22242	9381	2107
49	39.8	59.9	42.7	52.7	42.8	22.4	70.4	61.5	22345	9403	2135
51	39.9	60.0	42.8	52.6	43.1	22.6	70.5	61.7	22407	9368	2134
53	40.0	60.3	43.1	52.7	43.3	22.4	70.5	62.1	22565	9425	2150
55	40.2	60.4	43.3	52.6	43.3	23.0	70.4	62.3	22623	9496	2222
57	40.5	60.7	43.8	53.0	43.6	23.7	70.3	62.5	22707	9606	2229
59	40.6	60.8	44.1	53.4	43.6	23.2	70.5	62.6	22762	9542	2176
61	40.8	61.0	44.3	53.4	44.1	23.5	70.6	63.0	22873	9544	2203
63	41.0	61.3	44.5	53.6	44.4	23.7	70.5	63.1	22933	9574	2230
65	41.1	61.4	44.5	53.8	44.4	23.8	70.5	63.3	22997	9645	2236
67	41.3	61.6	44.7	53.9	44.4	24.1	70.4	63.5	23077	9705	2274
69	41.4	61.8	45.1	53.7	44.8	24.5	70.4	63.8	23166	9721	2279
71	41.6	62.0	44.7	54.2	44.9	24.7	70.4	63.9	23204	9738	2307
73	41.6	62.1	45.0	53.8	44.9	24.7	70.4	64.2	23322	9812	2307
75	41.8	62.4	45.5	54.1	45.1	24.9	70.4	64.2	23336	9815	2330
77	42.0	62.5	45.4	54.1	45.1	25.3	70.3	64.5	23435	9880	2358
79	42.2	62.7	46.0	54.4	45.2	26.0	70.4	64.7	23496	9880	2395
81	42.1	62.7	45.7	54.2	45.1	26.2	70.2	64.8	23549	10013	2404
83	42.3	62.8	45.9	54.5	45.3	26.4	70.1	65.2	23675	10085	2428
85	42.4	63.1	46.2	54.3	45.4	26.9	70.2	65.4	23750	10100	2491
87	42.8	63.4	46.6	54.6	45.8	27.0	69.9	65.5	23787	10263	2462
89	42.9	63.5	46.6	54.8	45.8	27.3	70.0	65.8	23905	10228	2477
91	42.9	63.6	46.8	55.0	45.9	27.0	70.0	65.8	23924	10254	2496
93	42.9	63.6	46.6	54.7	45.8	27.0	69.7	65.8	23924	10384	2508
95	43.0	63.7	46.8	54.8	46.1	27.0	69.8	66.3	24074	10433	2543

97	43.1	63.8	47.0	54.9	46.2	27.2	69.8	66.2	24067	10415	2509
99	43.1	63.7	47.1	54.9	46.1	27.0	69.7	66.3	24076	10449	2544

B.6 Model R50

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
1	0.7	1.5	0.6	1.5	0.7	0.2	16.6	0.6	217	1087	3
3	1.0	1.9	0.9	1.8	1.0	0.4	20.0	0.8	293	1170	5
5	2.9	5.3	2.8	4.9	3.3	1.2	48.9	3.3	1193	1245	39
7	6.1	10.7	5.8	8.9	7.0	2.5	68.9	8.5	3091	1396	94
9	9.9	17.2	9.7	14.6	11.2	4.5	75.9	14.8	5380	1712	175
11	13.9	23.7	13.9	19.6	15.9	6.4	78.2	20.6	7478	2080	220
13	17.7	29.7	18.1	24.9	19.8	7.9	79.3	25.2	9153	2389	269
15	20.8	34.2	21.5	29.4	22.9	9.4	79.8	28.8	10481	2649	344
17	23.5	38.2	24.2	32.7	26.0	11.0	79.8	31.9	11594	2926	356
19	25.5	40.9	26.4	35.2	28.0	11.9	80.4	34.0	12344	3002	384
21	27.1	43.5	28.2	37.4	29.7	12.6	80.0	35.8	12996	3242	421
23	28.7	45.3	29.9	39.3	31.5	14.2	80.2	37.5	13614	3358	447
25	29.8	46.8	31.3	40.9	32.8	15.3	80.3	38.7	14067	3458	470
27	30.9	48.0	32.6	42.3	33.9	15.2	80.1	39.5	14357	3572	494
29	31.7	49.2	33.4	43.0	34.8	15.6	80.7	40.5	14716	3527	503
31	32.3	49.9	34.2	44.0	35.4	17.2	80.6	41.1	14938	3605	506
33	32.8	50.7	34.6	44.6	36.0	18.0	80.9	41.9	15238	3598	525
35	33.2	51.1	35.1	44.8	36.2	17.2	80.7	42.4	15407	3679	570
37	33.6	51.6	35.5	45.2	36.8	17.1	80.7	42.8	15544	3711	586
39	34.0	52.2	35.6	45.9	37.3	17.7	80.9	43.3	15723	3724	603
41	34.2	52.5	36.1	46.3	37.4	17.6	81.4	43.7	15887	3640	604
43	34.5	52.8	36.3	46.4	37.8	17.8	81.2	43.8	15902	3692	623
45	34.7	53.1	36.5	46.6	38.0	18.0	81.1	44.1	16025	3743	624
47	35.0	53.4	36.9	47.0	38.1	18.7	81.1	44.4	16115	3747	635
49	35.1	53.5	37.2	47.2	38.2	18.8	81.0	44.6	16194	3792	618
51	35.1	53.6	36.9	47.3	38.3	18.7	81.2	44.7	16242	3772	624
53	35.6	54.2	37.6	47.7	38.7	18.9	81.2	45.1	16371	3784	639
55	35.7	54.3	37.7	47.8	39.1	19.0	81.1	45.3	16448	3822	631
57	35.9	54.6	38.0	48.0	39.2	19.4	81.1	45.6	16557	3857	651
59	36.0	54.8	38.2	48.0	39.5	19.5	81.0	45.6	16576	3892	678
61	36.2	55.0	38.1	48.3	39.5	19.7	81.2	45.9	16660	3865	666
63	36.4	55.3	38.4	48.4	39.7	20.0	81.3	46.3	16808	3865	684
65	36.7	55.6	38.8	48.6	40.0	20.2	81.3	46.4	16872	3891	694
67	36.8	55.8	38.9	48.4	40.3	20.4	81.3	46.6	16926	3899	707
69	36.9	55.8	39.0	48.5	40.2	20.9	81.1	46.8	17006	3952	708
71	37.1	56.1	39.1	48.9	40.6	21.0	81.1	47.2	17132	3980	715

73	37.2	56.2	39.3	49.1	40.7	21.2	81.0	47.4	17207	4032	714
75	37.3	56.4	39.5	49.0	40.7	21.3	81.2	47.6	17286	3993	725
77	37.5	56.6	39.7	49.3	40.8	21.8	81.1	47.8	17370	4042	723
79	37.5	56.7	39.8	49.1	41.0	21.8	81.1	47.9	17401	4049	751
81	37.7	56.8	40.4	49.6	41.3	22.2	81.0	48.0	17449	4085	784
83	37.9	57.0	40.5	49.7	41.4	22.3	80.8	48.3	17551	4169	777
85	38.1	57.2	40.7	49.9	41.6	22.6	80.9	48.7	17679	4161	804
87	38.2	57.5	40.7	49.9	41.7	22.5	80.9	48.8	17743	4177	810
89	38.4	57.6	41.0	50.2	41.9	23.3	81.0	49.1	17848	4191	803
91	38.4	57.8	40.9	50.2	42.1	23.8	80.8	49.3	17906	4253	812
93	38.4	57.7	41.0	50.1	41.9	23.7	80.7	49.4	17962	4307	835
95	38.5	57.8	41.2	50.2	42.2	23.1	80.6	49.6	18024	4327	844
97	38.6	58.0	41.2	50.2	42.3	23.3	80.8	49.6	18024	4293	826
99	38.6	58.0	41.3	50.3	42.2	23.4	80.7	49.7	18046	4309	838

B.7 Model R50_C4

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
1	0.7	1.4	0.7	1.3	0.8	1.2	4.1	1.7	612	14170	96
3	1.0	2.0	1.0	1.7	1.1	0.5	5.3	2.4	867	15623	152
5	3.1	5.9	3.0	5.2	3.5	1.4	15.2	8.3	3000	16679	639
7	6.4	11.3	6.4	10.3	7.9	2.6	25.5	17.4	6308	18407	1379
9	10.4	18.1	10.4	16.5	11.8	4.3	32.9	26.8	9736	19841	1939
11	14.2	24.5	14.4	21.9	16.2	5.6	38.1	34.2	12417	20215	2303
13	17.8	30.3	18.3	27.3	19.8	7.4	41.5	40.1	14575	20556	2624
15	20.9	34.9	21.6	31.8	23.3	8.4	44.4	44.2	16055	20140	2788
17	23.5	38.9	24.3	35.3	26.2	9.9	46.6	47.7	17324	19830	2979
19	25.5	41.7	26.4	37.8	28.4	10.5	48.3	50.2	18243	19544	3171
21	27.4	44.2	28.8	40.8	30.2	11.3	50.0	52.1	18942	18953	3188
23	28.9	46.4	30.5	42.2	32.0	12.5	50.9	53.7	19502	18778	3305
25	30.0	47.8	31.7	43.7	32.9	13.3	52.1	54.6	19842	18225	3373
27	30.9	49.0	32.5	44.9	34.2	13.9	53.0	55.7	20240	17924	3479
29	31.7	50.0	33.5	45.7	34.9	14.3	53.7	56.7	20597	17746	3500
31	32.4	51.0	34.5	47.0	35.8	14.5	54.4	57.3	20823	17471	3493
33	32.9	51.7	34.9	47.9	36.4	14.8	54.8	57.8	21003	17318	3550
35	33.3	52.3	35.5	47.9	37.0	15.1	55.0	58.5	21251	17386	3649
37	33.8	52.6	36.0	48.7	37.3	15.4	55.4	58.8	21381	17233	3703
39	34.1	53.2	36.5	48.9	37.7	15.5	55.9	59.4	21569	17018	3715
41	34.3	53.5	36.6	49.1	37.8	15.9	56.1	59.5	21631	16898	3777
43	34.6	54.0	37.2	49.8	38.2	16.5	56.3	59.8	21744	16882	3800
45	34.7	54.1	37.3	49.5	38.4	16.5	56.3	59.9	21769	16896	3819
47	35.1	54.4	37.8	50.0	39.0	16.9	56.5	60.3	21919	16849	3833

49	35.3	54.6	38.1	50.1	39.1	16.7	56.7	60.5	21989	16818	3861
51	35.2	54.7	37.8	50.0	39.1	17.2	56.6	60.7	22055	16890	3848
53	35.5	55.1	38.2	50.4	39.5	17.0	56.9	61.0	22172	16789	3911
55	35.8	55.3	38.4	51.0	39.6	17.2	56.9	61.2	22242	16879	3947
57	36.0	55.5	38.8	50.9	39.7	17.5	56.9	61.5	22331	16907	3995
59	36.1	55.6	38.9	51.0	39.9	17.4	57.0	61.5	22333	16833	4005
61	36.2	55.8	38.8	51.4	39.9	17.7	57.1	61.4	22324	16797	3997
63	36.4	56.0	38.9	51.2	40.2	18.5	57.1	61.9	22499	16880	3986
65	36.6	56.1	39.2	51.3	40.4	18.7	57.5	62.1	22552	16697	4042
67	36.8	56.4	39.5	51.7	40.5	18.9	57.5	62.1	22565	16654	4082
69	36.9	56.5	39.7	52.0	40.8	18.7	57.4	62.4	22668	16806	4126
71	36.9	56.7	39.4	51.9	40.9	18.7	57.6	62.7	22782	16752	4167
73	37.0	56.9	39.8	52.0	41.0	18.8	57.7	62.8	22829	16758	4184
75	37.1	57.0	39.9	52.2	41.1	19.2	57.7	63.1	22911	16819	4245
77	37.4	57.2	40.2	52.3	41.5	19.0	57.6	63.3	22982	16945	4264
79	37.3	57.3	40.0	52.1	41.1	19.4	57.7	63.3	23017	16869	4270
81	37.6	57.4	40.4	52.2	41.6	18.9	57.8	63.5	23062	16868	4324
83	37.7	57.5	40.5	52.2	41.8	19.2	57.5	63.8	23166	17089	4306
85	37.9	57.8	40.8	52.7	41.9	19.6	57.3	64.0	23254	17318	4381
87	38.0	58.1	40.9	52.7	42.1	19.8	57.4	64.3	23354	17343	4416
89	38.1	58.2	41.1	52.9	42.3	20.0	57.2	64.6	23458	17530	4485
91	38.1	58.2	40.8	52.9	42.4	20.2	57.1	64.7	23501	17651	4504
93	38.2	58.4	41.1	52.7	42.6	21.1	56.9	64.8	23549	17810	4543
95	38.3	58.6	41.3	52.7	42.7	20.8	56.9	65.1	23660	17956	4592
97	38.3	58.7	41.2	52.9	42.6	20.6	56.8	65.2	23685	18034	4558
99	38.4	58.6	41.3	52.9	42.7	21.0	56.8	65.2	23702	18022	4576

B.8 Model R50_DC5

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
1	0.6	1.1	0.5	1.0	0.7	1.3	4.1	1.6	565	13269	108
3	0.8	1.6	0.8	1.4	1.0	0.4	4.9	2.2	784	15331	158
5	2.8	5.3	2.6	4.5	3.4	1.5	16.9	8.1	2947	14520	692
7	6.4	11.7	6.1	9.4	7.9	3.0	31.3	18.1	6570	14407	1397
9	10.5	18.7	10.2	15.4	12.2	4.8	39.4	27.5	9995	15405	1924
11	14.6	25.7	14.5	21.4	16.8	6.3	44.2	35.2	12800	16157	2273
13	18.6	32.1	19.1	27.1	20.9	8.8	47.3	41.3	14990	16676	2613
15	21.7	36.9	22.4	31.8	24.3	10.3	49.2	45.6	16560	17083	2749
17	24.4	41.3	24.8	35.5	27.2	11.9	51.1	49.2	17884	17131	2993
19	26.5	44.3	27.6	37.9	29.3	12.1	52.2	51.8	18824	17215	3104
21	28.3	46.7	29.6	40.6	31.4	12.9	53.3	53.8	19553	17138	3197
23	29.9	48.9	31.2	42.1	32.9	13.9	54.3	55.4	20138	16923	3335

25	30.9	50.1	32.5	43.8	34.1	14.8	54.7	56.5	20516	16974	3390
27	31.7	51.2	33.2	44.5	34.9	14.8	55.2	57.5	20881	16963	3443
29	32.5	52.2	34.3	45.5	35.9	15.4	55.6	58.4	21228	16957	3512
31	33.1	53.0	35.0	46.4	36.6	15.8	56.0	59.0	21430	16818	3530
33	33.7	53.8	35.8	47.1	37.2	17.0	56.3	59.7	21695	16819	3535
35	34.2	54.4	36.2	47.6	37.7	16.3	56.4	60.1	21846	16855	3660
37	34.5	54.8	36.7	48.2	38.1	16.3	56.8	60.6	22016	16773	3740
39	34.9	55.3	37.1	48.5	38.6	17.5	57.0	61.2	22227	16764	3769
41	35.1	55.7	37.2	48.4	38.7	16.8	57.3	61.6	22365	16690	3826
43	35.3	55.7	37.6	48.9	38.9	18.0	57.1	61.5	22362	16814	3820
45	35.6	56.1	38.3	49.3	39.3	17.3	57.1	61.7	22429	16864	3883
47	35.8	56.5	38.3	49.6	39.7	17.6	57.2	61.9	22505	16863	3877
49	36.0	56.7	38.6	50.0	39.6	17.9	57.5	62.2	22604	16717	3911
51	36.2	56.8	38.8	50.0	40.0	18.2	57.5	62.4	22657	16771	3925
53	36.2	57.0	38.7	49.9	40.0	18.0	57.4	62.7	22782	16899	3949
55	36.3	57.1	39.1	49.9	40.2	17.9	57.6	62.8	22821	16823	4018
57	36.5	57.4	39.4	50.1	40.4	18.2	57.5	63.0	22873	16902	3979
59	36.6	57.5	39.4	50.3	40.5	18.1	57.6	63.1	22920	16857	4021
61	36.8	57.6	39.5	50.1	40.8	18.7	57.6	63.1	22930	16846	4036
63	37.0	57.9	39.7	50.2	41.1	18.9	57.7	63.5	23064	16915	4065
65	37.2	58.1	40.0	50.4	41.5	19.0	57.7	63.6	23122	16946	4168
67	37.3	58.3	40.2	50.5	41.5	19.4	57.7	63.9	23205	16981	4162
69	37.4	58.5	40.0	50.9	41.5	20.1	57.9	64.2	23328	16954	4202
71	37.6	58.7	40.1	51.1	41.7	19.4	58.0	64.3	23376	16931	4188
73	37.6	58.8	40.1	51.1	41.8	19.5	57.9	64.6	23461	17086	4194
75	37.8	59.0	40.4	51.3	42.0	19.6	58.1	64.8	23546	16977	4254
77	38.0	59.3	40.8	51.7	42.0	20.1	57.8	65.1	23654	17284	4252
79	38.1	59.2	41.0	51.6	42.0	20.5	57.8	65.1	23651	17262	4294
81	38.2	59.3	41.2	51.8	42.1	20.5	57.9	65.2	23699	17216	4358
83	38.3	59.6	41.1	52.0	42.4	20.4	57.9	65.3	23740	17262	4406
85	38.5	59.6	41.3	51.9	42.8	20.8	57.8	65.7	23873	17453	4512
87	38.7	60.0	41.7	52.0	42.9	20.7	57.6	65.9	23959	17656	4551
89	38.9	60.3	41.8	51.9	43.1	21.3	57.6	66.1	24024	17687	4588
91	38.9	60.2	41.9	52.1	43.1	21.4	57.6	66.2	24058	17735	4590
93	38.9	60.4	41.9	52.3	43.3	21.1	57.2	66.4	24141	18098	4634
95	39.0	60.4	42.0	52.3	43.4	21.4	57.2	66.6	24185	18102	4658
97	39.0	60.5	42.1	52.2	43.4	21.5	57.2	66.6	24216	18092	4655
99	39.0	60.4	42.1	52.5	43.5	21.5	57.2	66.6	24217	18155	4669

B.9 Model R50_FPN

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
1	0.7	1.4	0.7	1.3	0.7	0.2	4.9	1.0	372	7270	17
3	1.0	1.9	1.0	1.7	1.0	0.5	6.3	1.4	523	7732	36
5	3.0	5.4	2.9	4.8	3.4	1.5	25.5	5.5	2015	5872	232
7	6.4	11.4	6.4	9.4	7.4	3.1	46.6	13.0	4740	5423	603
9	10.5	18.3	10.5	15.0	11.8	5.2	56.1	21.6	7843	6131	870
11	14.6	25.1	14.8	21.2	16.6	7.1	59.9	29.0	10523	7041	1084
13	18.5	31.5	19.0	26.5	20.4	9.1	62.2	35.2	12779	7765	1260
15	21.8	36.4	22.5	31.0	24.2	10.7	63.0	39.4	14308	8393	1388
17	24.7	41.0	25.6	34.3	27.3	12.7	64.5	43.6	15827	8708	1517
19	26.7	43.7	27.9	36.9	29.2	13.7	65.0	46.2	16804	9045	1604
21	28.5	46.3	29.8	39.8	31.0	14.5	65.5	48.6	17663	9290	1651
23	30.2	48.6	31.8	41.1	33.4	15.5	65.9	50.4	18327	9499	1781
25	31.1	50.1	32.9	42.5	34.2	16.0	66.2	51.6	18761	9575	1784
27	32.4	51.5	34.4	43.8	35.4	16.8	66.6	52.9	19229	9649	1807
29	33.1	52.5	35.2	44.9	35.9	17.8	66.7	53.9	19570	9751	1898
31	33.8	53.1	36.1	45.6	36.7	18.0	67.1	54.5	19807	9722	1931
33	34.4	53.8	36.9	46.6	37.4	18.7	67.3	55.1	20035	9742	1997
35	35.0	54.6	37.4	47.5	37.7	19.1	67.4	55.9	20328	9819	2084
37	35.3	55.1	37.9	47.7	38.4	19.3	67.6	56.3	20463	9811	2080
39	35.7	55.7	38.4	48.1	38.9	19.3	67.7	57.0	20698	9869	2100
41	36.1	56.0	38.9	48.7	39.1	19.9	67.7	57.3	20815	9917	2119
43	36.2	56.1	39.1	48.6	39.2	19.7	67.8	57.5	20896	9945	2159
45	36.6	56.4	39.2	49.1	39.7	20.2	67.8	57.8	21019	9960	2173
47	36.7	56.7	39.7	48.8	39.9	20.4	67.8	58.2	21133	10021	2201
49	37.1	57.0	40.1	49.7	40.1	20.4	68.0	58.5	21240	9978	2197
51	37.1	57.0	40.4	49.6	40.3	20.4	68.1	58.7	21322	10003	2243
53	37.2	57.3	40.5	49.8	40.5	20.7	67.9	59.1	21459	10165	2239
55	37.5	57.6	40.7	49.9	40.7	20.8	67.8	59.2	21503	10193	2271
57	37.7	57.7	40.9	50.3	40.9	21.1	67.8	59.5	21602	10261	2288
59	37.8	57.9	41.0	49.9	40.9	21.1	68.0	59.8	21715	10236	2281
61	37.9	58.0	41.2	50.3	41.0	21.2	68.0	60.0	21789	10245	2295
63	38.0	58.3	41.2	50.4	41.0	21.3	68.1	60.2	21874	10253	2297
65	38.3	58.6	41.6	50.6	41.5	21.5	68.0	60.3	21910	10312	2354
67	38.5	58.7	41.9	50.6	41.5	21.8	67.8	60.4	21959	10443	2390
69	38.7	59.1	42.1	50.9	41.6	22.3	67.9	60.9	22125	10476	2413
71	38.8	59.3	42.3	51.2	41.8	22.2	67.9	61.2	22245	10515	2439
73	39.0	59.3	42.5	51.1	42.1	22.3	67.9	61.5	22333	10561	2430
75	39.0	59.5	42.4	51.1	42.2	22.7	67.9	61.6	22378	10574	2488
77	39.2	59.8	42.9	51.3	42.3	22.8	67.8	61.8	22449	10685	2519
79	39.1	59.7	42.4	51.1	42.2	22.9	67.6	62.0	22517	10784	2533

81	39.4	59.7	42.8	51.7	42.5	23.2	67.7	61.9	22502	10749	2584
83	39.4	59.9	42.8	51.6	42.6	22.8	67.5	62.3	22620	10907	2593
85	39.6	60.3	43.2	51.6	42.9	23.2	67.2	62.5	22706	11065	2649
87	39.8	60.5	43.3	51.7	43.0	23.4	67.3	62.9	22856	11095	2672
89	39.9	60.6	43.5	52.0	43.1	23.5	67.2	63.2	22948	11182	2666
91	39.9	60.6	43.5	51.9	43.1	23.8	67.2	63.2	22976	11203	2702
93	40.0	60.7	43.4	51.8	43.3	23.9	67.0	63.4	23028	11365	2732
95	40.1	60.9	43.6	51.8	43.5	24.0	66.9	63.7	23141	11464	2756
97	40.2	60.9	43.7	51.8	43.5	24.0	67.1	63.7	23147	11363	2732
99	40.2	61.0	43.9	51.8	43.5	24.2	67.0	63.7	23135	11385	2759

B.10 Model F50-D2

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
5	3.1	5.6	2.8	4.8	3.4	1.6	30.6	5.5	1999	4533	263
12	15.3	26.6	15.7	22.0	17.3	7.4	59.8	30.0	10897	7316	1133
19	24.9	41.7	25.9	35.0	27.3	12.1	64.9	44.0	16002	8654	1570
26	30.0	48.7	31.7	41.1	32.8	15.1	66.5	50.2	18245	9207	1739
33	32.3	51.8	34.7	43.9	35.3	16.4	67.1	53.2	19325	9469	1937
40	33.7	53.4	36.0	45.4	36.8	17.4	67.3	54.9	19961	9678	2004
47	34.7	54.8	37.3	46.4	37.5	18.5	67.6	56.4	20477	9804	2095
54	35.2	55.5	37.9	46.7	38.3	18.7	67.6	57.0	20703	9906	2150
61	35.7	56.0	38.5	47.2	39.0	19.2	67.5	57.8	20989	10107	2215
68	36.3	56.8	39.3	48.0	39.7	20.2	67.6	58.8	21356	10258	2312
75	36.7	57.4	39.7	48.1	40.1	20.7	67.6	59.5	21630	10371	2385
82	37.2	57.8	40.3	48.5	40.4	21.2	67.4	60.2	21865	10586	2489
89	37.7	58.4	41.0	48.9	40.9	22.5	66.8	61.1	22218	11047	2600
96	37.9	58.8	41.1	49.2	41.1	22.6	66.6	61.8	22450	11236	2649

B.11 Model R50-D2

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
5	3.0	5.4	2.8	5.1	3.3	2.4	62.4	2.7	973	586	23
12	15.6	26.3	16.0	22.2	17.8	7.2	84.0	19.8	7193	1373	193
19	25.1	40.6	26.0	34.9	27.8	11.8	84.9	30.5	11084	1966	335
26	29.6	46.9	30.8	40.3	33.1	14.5	84.6	35.5	12914	2350	407
33	31.9	49.9	33.5	43.1	35.4	16.3	84.8	38.2	13874	2478	447
40	33.2	51.6	35.1	44.6	37.2	17.1	84.8	39.8	14452	2590	490
47	34.2	52.7	36.3	45.6	38.1	18.5	84.7	40.7	14787	2675	540
54	34.8	53.5	37.3	46.2	38.8	18.7	84.7	41.6	15112	2734	550

61	35.2	54.1	37.4	46.3	39.4	18.9	84.6	42.3	15368	2793	585
68	35.8	54.7	38.3	47.0	40.0	20.1	84.7	43.1	15665	2831	616
75	36.2	55.2	38.6	47.4	40.3	20.3	84.5	43.7	15877	2923	623
82	36.6	55.9	39.3	47.5	40.7	20.9	84.1	44.5	16175	3049	651
89	37.2	56.4	40.2	48.0	41.4	22.2	83.8	45.3	16449	3177	695
96	37.4	56.7	40.2	48.3	41.5	23.0	83.7	45.8	16645	3253	720

Dodatek C

Miary skuteczności modeli wytrenowanych

Tabele zawarte w niniejszym dodatku zawierają wartości miar skuteczności dla modeli detekcji obiektów, które zostały wytrenowane, a następnie zbadane w ramach tej pracy.

Opisy poszczególnych modeli i sposobów ich treningu są zawarte w rozdziale 6. Znaczenie przedstawionych miar oraz przyjęte progi ufności i zaokrąglenia są takie same jak w Dodatku B.

Zakres wartości parametru Q dla wszystkich modeli wynosi od 5 do 96 włącznie, z krokiem 7 (łącznie 14 wartości).

C.1 Model F50-STD

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
5	2.4	4.4	2.2	3.5	2.7	1.2	23.4	4.9	1794	5875	313
12	11.9	20.8	12.0	15.8	13.2	6.6	55.6	25.3	9206	7351	1134
19	20.6	35.0	21.4	27.9	22.8	10.4	62.9	38.6	14009	8263	1530
26	26.2	43.4	27.2	35.5	28.5	13.0	65.3	45.8	16631	8847	1698
33	28.7	46.8	30.3	38.8	31.1	14.3	65.8	49.3	17911	9306	1903
40	30.1	48.9	32.1	40.4	32.8	15.2	66.4	51.1	18577	9414	2002
47	31.0	50.2	33.1	41.5	33.5	15.8	66.3	52.4	19036	9683	2115
54	31.7	51.1	33.7	42.6	34.3	16.2	66.5	53.4	19408	9786	2183
61	32.2	51.7	34.5	42.4	35.0	16.6	66.8	54.1	19670	9790	2212
68	32.9	52.5	35.0	43.2	35.6	17.6	66.4	55.0	19975	10095	2364
75	33.3	53.2	35.7	43.6	36.0	17.7	66.4	55.9	20319	10266	2441
82	33.7	53.5	36.3	43.2	36.5	18.6	66.5	56.6	20565	10369	2472
89	34.2	54.2	36.7	44.0	37.0	18.8	65.9	57.5	20904	10825	2629
96	34.3	54.6	37.3	44.1	37.1	19.2	65.8	58.2	21155	11017	2691

C.2 Model F50-Q20

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
5	10.4	18.6	10.4	14.6	12.0	4.9	49.6	22.1	8021	8155	1025
12	27.1	45.1	28.4	38.5	29.2	12.6	63.9	47.6	17279	9745	1865
19	29.7	48.4	31.7	40.8	31.7	14.7	64.5	51.6	18737	10320	2113
26	30.7	49.8	32.6	41.5	32.6	15.6	64.5	53.1	19303	10605	2257
33	31.0	50.3	33.0	41.3	33.0	16.2	64.1	53.8	19553	10956	2346
40	31.2	50.4	33.3	41.5	33.3	16.6	63.9	54.3	19727	11157	2434
47	31.4	50.7	33.4	41.7	33.3	16.7	63.5	54.6	19828	11397	2503
54	31.4	50.8	33.5	41.8	33.5	16.7	63.5	54.8	19903	11441	2552
61	31.5	51.0	33.8	41.7	33.7	17.0	63.4	55.1	20014	11549	2631
68	31.8	51.2	34.0	41.6	33.9	17.2	63.4	55.5	20158	11653	2677
75	31.7	51.1	33.7	41.6	33.8	17.3	63.1	55.6	20192	11794	2701
82	31.7	51.2	33.7	41.5	33.7	17.5	62.9	55.8	20293	11973	2759
89	31.8	51.5	34.0	41.6	33.9	17.5	62.7	56.3	20449	12174	2817
96	31.7	51.4	33.7	41.4	33.9	17.8	62.4	56.3	20473	12359	2870

C.3 Model F50-Q40

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
5	5.0	8.9	4.9	7.0	5.8	2.2	35.7	10.2	3705	6661	557
12	21.5	36.3	22.2	29.5	23.6	11.1	61.9	39.7	14428	8879	1563
19	28.5	46.8	29.9	39.5	30.5	13.9	65.6	49.1	17852	9350	1911
26	30.7	49.7	32.7	42.1	32.7	15.2	65.9	52.2	18950	9798	2089
33	31.7	50.9	33.5	43.1	33.6	16.4	65.6	53.6	19466	10220	2179
40	32.0	51.5	34.3	43.5	34.0	17.0	65.6	54.2	19704	10322	2313
47	32.2	51.8	34.2	43.3	34.4	17.2	65.2	54.8	19918	10621	2414
54	32.5	52.1	35.0	43.3	34.9	17.6	65.1	55.2	20049	10727	2429
61	32.7	52.3	35.2	43.5	34.9	17.8	65.0	55.6	20202	10867	2446
68	33.0	52.7	35.2	43.6	35.3	18.0	64.7	56.0	20346	11089	2548
75	32.9	52.7	35.5	43.3	35.4	18.2	64.7	56.5	20535	11204	2627
82	33.1	52.7	35.6	43.6	35.3	18.2	64.4	56.7	20594	11371	2656
89	33.2	52.7	35.7	43.9	35.4	18.4	64.0	57.2	20782	11692	2751
96	33.2	52.9	35.7	43.7	35.5	18.7	63.8	57.5	20876	11864	2773

C.4 Model F50-T20

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
5	10.6	18.7	10.7	14.7	12.5	5.0	49.3	21.8	7914	8131	973
12	29.5	48.1	31.2	41.5	32.1	14.0	65.0	49.9	18118	9769	1870
19	32.4	51.9	34.6	44.3	34.9	16.5	65.5	54.0	19639	10350	2157
26	33.4	53.1	35.7	45.0	36.1	17.6	65.3	55.6	20185	10727	2285
33	33.8	53.5	36.1	45.1	36.7	18.1	65.0	56.4	20499	11056	2447
40	33.8	53.7	36.2	45.1	36.6	18.7	64.9	56.9	20669	11182	2499
47	34.2	53.9	36.7	45.3	36.8	19.2	64.5	57.3	20807	11453	2571
54	34.1	54.0	36.7	45.2	37.0	19.3	64.4	57.5	20877	11542	2622
61	34.2	54.1	36.7	45.3	37.1	19.2	64.5	57.9	21049	11595	2661
68	34.3	54.3	37.0	45.3	37.2	19.2	64.2	58.1	21102	11771	2759
75	34.4	54.4	36.9	45.3	37.1	19.5	64.1	58.5	21243	11874	2805
82	34.3	54.2	36.7	45.2	37.1	19.7	63.7	58.5	21238	12113	2842
89	34.5	54.6	36.9	45.2	37.2	20.2	63.4	58.8	21372	12333	2953
96	34.5	54.5	36.9	45.1	37.3	20.2	63.3	58.9	21408	12408	2966

C.5 Model F50-T40

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
5	5.6	10.0	5.6	8.1	6.4	2.8	37.4	11.1	4033	6758	633
12	23.7	39.6	24.9	33.0	26.2	11.8	62.9	42.0	15263	9019	1660
19	30.8	49.8	32.7	42.3	33.1	15.3	65.8	51.7	18799	9757	2030
26	32.9	52.5	35.2	44.3	35.5	17.4	65.9	54.8	19929	10293	2193
33	34.0	53.7	36.5	45.3	36.7	18.5	65.8	56.3	20450	10615	2351
40	34.4	54.2	37.3	45.8	37.1	18.6	65.8	57.0	20709	10764	2444
47	34.7	54.7	37.5	46.0	37.4	19.5	65.6	57.6	20926	10984	2532
54	34.9	54.9	37.8	45.7	37.6	19.8	65.2	58.0	21072	11238	2626
61	35.1	55.2	37.8	46.2	38.1	19.2	65.4	58.5	21250	11250	2641
68	35.4	55.4	38.6	46.1	38.4	20.0	64.9	58.9	21403	11598	2744
75	35.4	55.6	38.2	46.0	38.5	19.9	64.7	59.4	21580	11761	2755
82	35.4	55.6	38.2	45.8	38.3	20.7	64.1	59.5	21627	12096	2835
89	35.6	55.7	38.4	46.5	38.3	20.7	63.8	60.0	21791	12347	2971
96	35.6	55.8	38.4	46.4	38.3	20.8	63.3	60.4	21930	12722	2975

C.6 Model R50-STD

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
5	2.8	5.0	2.7	4.4	3.3	1.3	60.4	2.4	859	564	33
12	14.1	23.8	14.4	19.2	16.0	6.5	84.0	17.9	6521	1241	198
19	22.7	36.9	23.6	31.0	24.9	11.9	85.8	27.4	9961	1643	315
26	26.9	42.8	28.1	36.3	29.6	13.2	85.9	32.2	11711	1923	381
33	29.1	45.7	30.5	39.0	32.1	15.7	85.7	34.9	12676	2122	424
40	30.3	47.3	32.3	40.4	33.6	15.5	85.4	36.1	13128	2252	481
47	31.1	48.4	33.0	41.2	34.3	16.1	85.5	37.3	13536	2290	497
54	31.7	49.0	33.8	42.0	35.1	16.6	85.3	38.0	13811	2384	513
61	32.1	49.7	34.3	42.3	35.7	17.2	85.2	38.8	14085	2440	543
68	32.7	50.5	35.0	42.9	36.5	17.3	85.2	39.5	14354	2490	556
75	33.1	51.1	35.4	43.5	36.8	17.6	85.1	40.2	14598	2564	578
82	33.5	51.4	35.8	43.5	37.1	18.4	84.7	40.8	14818	2687	619
89	34.1	52.2	36.7	43.7	37.8	19.5	84.5	41.6	15121	2780	659
96	34.3	52.5	36.6	44.1	38.0	19.6	84.4	42.2	15327	2833	669

C.7 Model R50-Q20

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
5	10.5	17.9	10.8	14.5	12.1	5.2	81.0	12.8	4662	1095	168
12	27.5	44.1	28.8	38.4	30.3	13.2	85.0	34.3	12479	2199	471
19	30.4	47.6	32.1	41.6	33.4	15.8	84.4	38.2	13879	2569	533
26	31.3	48.8	33.3	42.3	34.3	17.0	83.6	39.4	14320	2810	604
33	31.7	49.2	33.5	42.6	34.8	17.7	83.4	40.2	14614	2903	636
40	32.0	49.6	33.8	42.5	35.2	16.9	83.4	40.6	14757	2938	667
47	32.1	49.7	33.9	42.3	35.2	17.3	83.3	41.0	14883	2990	677
54	32.1	49.8	34.0	42.5	35.3	17.4	83.3	41.3	14997	2999	702
61	32.3	50.0	34.3	42.7	35.6	17.4	83.2	41.6	15098	3047	724
68	32.3	50.1	34.4	42.7	35.8	17.4	82.7	41.8	15195	3169	721
75	32.3	50.1	34.3	42.3	35.7	17.5	82.7	42.0	15267	3201	730
82	32.3	50.1	34.3	42.4	35.6	17.5	82.6	42.2	15346	3240	764
89	32.4	50.2	34.5	42.1	35.7	17.5	82.3	42.6	15474	3318	797
96	32.4	50.3	34.4	42.2	35.8	17.5	82.1	42.7	15512	3373	809

C.8 Model R50-Q40

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
5	5.1	9.1	5.1	7.1	5.8	2.8	72.9	5.3	1919	714	62
12	22.2	36.4	23.2	29.9	25.1	10.7	84.4	27.4	9960	1834	311
19	28.9	45.7	30.5	39.1	31.7	14.4	84.5	35.4	12877	2354	431
26	31.1	48.5	33.1	41.6	34.4	16.8	84.4	38.1	13853	2559	494
33	32.0	49.8	34.0	42.5	35.5	18.0	84.2	39.4	14331	2681	527
40	32.5	50.3	34.6	42.8	36.3	16.9	84.0	40.2	14591	2789	566
47	32.9	50.9	35.3	42.9	36.6	17.7	83.9	40.7	14793	2842	586
54	33.0	50.9	35.3	43.1	36.7	18.3	84.0	41.0	14896	2827	600
61	33.1	51.3	35.3	42.8	36.8	17.6	83.9	41.4	15040	2895	621
68	33.3	51.5	35.7	43.1	37.0	19.0	83.4	41.9	15217	3029	629
75	33.5	51.7	35.7	43.3	37.2	18.8	83.5	42.1	15302	3014	646
82	33.4	51.7	35.6	43.0	37.2	18.5	83.4	42.3	15387	3062	680
89	33.6	51.8	36.0	43.0	37.4	18.5	83.1	42.8	15548	3173	710
96	33.6	51.8	36.0	43.0	37.4	18.9	82.8	43.1	15646	3245	721

C.9 Model R50-T20

Q	AP	AP_{50}	AP_{75}	AP_l	AP_m	AP_s	PPV	TPR	TP	FP	EX
5	11.2	19.1	11.2	15.5	12.9	5.7	81.5	13.3	4838	1097	157
12	29.4	46.5	30.7	41.2	32.4	14.5	85.6	35.2	12788	2159	437
19	32.3	50.3	34.3	43.8	35.7	15.8	85.3	39.0	14175	2437	565
26	33.2	51.4	35.3	44.3	36.9	18.5	85.1	40.5	14723	2586	602
33	33.7	52.2	36.0	44.4	37.5	18.8	85.1	41.2	14984	2629	642
40	34.0	52.4	36.1	44.7	37.9	18.4	84.8	41.8	15175	2718	675
47	34.2	52.6	36.6	45.2	38.0	18.5	84.6	42.0	15263	2789	706
54	34.2	52.7	36.7	45.2	38.0	19.2	84.3	42.3	15381	2856	724
61	34.3	52.9	36.6	45.0	38.1	18.4	84.1	42.5	15453	2912	737
68	34.5	53.2	36.8	45.0	38.5	18.7	84.3	43.0	15606	2916	760
75	34.6	53.2	37.0	45.2	38.5	19.5	84.3	43.2	15679	2924	768
82	34.6	53.1	36.9	45.2	38.4	18.9	84.1	43.3	15750	2981	815
89	34.7	53.2	37.1	45.2	38.6	20.2	83.7	43.6	15847	3083	820
96	34.6	53.1	37.1	45.0	38.7	19.4	83.6	43.8	15898	3115	826

C.10 Model R50-T40

Q	AP	AP_{50}	AP_{75}	AP_{l}	AP_{m}	AP_{s}	PPV	TPR	TP	FP	EX
5	5.7	9.8	5.6	8.1	6.3	3.5	72.9	5.6	2042	759	70
12	23.9	38.9	24.9	33.1	26.4	12.2	85.7	28.6	10404	1741	349
19	31.0	48.5	32.6	42.3	33.6	16.7	85.8	36.7	13352	2214	479
26	33.1	51.3	35.2	44.4	36.2	18.5	85.4	39.6	14381	2453	548
33	34.0	52.5	36.0	44.9	37.2	19.7	85.3	40.8	14838	2549	606
40	34.4	52.9	36.6	44.9	38.0	19.5	84.9	41.5	15094	2692	639
47	34.7	53.4	36.9	45.2	38.1	20.1	84.8	42.2	15317	2749	658
54	35.0	53.7	37.1	45.4	38.5	20.6	84.6	42.5	15436	2806	666
61	35.2	54.1	37.5	45.6	38.7	20.4	84.7	42.9	15601	2821	705
68	35.3	54.1	37.5	45.6	38.8	20.8	84.5	43.4	15768	2882	712
75	35.3	54.2	37.5	45.6	38.8	21.0	84.4	43.7	15882	2944	740
82	35.4	54.2	37.8	45.5	38.8	21.1	83.9	43.8	15928	3060	758
89	35.5	54.4	37.9	45.5	39.1	21.6	83.7	44.4	16140	3148	809
96	35.5	54.5	37.9	45.5	39.1	21.7	83.7	44.6	16213	3156	816

Dodatek D

Kody źródłowe programów

D.1 Wykonanie detekcji obiektów

Program `custom_validate_coco.py` w Listingu D.1 służy do wykonania detekcji obiektów (tzw. *inferencji modelu*) na zdegradowanych obrazach zbioru COCO `val2017`. Parametrami programu są: nazwa modelu, ścieżki plików konfiguracyjnego i wag modelu, minimalna i maksymalna wartość parametru Q (obie z zakresu $[1, 100]$, podanie wartości większej od 100 spowoduje wykonanie modelu na oryginalnym zbiorze), oraz krok, z którym mają być uwzględniane wartości parametru Q. Program wymaga obecności na dysku pliku archiwum `val2017.zip`, z którego przed każdą inferencją ekstrahowane są obrazy zbioru testowego.

Dla każdej wartości parametru Q, która wynika z zadanego przedziału (oraz kroku), jest uruchamiana degradacja obrazów (kompresja JPEG „w miejscu"), a następnie inferencja za pomocą biblioteki Detectron2, z zapisaniem wyników do katalogu `out_folder` Podanie ścieżki wyjściowej powoduje również zapisanie pliku JSON z wykrytymi obiektami (`coco_instances_results.json`), oraz pliku `results.json`, który zawiera część miar skuteczności (niezależne od progu ufności: miarę AP, miary AP dla poszczególnych klas, a także miary AP_{50}, AP_{75}, AP_l, AP_m oraz AP_s).

Listing D.1: custom_validate_coco.py

```
import argparse, logging, os, time, json

import torch
from detectron2.config import get_cfg
from detectron2.data import build_detection_test_loader
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
from detectron2.engine import DefaultPredictor

logging.basicConfig(level=logging.INFO)

def validate_quality(
    q, model: str, config: str, weights: str, min_score: str, verbose: bool = False
):
    if not os.path.exists("datasets"):
        print("Copy or symlink datasets/ directory here.")
        exit()
    if not os.path.exists("val2017.zip"):
        print("Copy or symlink val2017.zip file here.")
```

```
        exit()

    out_folder = f"evaluator_dump_{model}_{q:03d}"
    pre_unzip = time.time()
    os.system(f"unzip {'' if verbose else '-q'} -o val2017.zip -d datasets/coco/")
    post_unzip = time.time()
    if 1 <= q <= 100:
        os.system(f"mogrify {'-verbose' if verbose else ''} -quality {q} datasets/coco/val2017/*")
    else:
        print("Skipping quality degradation.")
    post_degrade = time.time()
    print(
        f"Done unzipping in {post_unzip-pre_unzip:.1f} s, "
        f"degrading in {post_degrade-post_unzip:.1f} s, "
        f"total {post_degrade-pre_unzip:.1f} s"
    )
    cfg = get_cfg()
    cfg.merge_from_file(config)
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = min_score  # R-CNN
    cfg.MODEL.RETINANET.SCORE_THRESH_TEST = min_score  # RetinaNet
    cfg.MODEL.WEIGHTS = weights
    predictor = DefaultPredictor(cfg)
    data_loader = build_detection_test_loader(cfg, "coco_2017_val")
    evaluator = COCOEvaluator("coco_2017_val", cfg, False, out_folder)
    start = time.time()
    results = inference_on_dataset(predictor.model, data_loader, evaluator)
    inference_time = time.time() - start
    print(results)
    torch.save(results, out_folder + "/results.pth")
    with open(out_folder + "/results.json", "w") as jsf:
        json.dump(
            {
                "quality": q,
                "model": model,
                "results": results,
                "elapsed": inference_time,
                "device": torch.cuda.get_device_name(),
            },
            jsf,
        )

def _parse_cli():
    parser = argparse.ArgumentParser()
    parser.add_argument("model")
    parser.add_argument("config")
    parser.add_argument("weights")
    # default: single -- no degradation!
    parser.add_argument("--minQ", "-m", type=int, help="min JPEG quality", default=101)
    parser.add_argument("--maxQ", "-M", type=int, help="max JPEG quality", default=101)
    parser.add_argument("--stepQ", "-S", type=int, help="JPEG quality step", default=1)
    parser.add_argument(
        "--min-score",
        "-t",
        type=float,
        help="score threshold for objects",
        default=0.05,
    )
    parser.add_argument("--verbose", "-v", action="store_true")
    return parser.parse_args()

def main():
    args = _parse_cli()
    for i in range(args.minQ, args.maxQ + 1, args.stepQ):
        validate_quality(
            i, args.model, args.config, args.weights, args.min_score, args.verbose
        )
```

```
if __name__ == "__main__":
    main()
```

D.2 Wyznaczanie wartości miar skuteczności

Program `evaluate_coco.py` w Listingu D.2 służy do wyznaczania wartości miar skuteczności detekcji obiektów na podstawie pliku JSON z wykrytymi obiektami, który został wygenerowany przez program `custom_validate_coco.py`.

Jako argumenty można podać próg ufności (odrzucając obiekty o niższej ufności, tak jakby nie były zwrócone przez detektor), oraz flagę `--full`, która powoduje wyliczenie wszystkich miar na nowo (w przeciwnym razie wartości miar z pliku `results.json` są zwracane bez modyfikacji).

Dla każdego folderu wyników (o nazwie w postaci `evaluator_dump_<model>_<Q>`) są wyliczane miary TPR, PPV, TP, FP i EX. Następnie komplet miar jest zapisywany do pliku `rich_results.json` w tym samym folderze. Te pliki wyjściowe stanowiły wejście dla dalszego przetwarzania z użyciem bibliotek Pandas i Matplotlib, z pomocą których sporządzono wykresy i tabele zamieszczone w niniejszej rozprawie.

Listing D.2: evaluate_coco.py

```
import argparse
import glob
import json
from operator import itemgetter
import os
import numpy as np
from pycocotools.coco import COCO
from pycocotools.cocoeval import COCOeval
from imports import backup, load

IoU_T_IDX = 0  # first IoU threshold = 0.5
MAXDET_IDX = -1  # last "maxDets"
AREARNG_IDX = 0  # 'all'

ANNOTATIONS = "datasets/coco/annotations/instances_val2017.json"
# same as:
# from detectron2.data import MetadataCatalog
# ANNOTATIONS = MetadataCatalog.get('coco_2017_val').json_file

def parse_cli() -> argparse.Namespace:
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "detection_files", nargs="+",
        help="path to coco_instances_results.json[.gz|.bz2|], or folder(s)",
    )
    parser.add_argument(
        "--min-score", "-t", type=float,
        help="confidence threshold for detections"
    )
    parser.add_argument(
        "--top", "-n", type=int, default=3,
        help="how many best and worst classes to report",
    )
    parser.add_argument(
        "--full", "-f", action="store_true", help="perform full accumulate / summarize"
    )
```

```
    return parser.parse_args()

def load_gt():
    if not os.path.exists(ANNOTATIONS):
        print("Please unzip annotations_trainval2017.zip to datasets/coco/")
        exit()
    return COCO(ANNOTATIONS)

def load_detections_results(file_or_dir):
    if os.path.isdir(file_or_dir):
        dump_dir = file_or_dir
        det_filename = glob.glob(
            os.path.join(dump_dir, "coco_instances_results.json*"))[0]
    else:
        dump_dir = os.path.dirname(file_or_dir)
        det_filename = file_or_dir

    results_file = glob.glob(os.path.join(dump_dir, "results.json*"))[0]
    results = load(results_file)
    detections = load(det_filename)
    print(f'Loaded dets by {results["model"]}, count = {len(detections):,}.')
    return detections, results, dump_dir, det_filename

args = parse_cli()
gt = load_gt()
metrics = []

for df in args.detection_files:
    print(f"Processing {df}")
    detections, results, dump_dir, det_filename = load_detections_results(df)
    if args.min_score:
        detections = [d for d in detections if d["score"] > args.min_score]
        print(f"Filtered with T={args.min_score} to {len(detections)} dets.")
    min_score = min(d["score"] for d in detections)

    dt = gt.loadRes(detections)
    coco = COCOeval(gt, dt, iouType="bbox")
    if not args.full:
        # don't evalImage for 'small', 'medium', 'large'
        coco.params.areaRng = [[0.0, 1e9]]
    coco.evaluate()
    tp = 0
    fp = 0
    n_ign = 0
    n_gt = 0
    nCats = len(coco.params.catIds)
    nArea = len(coco.params.areaRng)
    nImgs = len(coco.params.imgIds)

    assert len(coco.evalImgs) == nCats * nArea * nImgs

    print(f"nArea: {nArea}, len(evalImgs): {len(coco.evalImgs)}")
    for catIx in range(nCats):
        offs = catIx * (nArea * nImgs)
        for ix, img in enumerate(coco.evalImgs[offs : offs + nImgs]):  # noqa
            if img is None:
                continue

            ign = img["dtIgnore"][IoU_T_IDX]
            mask = ~ign
            n_ignored = ign.sum()
            n_ign += n_ignored
            tp += (img["dtMatches"][IoU_T_IDX][mask] > 0).sum()
            fp += (img["dtMatches"][IoU_T_IDX][mask] == 0).sum()
            n_gt += len(img["gtIds"]) - img["gtIgnore"].astype(int).sum()
```

```
recall = tp / n_gt
precision = tp / (tp + fp)

assert tp + fp + n_ign == len(
    detections
), f"TP/{tp}/ + FP/{fp}/ +I/{n_ign}/ == {tp+fp} != |D| /{len(detections)}/"

f1 = 2 * precision * recall / (precision + recall)

print(f"Total objects found: {tp:,} (of {n_gt:,} GT, {n_ign:,} ignored, {fp:,} FP)")
print(
    f"precision {precision*100:.1f} recall {recall*100:.1f} f1 score: {f1*100:.1f}"
)

model = results["model"].replace("_", r"\_")
ap = results["results"]["bbox"]["AP"]
ap50 = results["results"]["bbox"]["AP50"]
ap75 = results["results"]["bbox"]["AP75"]
apl = results["results"]["bbox"]["APl"]
apm = results["results"]["bbox"]["APm"]
aps = results["results"]["bbox"]["APs"]

print(
    results["model"],
    {
        k: np.round(v, 1)
        for k, v in results["results"]["bbox"].items()
        if "-" not in k
    },
)

if args.full:
    coco.accumulate()
    coco.summarize()
    raw_rc = coco.eval["recall"][IoU_T_IDX, :, AREARNG_IDX, MAXDET_IDX]
    e_recall = np.mean(raw_rc[raw_rc > -1])
    print(f"Recall by .eval: {e_recall}")
    new_results = dict(
        zip(results["results"]["bbox"].keys(), (100 * coco.stats[:6]).round(1))
    )
    print("New results by coco.stats =", new_results)
    ap, ap50, ap75, aps, apm, apl = new_results.values()

classes = sorted(
    [(v, k) for k, v in results["results"]["bbox"].items() if "-" in k],
    reverse=True,
)
print(
    f"Top {args.top} classes (orig results):\n ",
    ",\n  ".join(f"{v:4.1f} = {k}" for v, k in classes[: args.top]),
)
print(
    "Bottom {args.top} classes (orig results):\n ",
    ",\n  ".join(f"{v:4.1f} = {k}" for v, k in classes[-args.top :]),  # noqa
)

metrics.append(
    [
        model, ap, ap50, ap75, apl, apm, aps,100 * recall,
        100 * precision, tp, fp, n_ign,
    ]
)

with open(".evaluate_log.txt", "a") as log:
    print(
        f"{model:10s}: TP {tp:,} (GT {n_gt:,}, FP {fp:,}, EX {n_ign}), "
```

```
            f"PPV {precision*100:.1f} TPR {recall*100:.1f} F1 {f1*100:.1f} - {det_filename}", file=lo
        )
        if args.full:
            print("New Results: ", new_results, file=log)

    new_results_file = os.path.join(dump_dir, "rich_results.json")
    backup(new_results_file)
    bbox = {k: v for k, v in results["results"]["bbox"].items() if "-" not in k}
    rich_results = {
        "quality": results["quality"],
        "model": results["model"],
        "elapsed": results["elapsed"],
        # TypeError: Object of type int64 is not JSON serializable...
        "tp": int(tp),
        "fp": int(fp),
        "ex": int(n_ign),
        "precision": precision,
        "recall": recall,
        "f1": f1,
        "min_score": min_score,
        "score_T": args.min_score,
        **bbox,
    }
    # import code; code.interact(local=locals())
    with open(new_results_file, "w") as jsf:
        json.dump(rich_results, jsf, indent=2)

    print("-" * 79)

# Summary ("bestAP" table)
# Remember to define:
# \newcommand\tsub[1]{\textsubscript{#1}}
metrics.sort(key=itemgetter(0))
print(
    r"Model & AP & AP\tsub{50} & AP\tsub{75} & AP\tsub{l} & AP\tsub{m}"
    r" & AP\tsub{s} & TPR & PPV & TP & FP & EX \\"
)
print(r"\midrule")
for row in metrics:
    model, ap, ap50, ap75, apl, apm, aps, tpr, ppv, tp, fp, n_ign = row
    print(
        f"{model} & {ap:.1f} & {ap50:.1f} & {ap75:.1f} & {apl:.1f} & {apm:.1f}"
        f" & {aps:.1f} & {tpr:.1f} & {ppv:.1f} & {tp:,} & {fp:,} & {n_ign} \\\\"
    )
```