

POLITECHNIKA BYDGOSKA

im. Jana i Jędrzeja Śniadeckich

**WYDZIAŁ TELEKOMUNIKACJI, INFORMATYKI I
ELEKTRONIKI**



PRACA DYPLOMOWA INŻYNIERSKA

na kierunku Informatyka Stosowana

Inteligentny system rozpoznawania obiektów na jezdni

Pracę wykonała:

Michalina Matuszak

Nr albumu:

113068

Kierujący pracą:

dr inż. Sandra Śmigiel

Bydgoszcz, styczeń 2023

Politechnika Bydgoska
im. Jana i Jędrzeja Śniadeckich
Wydział Telekomunikacji, Informatyki i Elektrotechniki

KARTA PRACY DYPLOMOWEJ

Kierunek: Informatyka Stosowana

Forma studiów: stacjonarne / niestacjonarne

PRACA INŻYNIERSKA / MAGISTERSKA

NR 61/22/23/151/3P

Student: Michalina Matuszak

Nr albumu: 113068

Temat pracy (w języku polskim): Inteligentny system rozpoznawania obiektów na jezdni

Temat pracy (w języku angielskim): Intelligent road objects recognition system

Słowa kluczowe (w języku polskim): sztuczna inteligencja, sieci neuronowe, detekcja, segmentacja, rozpoznawanie obiektów na obrazie

Słowa kluczowe (w języku angielskim): artificial intelligence, neural networks, detection, segmentation, image object recognition


Zadania szczegółowe:

1. Analiza literatury w zakresie detekcji i rozpoznawania obiektów. 2. Dobór, przygotowanie i odpowiednie przetworzenie zbioru danych. 3. Zaproponowanie i implementacja modeli sztucznych sieci neuronowych. 4. Realizacja badań skuteczności sieci neuronowych w detekcji i rozpoznawaniu obiektów na jezdni. 5. Sformułowanie wniosków w kontekście przeprowadzonych badań.

Miejsca przeprowadzenia prac:

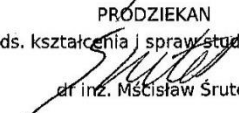
- Zakład Techniki Cyfrowej
- Wydział Inżynierii Mechanicznej

Kierujący pracą: dr inż. Sandra Śmigiel


.....
(Podpis)

Podpis studenta: Michalina Matuszak

Recenzent/Recenzenci (imię i nazwisko):

PRODZIEKAN
ds. kształcenia i spraw studenckich

dr inż. Młcisław Śrutek

Streszczenie

Przedmiotem niniejszej pracy inżynierskiej jest przedstawienie możliwości wykorzystania sieci neuronowych jako inteligentnego systemu rozpoznawania obiektów na jezdni. Celem pracy jest przetestowanie wielu modeli sieci neuronowych w zagadnieniu detekcji i segmentacji obiektów na obrazie, m. in. takich jak: samochody, znaki drogowe, piesi, rowery, sygnalizacja świetlna oraz zbadanie skuteczności ich działania w zależności od zastosowanej architektury i parametrów. W pracy przedstawiono opisy zastosowanych modeli, analizę zbioru danych i zestawienie wyników.

Słowa kluczowe: sztuczna inteligencja, sieci neuronowe, detekcja, segmentacja, rozpoznawanie obiektów na obrazie.

Summary

The subject of this thesis is to present the possibility of using neural networks as an intelligent system for recognizing road objects. The goal of this thesis is to test multiple neural network models in detection and segmentation of the road objects, e.g., such as: cars, road signs, pedestrians, bicycles, traffic lights and to test the effectiveness of their operation depending on the architecture and parameters used. The paper presents descriptions of the models used, an analysis of the dataset and a summary of obtained results.

Keywords: artificial intelligence, neural networks, detection, segmentation, image object recognition

Spis treści

1. Wstęp	5
1.1. Cel pracy	6
1.2. Zakres pracy	6
2. Podstawy teoretyczne	8
2.1. Uczenie maszynowe, a uczenie głębokie	8
2.2. Architektura sieci neuronowych.....	9
2.3. Proces uczenia sieci neuronowych	10
3. Rodzaje rozpoznawania obiektów	11
3.1. Detekcja.....	11
3.2. Segmentacja	12
4. Zbiór danych	14
4.1. Analiza zbioru danych.....	14
4.2. Format i adnotacje	17
4.2.1. Układ współrzędnych ramek w detekcji	18
4.2.2. Etykiety segmentacji.....	18
4.3. Przetwarzanie danych.....	19
5. Technologie wykorzystane w projekcie	20
5.1. Środowisko programistyczne i wymagane biblioteki	20
5.2. Parametry uczenia	22
5.3. Wybór i implementacja sieci neuronowych do detekcji	23
5.3.1. Algorytmy dwukrokowe	23
5.3.2. Algorytmy jednokrokowe	28
5.4. Zaimplementowane sieci do segmentacji.....	28
6. Ewaluacja wyników	31
6.1. Opis wybranych metryk	31
6.2. Zestawienie i prezentacja wyników	36
6.2.1. Wyniki detekcji	36
6.2.2. Wyniki segmentacji.....	43
7. Wnioski	48
8. Literatura	49
9. Spis rysunków	52
10. Spis tabel	53

1. Wstęp

Intensywny rozwój technologiczny w ostatnich latach umożliwił wdrażanie nowych rozwiązań informatycznych do wielu dziedzin przemysłowych. Jednym z takich rozwiązań jest zastosowanie sztucznych sieci neuronowych do konkretnych zadań praktycznych, np. wizji komputerowej. Sieci neuronowe to zaimplementowane modele matematyczne, których działanie jest wzorowane na biologicznej sieci nerwowej mózgu. Takie modele traktowane są obecnie jako podzbiór większej dziedziny technologicznej - sztucznej inteligencji. Powodem tego jest zdolność do uczenia się, zapamiętywania i rozpoznawania charakterystycznych własności obiektów. Do procesu trenowania sieci neuronowych wykorzystuje się zbiory dużej ilości danych, z których model pozyskuje informacje. Obecnie istnieje wiele możliwości pozyskiwania wysokiej jakości danych, które mogą być materiałem treningowym dla sztucznych sieci neuronowych. Dla przykładu, dane pozyskiwane z różnego rodzaju urządzeń cyfrowych, w tym sensorów i kamer samochodowych, mogą być wykorzystane do wytrenowania modelu sieci, która będzie w stanie rozpoznawać otoczenie na jezdni. Tak wytrenowany model można nazwać inteligentnym systemem rozpoznawania obiektów na jezdni, ponieważ będzie on potrafił rozpoznać poszczególne obiekty drogowe, a także zrozumieć ich cechy charakterystyczne.

Inteligentne systemy rozpoznawania obiektów na jezdni są ważnym aspektem w rozwoju technologii i przemysłu. Zastosowanie takich systemów może przyczynić się do zwiększenia bezpieczeństwa na drodze, dzięki rozpoznawaniu takich elementów jak: samochody, piesi, rowerzyści. W ten sposób inteligentne systemy mogą ostrzegać kierowców o potencjalnym zagrożeniu i pomóc im uniknąć wypadku. Kolejnym możliwym zastosowaniem może być poprawa płynności ruchu drogowego, dzięki wdrożeniu inteligentnych systemów rozpoznawania w radarach drogowych, badających pomiar natężenia ruchu. Dodatkowo, z użyciem takich systemów możliwa jest poprawa w monitorowaniu warunków drogowych, np. wykrywaniu ubytków w asfalcie jezdni. Zdobyte informacje mogą być automatycznie sklasyfikowane przez sieci neuronowe i pomóc w podejmowaniu decyzji dotyczących konserwacji i napraw dróg, a tym samym obniżyć koszty związane z utrzymaniem infrastruktury. W przyszłości możliwe jest także wykorzystanie takich systemów w pojazdach autonomicznych. Aby takie pojazdy były bezpieczne, wymagane są dalsze badania w tym temacie. Podsumowując, inteligentne systemy rozpoznawania obiektów drogowych są cennym narzędziem poprawy bezpieczeństwa, wydajności oraz opłacalności w branży transportowej i przemysłowej.

Zważając na omówiony potencjał wdrażania sztucznej inteligencji do branży przemysłowej, w niniejszej pracy postanowiono dokładniej zbadać ich przydatność. Praca skupia się na stworzeniu kompletnego systemu rozpoznawania obiektów na jezdni, aby przyczynić się do lepszego zrozumienia możliwości i ograniczeń sieci neuronowych w drogowej wizji komputerowej. Uzyskane rezultaty pomogą w realnej ocenie efektywności działania sieci neuronowych w owym systemie i zdecydować, jaka architektura modelu sprawdza się najlepiej w omawianym zadaniu.

1.1. Cel pracy

Celem niniejszej pracy dyplomowej jest przedstawienie możliwości wykorzystania sztucznej inteligencji w rozpoznawaniu wielu obiektów na jezdni, m. in. takich jak: samochody, znaki drogowe, piesi, rowery, sygnalizacja świetlna. Dokładniej, celem jest utworzenie kompletnego systemu, czyli implementacja odpowiednich algorytmów sieci neuronowych i badanie skuteczności ich działania. Praca wymaga kilku konkretnych etapów docelowych. Pierwszym z nich jest wyjaśnienie działania sieci neuronowych oraz wprowadzenie teoretyczne, tłumaczące różnicę w sposobach wykrywania obiektów. Dalszym zadaniem jest przedstawienie i analiza wykorzystanego w projekcie zbioru danych. Wybrany zbiór powinien przedstawiać adekwatne scenerie drogowe. W tym wypadku są to głównie miasta, drogi, tunele i autostrady. Kolejnym zadaniem jest implementacja i przetestowanie wielu modeli sieci neuronowych wykorzystujących proces detekcji lub segmentacji obiektów. Następnym ważnym elementem docelowym jest zbadanie skuteczności działania sieci neuronowych, w zależności od zastosowanej architektury i parametrów. W tym celu niezbędne jest wyznaczenie i opisanie przystosowanych do tego metryk ewaluacji wyników oraz analiza otrzymanych w pracy rezultatów. Taki schemat pracy pozwoli ocenić efektywność działania modeli sztucznych sieci neuronowych w zadaniu rozpoznawania obiektów na jezdni.

1.2. Zakres pracy

Praca obejmuje omówienie podstaw teoretycznych, niezbędnych do zrozumienia rozpatrywanego tematu. W tym celu wyjaśnione zostają fundamentalne zasady sztucznych działania sieci neuronowych, a także procesu trenowania. Wstęp teoretyczny zamieszczony został w rozdziale 2. Rozpoznawanie obiektów podzielone zostało w pracy na dwa główne sposoby: detekcja i segmentacja. W rozdziale 3. zamieszczony został opis działania każdego z zastosowanych sposobów, a także różnica w budowie przystosowanych do tego algorytmów, tj. jednokrokowe i dwukrokowe. Praca obejmuje kompleksową analizę zbioru danych

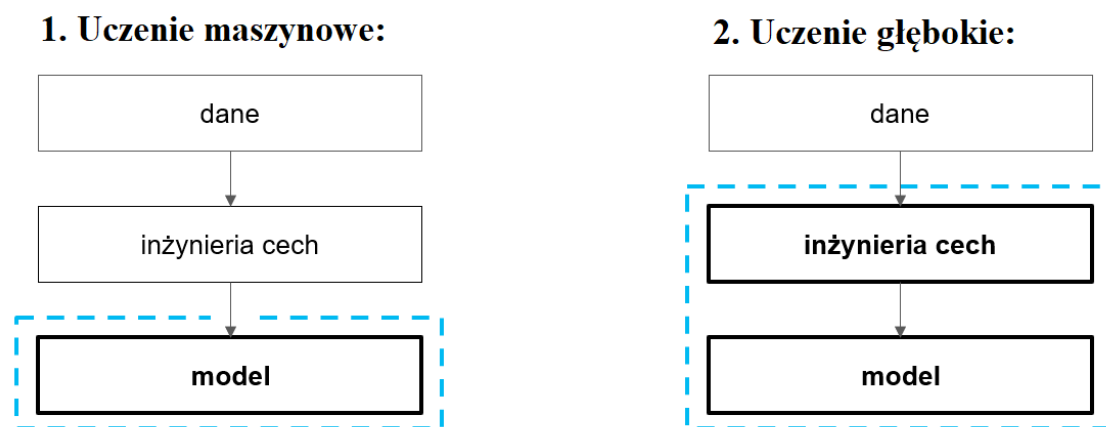
„BDD100K”, użytego do trenowania modeli sieci neuronowych. Dodatkowo uwzględnione zostało omówienie statystyki liczebności klas w zbiorze. Przygotowano prezentację wykorzystywanych formatów etykiet do detekcji i segmentacji. Wszystkie informacje o użytych w projekcie danych przedstawiono w rozdziale 4. Przetestowane zostały poszczególne konfiguracje modeli, wyjaśniono architekturę i schemat działania wykorzystywanych w pracy algorytmów. Do detekcji wykorzystane zostały poszczególne konfiguracje sieci z rodziny R-CNN (Faster R-CNN, Cascade R-CNN) oraz sieć FCOS. Do segmentacji wykorzystano Mask R-CNN oraz Cascade Mask R-CNN. Opis konfiguracji i architektury algorytmów znajduje się w rozdziale 5. Ostatnim etapem jest uwzględnienie adekwatnych metryk do pomiaru efektywności zastosowanych modeli, zestawienie wyników i podsumowanie otrzymanych rezultatów. Opis wyników i wniosków znajduje się odpowiednio w rozdziałach 6 i 7.

2. Podstawy teoretyczne

Sieci neuronowe w dziedzinie sztucznej inteligencji są implementacją skomplikowanych modeli matematycznych, które budową przypominają biologiczną sieć neuronową mózgu. Obecnie sieci neuronowe wykorzystywane są w wielu zadaniach, takich jak klasyfikacja obiektów, rozpoznawanie mowy, generowanie obrazów, wykrywanie fake-news lub prognozowanie pogody. W tym rozdziale opisane są podstawy teoretyczne sztucznych sieci neuronowych.

2.1. Uczenie maszynowe, a uczenie głębokie

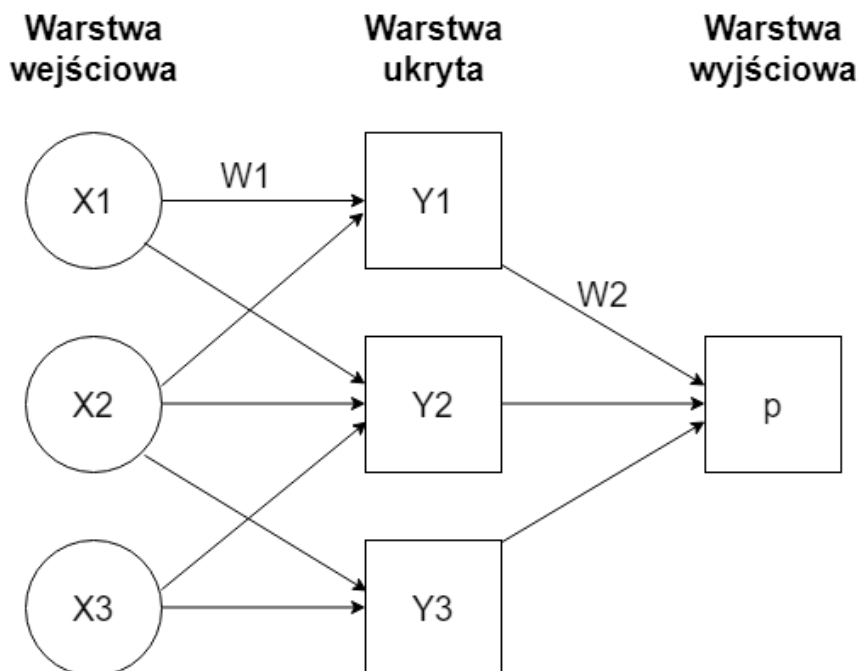
Pierwowzorem sztucznej inteligencji jest uczenie maszynowe, czyli wykorzystanie trenowanych modeli matematycznych do obliczeń. Znaczącą różnicą między klasycznym uczeniem maszynowym, a uczeniem głębokim (do którego można przypisać obecną sztuczną inteligencję) jest sposób wykorzystania informacji z danych. Modele sieci neuronowych umożliwiają aktywne wykorzystanie inżynierii cech. Inżynieria cech w głębokim uczeniu jest procesem możliwym do wytrenowania, co umożliwia modelowi nauczenie się i zapamiętywanie najważniejszych informacji oraz wykorzystanie zdobytej wiedzy w innych zadaniach praktycznych.



Rys. 1. Schemat przedstawiający różnicę między uczeniem maszynowym, a uczeniem głębokim. Proces możliwy do wytrenowania oznaczony jest przerywaną linią

2.2. Architektura sieci neuronowych

Podstawową jednostką przechowującą informacje w sieciach są neurony. Neurony połączone są ze sobą w odpowiedni sposób, co umożliwia konkretny przepływ informacji. Sieć neuronowa podzielona jest na warstwy, w których znajdują się określone liczby neuronów. Warstwa początkowa służy za odbieranie informacji, które następnie zostają przesłane do kolejnych poziomów sieci, tzw. warstw ukrytych. W tym miejscu następują odpowiednie obliczenia, które służą do wykrywania charakterystycznych cech, a tym samym zastosowania takiej umiejętności w różnych zadaniach praktycznych.

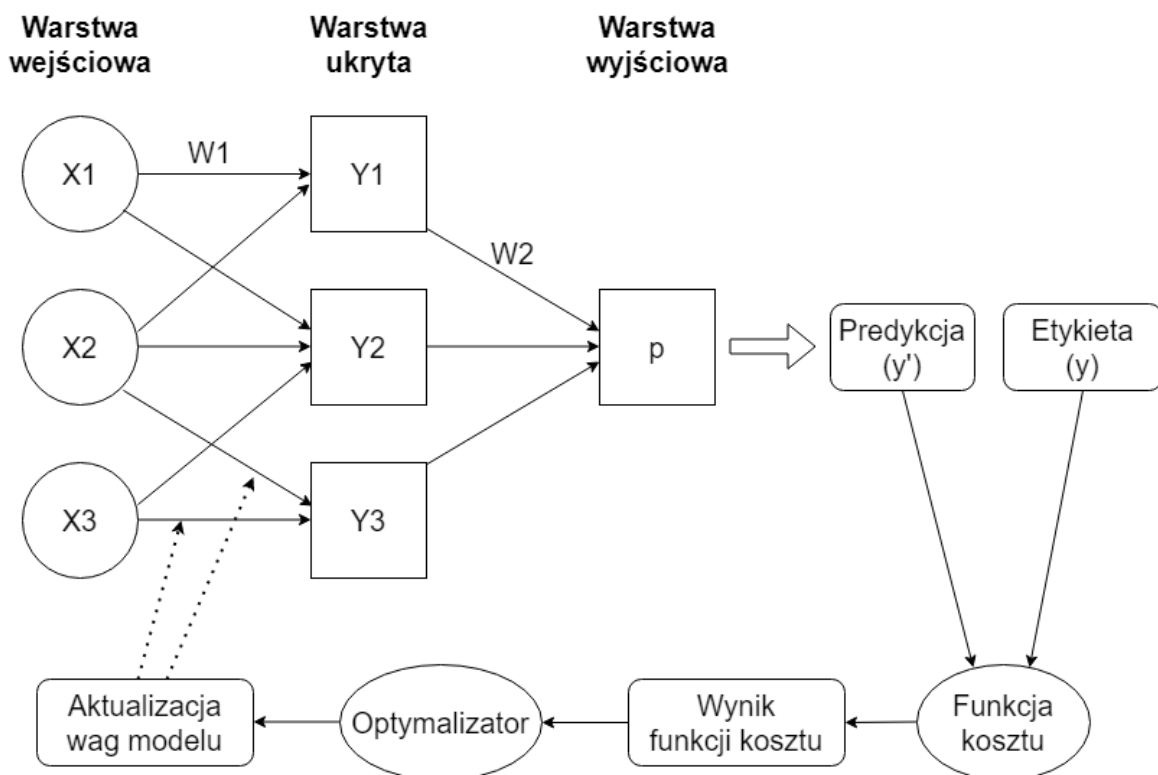


Rys. 2. Schemat przedstawiający architekturę sieci neuronowej

Na powyższym wykresie przedstawiono architekturę działania sztucznej sieci neuronowej. \mathbf{X} to oznaczenia neuronów wejściowych. \mathbf{Y} to neurony w warstwie ukrytej, które posiadają funkcję aktywacji. Odpowiada ona za określenie jak należy przetworzyć informacje otrzymywane od innych neuronów. Oznaczona na wyjściu klasyfikacja \mathbf{p} jest neuronem produkującym wynik. \mathbf{W} to wagi neuronów, które odpowiadają za nadanie konkretnym połączeniom pomiędzy neuronami większego lub mniejszego znaczenia.

2.3. Proces uczenia sieci neuronowych

Proces trenowania sieci neuronowych podzielony jest na wiele etapów. Początkowo wagi modelu są losowe i w kolejnych iteracjach treningowych są aktualizowane. Działą to poprzez porównanie wyprodukowanych przez model predykcji (wartości wyjściowych) do etykiet ze zbioru danych (wartości prawdziwych). Na podstawie różnicy tych wartości obliczana jest wartość określonej funkcji kosztu. Następnym etapem jest aktualizacja wag. Za ten etap odpowiada optymalizator, który stosuje wsteczną propagację gradientów i za pomocą odpowiedniej funkcji aktualizuje wagi modeli. Dzięki takiemu procesowi wagi modelu są skutecznie dostosowywane, tak aby regulować znaczenie wykrywanych cech, a tym samym zmniejszać funkcję kosztu poprzez pokazanie algorytmowi, które predykcje są bliższe prawdzie. Schemat procesu trenowania sieci neuronowych został zaprezentowany na poniższej grafice:



Rys. 3. Schemat przedstawiający proces trenowania sieci neuronowej

Po wytrenowaniu modelu na odpowiednim zbiorze treningowym należy zmierzyć jego efektywność. Stosuje się do tego zbiór testowy zawierający nowe dane, których model nie widział podczas treningu. Pozwala to ocenić jakość modelu i jego umiejętność wykonywania określonego zadania w praktyce. Do ewaluacji wyników stosuje się odpowiednie miary, które zostały przystosowane do pomiaru konkretnego zadania.

3. Rodzaje rozpoznawania obiektów

W sztucznej inteligencji istnieją głównie dwie metody wykrywania obiektów: detekcja oraz segmentacja. Detekcja obiektów to proces rozpoznawania i lokalizowania pozycji każdego obiektu na obrazie, a także przypisanie każdemu z nich odpowiedniej klasy i oznaczenie go w ramce. Natomiast celem segmentacji jest przypisanie do klasy każdego piksela na obrazie. Modele do wykrywania obiektów składają się z kilku głównych składników:

1. Warstwy spłotowe są odpowiedzialne za wykrywanie cech obiektów na obrazie.
2. Warstwy regresji są odpowiedzialne za wykrywanie pozycji obiektu na obrazie.
3. Warstwy klasyfikacji są odpowiedzialne za rozpoznanie klasy obiektu.
4. Dodatkowo w przypadku segmentacji używane są warstwy segmentacji. Takie warstwy generują maski, czyli oznaczenia obszaru i konturu obiektu, w regionie proponowanym przez warstwę regresji.

W tym rozdziale opisane są poszczególne metody detekcji i segmentacji zastosowane w niniejszym projekcie oraz schemat działania zaimplementowanych do tego modeli sieci neuronowych

3.1. Detekcja

Analizując publikację [1] badającą różne rodzaje inteligentnych detektorów można wywnioskować, że współczesne algorytmy do detekcji dzielą się głównie na dwa typy: detektory dwukrokowe (ang. two-stage detectors) oraz detektory jednokrokowe (ang. single-stage detectors).

Działanie detektora dwukrokowego polega na dwustopniowej pracy: etap propozycji regionu wydziela ustaloną ilość ramek, na których obiekt potencjalnie może się znajdować. Następnie na etapie klasyfikacji i lokalizacji sieć ocenia, czy w danej ramce znajduje się obiekt rozpoznanej klasy. Algorytmy dwukrokowe zostały opracowane w celach dokładniejszej detekcji. W porównaniu z detektorami jednokrokowymi osiągają precyzyjniejsze wyniki, ale jednocześnie ich szybkość obliczania jest mniejsza, ponieważ sieć musi sklasyfikować wszystkie zaproponowane wcześniej ramki.

Detektory jednokrokowe wykrywają obiekty bezpośrednio z wejściowego pola obrazu, bez etapu proponowania regionów ramek. Zamiast tego sieć dzieli obraz na siatkę i oblicza możliwość i prawdopodobieństwo wystąpienia rozpoznanej klasy na obrazie, tym samym osiągając wysoką szybkość działania. Z tego względu, sieci używające detektorów jednokrokowych dobrze nadają się do użytku w czasie rzeczywistym [1].

3.2. Segmentacja

Segmentacja jest kolejną techniką wykrywania obiektów, stosowaną za pomocą sztucznej inteligencji w dziedzinie przetwarzania obrazów cyfrowych. W przypadku użycia detektora obiektów w samochodzie autonomicznym istnieje ryzyko, że ramki wielu obiektów mogą na siebie nachodzić i jednocześnie powodować nieścisłości w interpretacji takiej sytuacji przez pojazd. Aby temu zapobiec wynaleziono dokładniejszy sposób wykrywania obiektów, czyli segmentację. Taki proces polega na obliczeniu i przypisaniu odpowiedniej klasy lub kategorii dla każdego piksela na obrazie, czego skutkiem jest wyznaczenie dokładnego obszaru i kształtu obiektu, a także jego konturu. Wykorzystywane są algorytmy propozycji regionów i sieci splotowe (CNN) do klasyfikacji wszystkich pikseli na obrazie. Dane wejściowe, podobnie jak w przypadku detekcji, posiadają oznaczenia obszaru, na których znajduje się obiekt danej klasy. Różnicą jest natomiast szczególne uwzględnienie granicy kształtu danego obiektu. Formatem oznaczeń stosowanym w segmentacji zazwyczaj są maski bitowe, gdzie etykiety dla każdego obrazu są przechowywane w pliku RGBA. Format został wyjaśniony w rozdziale 4.2.2.

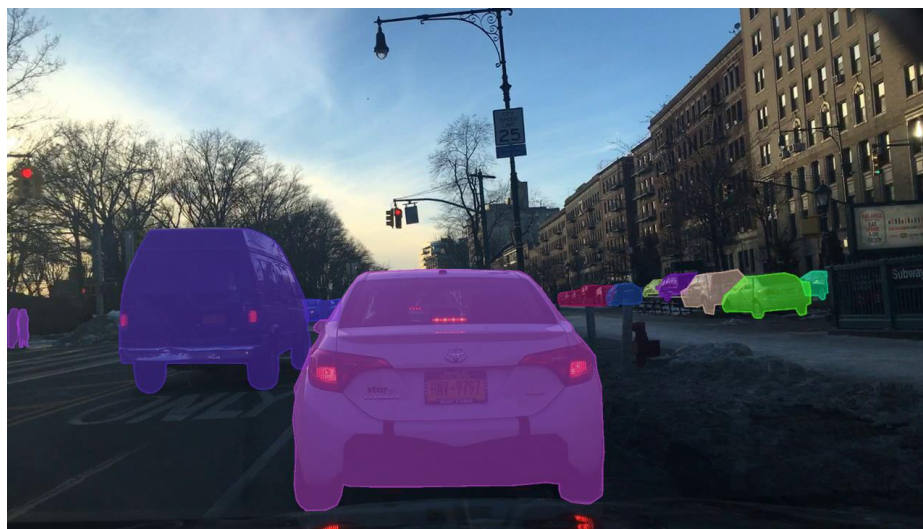
Warto wyszczególnić 3 główne rodzaje segmentacji stosowane w sztucznej inteligencji:

1. Segmentacja semantyczna (ang. Semantic segmentation) polega na sklasyfikowaniu każdego piksela obrazu do znanej klasy lub kategorii. Do klasyfikowania obiektów w predefiniowane klasy używa się zwykle sieci splotowych (CNN). Przetwarzane jest całe pole obrazu wejściowego, a następnie do każdego sklasyfikowanego piksela przypisana etykieta. Obiekty tej samej klasy otrzymują tę samą etykietę. Celem segmentacji semantycznej jest wydzielenie kategorii obiektów. Poniżej przedstawiono przykład segmentacji semantycznej (Rys. 4):



Rys. 4. Przykład działania segmentacji semantycznej [2]

2. Segmentacja instancji (ang. Instance segmentation) również polega na sklasyfikowaniu każdego piksela obrazu, jednakże zasadniczą różnicą od segmentacji semantycznej jest rozdzielenie każdego obiektu na osobne instancje danej klasy. W przypadku segmentacji instancji, obiekty tej samej klasy, otrzymają osobną etykietę. Dla przykładu: każdy samochód na zdjęciu otrzyma unikalną etykietę, rozróżniającą poszczególne auto, od innych. Pozwala to na identyfikację i lokalizację poszczególnych instancji klas obiektów na obrazie. Przykład działania segmentacji instancji został przedstawiony na poniższej grafice (Rys. 5):



Rys. 5. Przykład działania segmentacji instancji [2]

3. Segmentacja panoptyczna (ang. Panoptic segmentation) łączy zadania segmentacji instancji i segmentacji semantycznej. Obejmuje zarówno identyfikację i klasyfikację poszczególnych obiektów na obrazie, jak i klasyfikację pikseli w każdym obiekcie do predefiniowanych klas lub kategorii. Pozwala to na pełniejsze zrozumienie zawartości obrazu, ponieważ można zidentyfikować zarówno poszczególne obiekty oraz ich ogólną kategorię, a tym samym zrozumieć ich wzajemną relację między sobą.

4. Zbiór danych

Zadanie graficznego rozpoznawania otoczenia i wykrywania obiektów na jezdni, realizowane na potrzeby robotów i samochodów autonomicznych charakteryzuje się małą tolerancją na błędy, ponieważ potrzebne jest przestrzeganie wymogów bezpieczeństwa na drodze. Z uwagi na to, bardzo ważną kwestią jest wybranie do takiego zagadnienia jakościowego zbioru danych, oferującego dużą ilość obrazów wysokiej rozdzielczości, dokładnych adnotacji obiektów oraz zróżnicowania pogodowego.

Źródłem danych do wytrenowania i przetestowania użytych modeli w niniejszym projekcie jest zbiór danych obrazowych oraz wideo [2, 3] „The Berkeley DeepDrive Dataset (BDD100K)” udostępniony przez The University of California w 2018 roku. Zgodnie z załączoną licencją, dane można wykorzystywać bez opłat w celach edukacyjnych i badawczych. Zbiór ten przedstawia widoki jezdni, uzyskane z kamer samochodowych. Oferuje wysoką różnorodność scenerii, a także obrazy zawierające zróżnicowane warunki pogodowe o różnych porach dnia. Zgodnie z teorią, dzięki różnorodności zbioru danych, wytrenowany model uzyska większy stopień generalizacji, a jego predykcje obiektów są dokładniejsze i mniej uzależnione od konkretnego czynnika pogodowego.

Elementem świadczącym o wielozadaniowości owego zbioru jest dostępność adnotacji do wielu zadań wizji komputerowej w sztucznej inteligencji. Poza detekcją obiektów, zbiór oferuje adnotacje do wielu zadań, m. in. wykrywanie linii jezdni, segmentacja semantyczna, segmentacja panoptyczna oraz śledzenie obiektów w ruchu, dzięki adnotacjom załączonym do plików wideo [3].

Rozdział ten poświęcony jest analizie danych zbioru „BDD100K”, użytego do trenowania i ewaluacji wyników sieci neuronowych zaimplementowanych w niniejszej pracy.

4.1. Analiza zbioru danych

Po dokonaniu dokładnej, eksploracyjnej analizy danych wyszczególnione zostają znaczące cechy i statystyki zbioru. Zbadanie ich zapewni lepsze zapoznanie się z charakterystyką zbioru danych, co jest kluczowym elementem dobrego zrozumienia problemu i zaplanowania dalszych prac.

Zbiór danych BDD100k oferuje 100 tysięcy zdjęć scenerii jezdni do zadań detekcji, pochodzących od 50 tysięcy osobnych podróży samochodowych. Wielkość zbioru danych to 5.3 GB. Część z danych została oznaczona również do zadań segmentacji (semantyczna, instancyjna, panoptyczna), a ich ilość wynosi 10 tysięcy zdjęć. Każdy z dostępnych folderów

został podzielony na zbiór treningowy, walidacyjny oraz testowy, wg. następującego podziału procentowego:

Treningowy 70%

Walidacyjny 10%

Testowy 20%.

Do zadań detekcji zbiorów danych wyszczególnia 13 klas, natomiast do zadań segmentacji istnieją oznaczone 23 klasy obiektów. W niniejszym projekcie wygenerowano odpowiednie statystyki do analizy za pomocą biblioteki „fifty.one”.

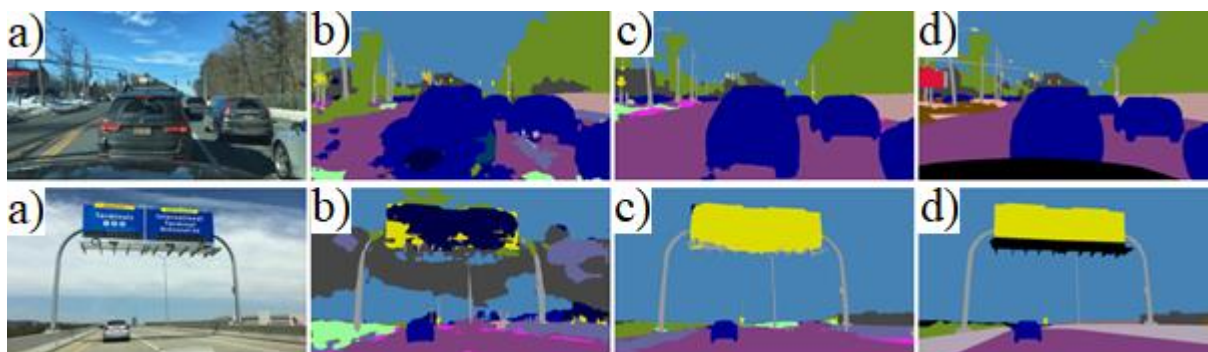
Tabela 1. Dystrybucja klas w zbiorze do detekcji obiektów

Lp.	Nazwa instancji	Ilość wystąpień w zbiorze	
		Treningowy	Walidacyjny
1	car	700703	102837
2	traffic sign	238270	34724
3	traffic light	187871	26884
4	pedestrian	92159	13425
5	truck	27892	4243
6	bus	11977	1660
7	bicycle	7124	1039
8	rider	4560	658
9	motorcycle	3023	460
11	other vehicle	804	85
12	train	128	15
13	trailer	71	2

Powołując się na oficjalną publikację [4] autorów zbioru BDD100K, każda adnotacja danej klasy zawarta jest w konkretnej kategorii, przypisanej dla każdej klatki referencyjnej wszystkich 100 tysięcy filmów dostępnych w zbiorze. Dodatkowo autorzy zapewnili dodatkowe atrybuty określające widoczność danego obiektu: „zasłonięty” i „obcięty”.

Zbiór BDD100K oferuje wysoką różnorodność typów scenerii, w tym ulice miast, obszary mieszkalne i autostrady. Kontynuując sprawdzenie wniosków zawartych we wcześniej wspomnianej publikacji, można spotkać stanowisko, w myśl którego model przykładowej sieci DRN [5], wytrenowany w celach porównawczych na osobnych zbiorach danych: “BDD100K” oraz “Cityscapes” [6], osiąga lepsze wyniki na pierwszym z nich. Autorzy uzasadniają tę

różnicę zwiększoną liczbą krajobrazów, scenerii i typów umiejscowienia drogi na obrazach w zbiorze. Omawiane wyniki zostały przedstawione w następujący sposób (Rys. 6):



Rys. 6. a) zdjęcie referencyjne, b) trenowane na „Cityscapes”, c) trenowane na „BDD100k”, d) adnotacja [4]

Rozszerzając wnioski badawcze autorów publikacji, w niniejszej pracy została utworzona dodatkowa analiza statystyczna dystrybucji większości atrybutów i klas dostępnych w zbiorze „BDD100K”, której przedstawienie pozwoli na dokładne określenie możliwości i ograniczeń dalszego trenowania sieci neuronowych na tym zbiorze. Pierwszym parametrem poddanym analizie jest parametr „scena”, określający podział scenerii lub krajobrazu, wedle którego zostały porównane omawiane wcześniej wyniki (Rys. 6):

Tabela 2. Dystrybucja scenerii w zbiorze do detekcji obiektów

Lp.	Nazwa sceny	Ilość wystąpień w zbiorze	
		Treningowy	Walidacyjny
1	city street	43516	6112
2	highway	17379	2499
3	residential	8074	1253
4	parking lot	377	49
5	undefined	361	53
6	tunnel	129	27
7	gas station	27	7

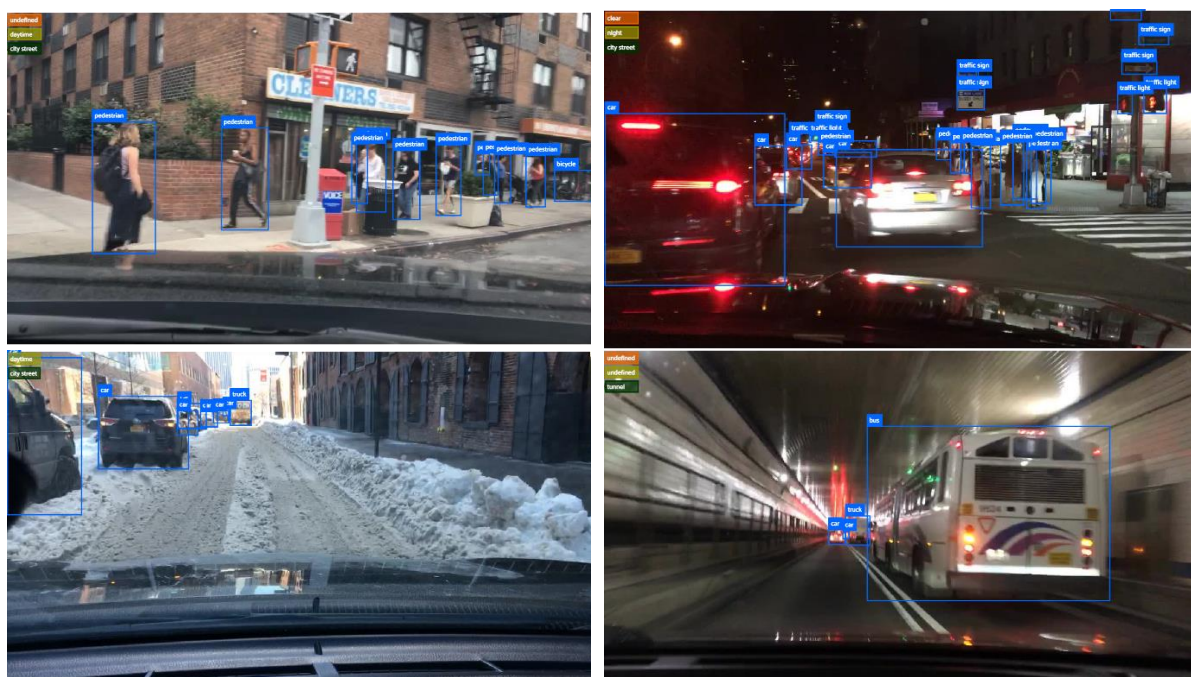
Przeprowadzając dalszą analizę statystyczną pod kątem udostępnionych atrybutów w zbiorze, wyszczególnia się grupy danych określonych konkretną porą dnia. Jest to istotna cecha dla trenowanych sieci neuronowych, ponieważ różnice w widoczności i poziomie jasności zdjęcia, powodują ogromne zmiany w sposobie postrzegania obiektów przez modele sieci. Dzięki takiej różnorodności w zbiorze, wytrenowany model potrafi lepiej radzić sobie ze zmienną wartością jasności kolorów tego samego obiektu, nawet przy nagłej zmianie, np. w

wideo przedstawiającym wjazd w słabo oświetloną drogę lub tunel. W zbiorze wyszczególnia się następujące statystyki pory dnia, przedstawione w Tabeli 3:

Tabela 3. Dystrybucja atrybutów pory dnia w zbiorze do detekcji obiektów

Lp.	Nazwa atrybutu	Ilość wystąpień w zbiorze	
		Treningowy	Walidacyjny
1	daytime	36728	5258
2	night	27971	3929
3	dawn/dusk	5027	778
4	undefined	137	35

Poniżej (Rys. 7.) zamieszczone zostały przykładowe obrazy z załączonym oznaczeniem ramki, nazwą wykrytej klasy oraz przypisanymi atrybutami:



Rys. 7. Przykładowe zdjęcia zbioru BDD100K oznaczonymi adnotacjami

4.2. Format i adnotacje

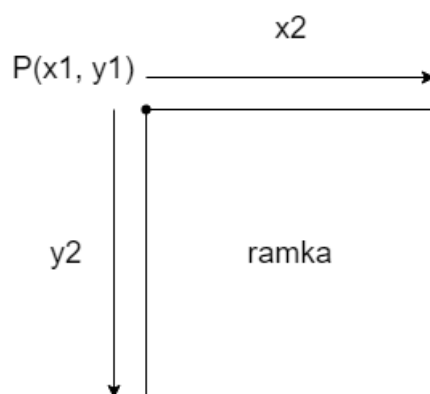
Udostępnione dane w zbiorze BDD100K oryginalnie znajdują się w formacie zgodnym z etykietami generowanymi przez pakiet Scalabel [7]. Jest to określony w dokumentacji [8] plik JSON, zawierający definicje klas i atrybutów oraz listę obiektów z współrzędnymi oznaczeń ramki (tzw. „bounding box”).

Poza opisanymi we wcześniejszym rozdziale atrybutami ramek, istnieją także atrybuty oznaczeń, przydatne przy pozostałych zadaniach wizji komputerowej z użyciem sztucznej

inteligencji. Dla przykładu: kolor sygnalizacji świetlnej, perspektywa linii jezdni, kierunek linii jezdni, rodzaj linii (ciągła, przerywana).

4.2.1. Układ współrzędnych ramek w detekcji

Zgodnie z dokumentacją [8] sposób etykietowania ramek rozpoczyna się od lewego górnego rogu. Według przedstawionego schematu (Rys. 8.), piksel oznaczony współrzędnymi $(x1, y1)$ rozpoczyna definicję położenia obiektu na zdjęciu. W udostępnionych w zbiorze skryptach, sposób definiowania długości i szerokości obiektu polega na dodaniu odpowiedniej wartości liczbowej do piksela początkowego P.



Rys. 8. Sposób definiowania współrzędnych w zbiorze BDD100K [8]

4.2.2. Etykiety segmentacji

Etykiety do segmentacji instancji i segmentacji panoptycznej są zapisywane jako maski bitowe, gdzie etykiety dla każdego obrazu są przechowywane w pliku RGBA. Maski segmentacji nie mogą na siebie nachodzić, ponieważ każdy piksel powinien mieć przypisaną tylko jedną klasę. Według dokumentacji zbioru BDD100K [8], poza położeniem obiektu, dodatkowo kodowane informacje na obrazie RGBA są następujące:

- Kanał R – pierwszy bajt, używany jako identyfikator kategorii. Mieści się w zakresie [0,1] przy czym: 0 = tło, 1 = pozostałe.
- Kanał G dotyczy atrybutów instancji. Obecnie używane są cztery atrybuty: „obcięte”, „przesłonięte”, „tłum” i „ignoruj”. Pola z etykietami „tłum” lub „ignoruj” nie będą brane pod uwagę podczas ewaluacji. Powyższe cztery atrybuty są przechowywane w najmniej znaczących bitach G. Można to przedstawić poniższym wzorem (1):

$$G = (\text{obcięte} \ll 3) + (\text{przesłonięte} \ll 2) + (\text{tłum} \ll 1) + \text{"ignoruj"} \quad (1)$$

- Kanał B przechowuje „ann_id”, czyli identyfikator danej adnotacji.
- Kanał A jest dodatkowy i służy do śledzenia segmentacji na wideo, można go obliczyć jako:

$$(B \ll 8) + A \quad (2)$$

4.3. Przetwarzanie danych

Wstępne przetwarzanie danych (preprocessing) oznacza proces przygotowania obrazów wejściowych do dalszego użycia w modelach uczenia maszynowego. Preprocessing danych graficznych jest ważną częścią projektu, ponieważ jakość danych wejściowych ma bezpośredni wpływ na wynikową jakość modelu. Dzięki takiemu przetworzeniu danych możliwe jest zmodyfikowanie pewnych cech na obrazie, a tym samym zoptymalizowanie procesu uczenia się modelu i zwiększenie jego wydajności. Elementy przetwarzania danych zastosowanych w projekcie to:

1. Ujednolicenie wymiarów wszystkich obrazów - wszystkie dane treningowe zostały przeskalowane do rozmiaru domyślnego 780x1280 pikseli, tak aby uniknąć niespójności.

2. Normalizacja kolorów - technika używana do dostosowywania właściwości kolorów obrazu, tak aby był spójny w różnych warunkach oświetleniowych lub środowiskach. Wszystkie kanały RGB z wartościami z zakresu [0,255] zostają zapisane do tensora, a wartość kolorów zostaje przekształcona do zakresu [0.0, 1.0]. Wartości kolorów są poddane następującemu przekształceniu:

$$wyjście(R, G, B) = \frac{(wejście(R, G, B) - średnia(R, G, B))}{odchylenie_standardowe(R, G, B)} \quad (3)$$

Normalizacja wszystkich kanałów RGB jest znaczącym elementem w przygotowywaniu danych graficznych, ponieważ pomaga upewnić się, że model nie jest stronniczy względem zapamiętywania określonych kolorów lub typów oświetlenia. Bez normalizacji model może działać słabo na obrazach, które mają inną kolorystykę, niż dane uczące. Ponadto normalizacja może pomóc poprawić umiejętność generalizacji modelu, czyli w tym przypadku zdolności wykrywania obiektów na zróżnicowanych typach scenarii. Oznacza to, że model będzie w stanie lepiej rozpoznawać obiekty na nowych obrazach, których wcześniej nie widział.

5. Technologie wykorzystane w projekcie

Pracując na wielkich zbiorach danych dobrym wyborem jest użycie języków skryptowych, takich jak Python, który obecnie oferuje szerokie możliwości programistyczne w dziedzinie uczenia maszynowego. Obecnie istnieje wiele darmowo dostępnych narzędzi przystosowanych do pracy z dużą ilością danych oraz specjalistycznych bibliotek do programowania sztucznej inteligencji.

Ten rozdział opisuje zastosowane modele sieci neuronowych oraz środowisko programistyczne, na którym zostały zaimplementowane i używane.

5.1. Środowisko programistyczne i wymagane biblioteki

Język programowania:

Głównym językiem programowania służącym do utworzenia i obsługi odpowiednich skryptów jest Python [9]. Jest to język wysokiego poziomu, posiadający szeroki zasób bibliotek pomocnych do pracy z uczeniem maszynowym i sieciami neuronowymi oraz zarządzaniem dużymi zbiorami danych. Python to projekt typu Open Source, posiadaczem licencji jest Python Software Foundation. Język ten jest wieloplatformowy, jego wysoka integralność z systemem operacyjnym Windows, jak i wieloma dystrybucjami systemu Linux zapewnia dużą elastyczność.

Środowisko wirtualne:

Idealnym narzędziem do zarządzania pakietami i wydzieleniem środowiska wirtualnego jest Conda [10]. Jest to program typu Open Source, który tworzy oddzielną, wirtualną przestrzeń programistyczną. Taki zabieg pozwala uniknąć utworzenia się błędnych zależności pomiędzy instalowanymi pakietami i bibliotekami, ponieważ w środowisku wirtualnym instaluje się biblioteki potrzebne jedynie do danego projektu.

Platforma:

Google Colab [11] autorstwa firmy Google Research, jest platformą przeglądarkową umożliwiającą hostowanie interaktywnych notatników formatu Jupyter Notebook. Szczególnie nadaje się do uczenia maszynowego i analizy danych. Projekt ten został stworzony jako gotowa platforma niewymagająca dodatkowych konfiguracji oraz zapewniająca darmowy, ograniczony dostęp do zasobów obliczeniowych.

Biblioteki:

CUDA [12] – architektura obliczeniowa firmy NVIDIA, która pozwala na wykonywanie obliczeń na kartach graficznych (GPU) zgodnych z CUDA. CUDA umożliwia programistom napisanie kodu przetwarzalnego przez GPU, co pozwala na zwiększenie wydajności obliczeniowej w porównaniu z wykorzystaniem procesorów centralnych (CPU). Jest to popularna technologia w dziedzinie uczenia maszynowego, gdzie obliczenia na GPU pozwalają na przyspieszenie treningu sieci neuronowych. CUDA jest również używana w wielu innych dziedzinach, takich jak wizualizacja, rozszerzona rzeczywistość, nauki i inżynieria.

PyTorch [13] - biblioteka sztucznej inteligencji stworzona przez Facebook'a, która umożliwia tworzenie modeli uczenia maszynowego za pomocą dynamicznych grafów obliczeniowych. PyTorch pozwala na łatwą implementację algorytmów uczenia głębokiego, takich jak sieci neuronowe i umożliwia łatwe pisanie kodu. Jest kompatybilna z rozszerzeniem CUDA wspierającym użycie GPU. Jest to popularna biblioteka wśród naukowców i inżynierów pracujących w dziedzinie uczenia maszynowego, która jest często używana do rozwiązywania problemów związanych z obrazem, językiem i danymi.

MMDetection [14] – zestaw narzędzi do wykrywania obiektów z użyciem sieci neuronowych. MMDetection to rozbudowany pakiet typu Open Source oparty na bibliotece PyTorch. Jest częścią projektu OpenMMLab firmy Apache, który wygrał międzynarodowe wyzwanie wykrywania obiektów przez sieci neuronowe „COCO Detection Challenge” z 2018 roku. Pozwala na zarządzanie gotowymi strukturami kodu do wykrywania obiektów oraz niestandardową ich edycję, tworząc lub łącząc różne moduły. Pakiet został użyty w tejże pracy z powodu wielu narzędzi umożliwiających projektowanie, trening i ewaluację modeli detekcji obiektów na różnych zestawach danych, a także wysoką wydajność, ze względu na kompatybilność z pakietem CUDA od NVIDIA. MMDetection współdziela również z biblioteką Pycocotools. Jest to biblioteka używana do pracy na etykietach w formacie COCO używanych powszechnie w detekcji. W niniejszym projekcie oryginalny format adnotacji „Scalabel” został przekształcony do formatu „COCO” z wykorzystaniem biblioteki Pycocotools.

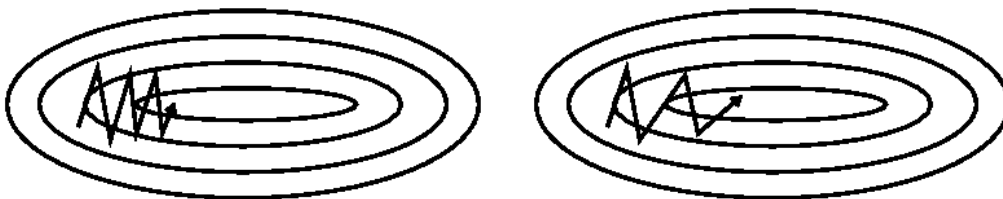
Scalabel [15] – platforma umożliwiająca wizualizację i etykietowanie danych graficznych 2D oraz 3D. Narzędzie to zostało wykorzystane w analizie zbioru danych, w celu obliczenia ilości wystąpień danych klas w zbiorze oraz do wizualizacji otrzymanych rezultatów przez wytrenowane sieci neuronowe. Według dokumentacji zbioru danych BDD100K [8] adnotacje użyte w zbiorze są zgodne z formatem Scalabel [7].

5.2. Parametry uczenia

Kluczowe właściwości zastosowane podczas treningu sieci neuronowych w niniejszej pracy:

- optymalizator SGD,
- momentum = 0.9,
- learning rate = 0.04,
- weight decay = 0.0001,
- liczba epok = 60

Do trenowania sieci neuronowych został zastosowany optymalizator SGD „Stochastic gradient descent” [16], który polega na iteracyjnym aktualizowaniu parametrów modelu na podstawie pojedynczego lub małego zestawu przykładów treningowych (wielkość zestawu danych wtłaczanych do pojedynczej iteracji treningowej jest określany jako parametr „batch_size”). W każdej iteracji SGD oblicza gradient parametrów względem funkcji kosztu, a następnie aktualizuje parametry modelu w kierunku przeciwnym do gradientu z zachowaniem momentum, czyli uwzględnieniem wcześniejszego kierunku spadku. Dzięki zastosowaniu momentum, optymalizator SGD jest bardziej stabilny i szybszy w porównaniu do klasycznego SGD, dzięki temu unika się sytuacji utknięcia w lokalnych minimum funkcji (Rys. 9). Proces optymalizacji powtarzany jest iteracyjnie, aż do osiągnięcia zadowalającej dokładności modelu. Proces ten jest szerzej wytłumaczony w [17].



Rys. 9. Porównanie optymalizatora SGD bez momentum oraz z momentum. Środek płaszczyzny przedstawia optymalne minimum funkcji kosztu [17]

„Learning rate” jest to wartość, która określa, o ile parametry modelu powinny być aktualizowane podczas każdej iteracji uczenia. Jeśli współczynnik uczenia jest zbyt duży, model może „przeskakiwać” optymalne minimum funkcji i zatrzymywać się w niestabilnym obszarze. Natomiast jeśli jest zbyt mały, proces uczenia może być zbyt powolny.

„Weight decay” (kompresja wag) jest metodą regularyzacji, która polega na dodawaniu kary za zbyt duże wartości wag modelu w stosunku do funkcji kosztu. Celem tej metody jest zwiększenie stabilności i zapobieganie przeuczeniu modelu.

Liczba epok to parametr, który określa ilość przebiegów przez cały zbiór danych treningowych podczas procesu uczenia sieci neuronowej. W trakcie każdej epoki, sieć neuronowa aktualizuje wagi neuronów zgodnie z aktualnym kierunkiem gradientu. Zazwyczaj, im więcej epok jest używanych do uczenia sieci, tym lepsza dokładność jest osiągana. Jednakże po pewnym czasie, dalsze uczenie nie przynosi już znaczących poprawek i może prowadzić do przeuczenia modelu, czyli zbyt doskonałego dopasowania do aktualnych danych treningowych, co spowoduje słabszą generalizację i gorsze działanie na nowych danych (testowych).

5.3. Wybór i implementacja sieci neuronowych do detekcji

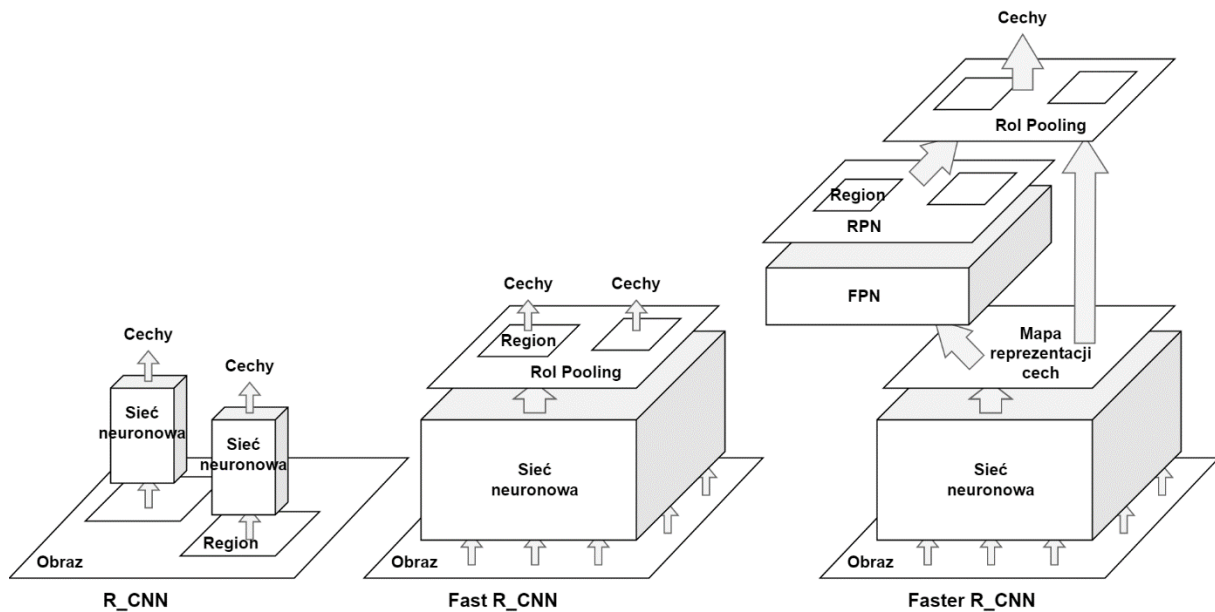
W niniejszej pracy zostały zaimplementowane i porównane jednokrokowe oraz dwukrokowe rodzaje algorytmów detekcji, których działanie opisane jest w tym podrozdziale:

5.3.1. Algorytmy dwukrokowe

W projekcie użyto sieci z rodziny "Region-Based Convolutional Neural Network (R-CNN)" [18], wykorzystujące dwuetapowy schemat pracy, który został opisany wcześniej w rozdziale 3.1.

Pierwsza sieć typu R-CNN została opublikowana w 2014 roku [19]. Algorytm proponujący ramki dla potencjalnych obiektów wyznaczał współrzędne dla ok. 2000 regionów na zdjęciu, po czym sieć klasyfikująca obliczała cechy zawarte w poszczególnych regionach. Wadą takiego podejścia był złożony obliczeniowo i wieloetapowy trening sieci, ponieważ dla jednego obrazka trzeba przejść 2000 razy przez sieć klasyfikacyjną. Dodatkowo wydłużał się czas testowania. Ze względu na ograniczenia sieci R-CNN zaproponowano szybszą wersję o nazwie Fast R-CNN [20]. Działanie Fast R-CNN polegało na przepuszczeniu obrazka wejściowego przez warstwy splotowe sieci, a następnie wygenerowaniu propozycji regionów ramek dopiero na podstawie splotowej reprezentacji cech obrazka. Dodatkowo, współrzędne ramek były automatycznie poprawiane przy pomocy tzw. „głowy regresyjnej”.

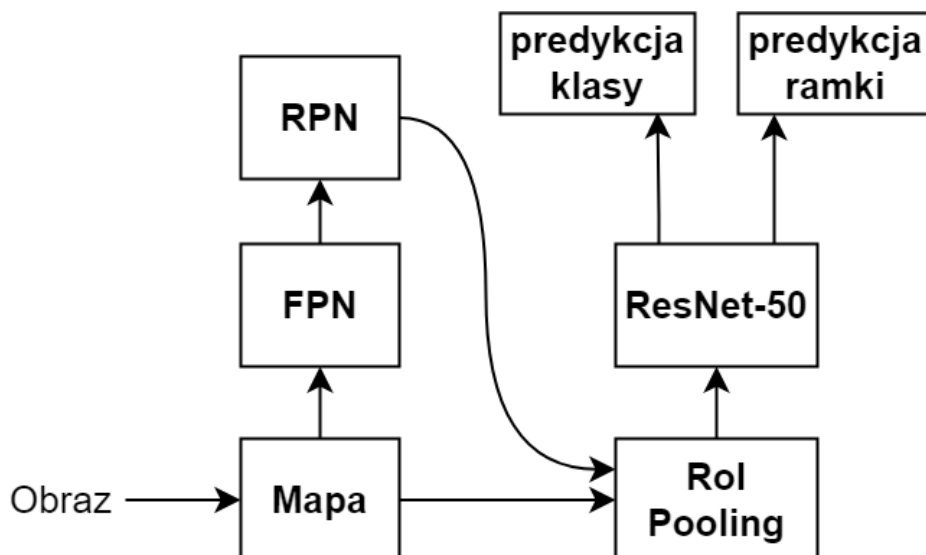
Jednym z najbardziej reprezentatywnych obecnie modeli sieci z omawianej rodziny jest Faster R-CNN [21], który wykorzystuje dodatkowy komponent sieci neuronowej „Region Proposal Network (RPN)” zamiast sztywnej, algorytmicznej metody do proponowania pozycji regionów. Prowadzi to do znacznego przyspieszenia działania sieci, ponieważ dalej procesowane są tylko regiony wygenerowane przez RPN, zamiast 2000 wyznaczanych wcześniej regionów przez stary algorytm.



Rys. 10. Schemat porównujący architekturę sieci R-CNN, Fast R-CNN i Faster R-CNN

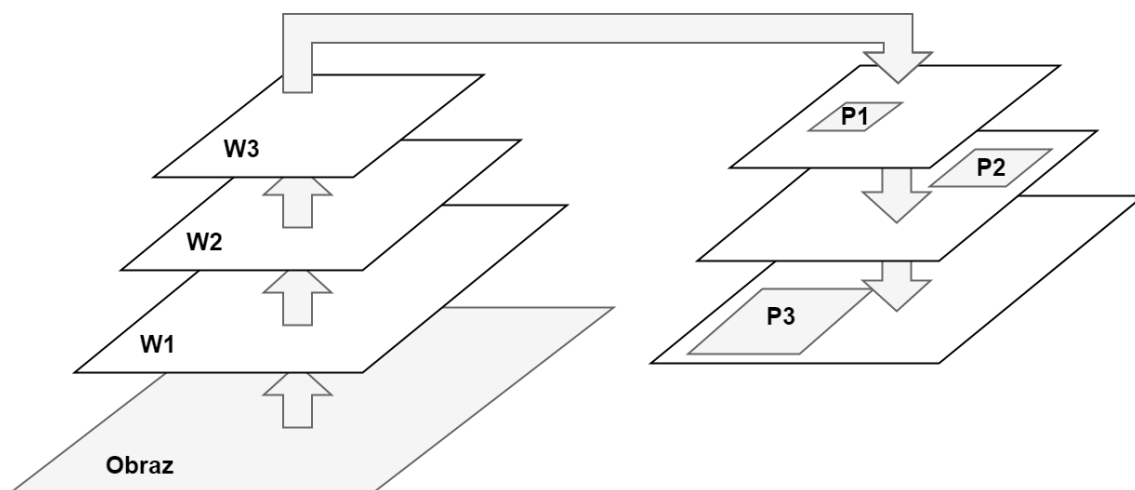
Główną wadą wszystkich opisanych sieci R-CNN jest brak widoczności kontekstu całego obrazka dla sieci klasyfikującej wydzielony region.

W projekcie przetestowane zostały poszczególne konfiguracje sieci udostępnione na stronie internetowej zbioru BDD100K [22], które posiadają wstępnie wytrenowane wagi modeli zaimportowane z ogólnodostępnej biblioteki PyTorch [23]. W projekcie przetestowano dwie sieci z użyciem dwukrokowego detektora: Faster R-CNN i Cascade R-CNN. Poniżej przedstawiono i opisano konfigurację zaimplementowanej w projekcie sieci Faster R-CNN:

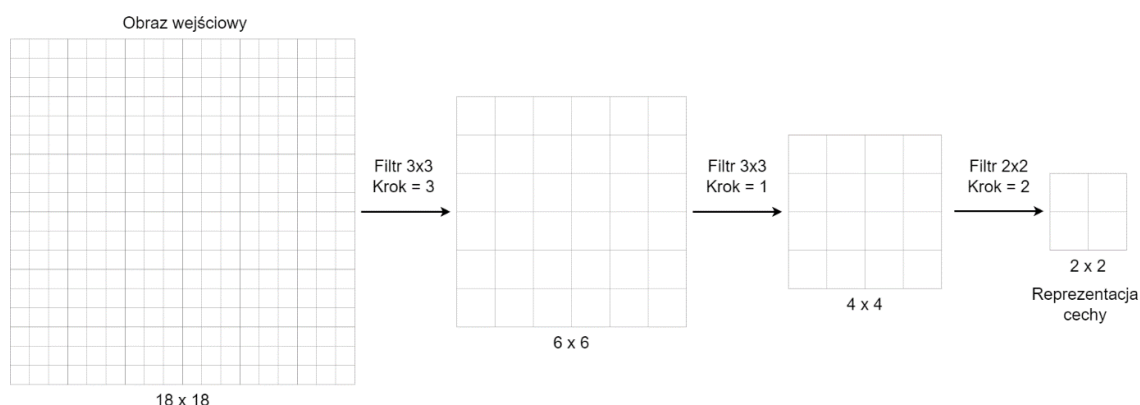


Rys. 11. Schemat przedstawiający działanie zaimplementowanej sieci Faster R-CNN, gdzie „Mapa” to warstwa sieci neuronowej zakodowana jako mapa reprezentacji cech

Feature Pyramid Network (FPN) - komponent sieci Faster R-CNN, który jest przeznaczony do rozpoznawania cech charakterystycznych dla konkretnych obiektów [24]. Działa on poprzez tworzenie piramidy warstw spłotowych reprezentujących obraz w różnej skali rozdzielczości, która pozwala na rozpoznanie cech obiektów o różnej wielkości. Jego działanie można zobrazować na poniższych schematach:



Rys. 12. Schemat przedstawiający działanie modułu FPN, gdzie W_x to kolejne warstwy spłotowe, P_x to predykcje cech [24]



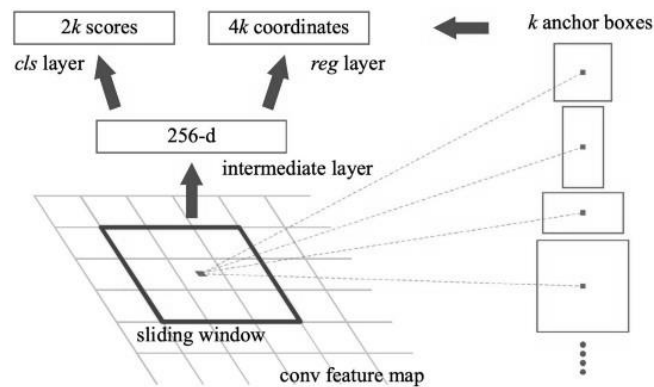
Rys. 13. Schemat przedstawiający zmniejszenie reprezentacji obrazu w FPN. Filtr (np. MaxPooling) koduje w kolejnej warstwie tylko najbardziej znaczące wartości pikseli, dzięki temu wyszczególnione zostają istotne cechy na obrazie [25]

Do konfiguracji modułu FPN wymagane jest podanie 2 podstawowych parametrów: 1) liczba kanałów wejściowych przekazywana do każdej warstwy FPN. 2) liczba kanałów wyjściowych. Dodatkowo w konfiguracji modułu ustawiono liczbę wyjściowych predykcji cech. Zastosowane w projekcie parametry modułu FPN:

Tabela 4. Parametry modułu FPN w Faster R-CNN

Rozmiar kanałów wejściowych	[256, 512, 1024, 2048]
Rozmiar kanału wyjściowego	256
Liczba wyjściowych predykcji	5

Region Proposal Network (RPN)- komponent sieci Faster R-CNN, służący do generowania potencjalnych regionów, w których może znajdować się obiekt. RPN generuje propozycje obszarów na podstawie mapy reprezentacji cech utworzonej przez moduł FPN. RPN przesuwając się po mapie reprezentacji cech i za pomocą regresji i klasyfikacji określa, czy na wybranym okienku może znajdować się potencjalny obiekt [26]. Jeśli tak, wygenerowane zostaną współrzędne takiego regionu (Rys.14).



Rys. 14. Schemat przedstawiający działanie przesuwającego okienka w RPN. Po określeniu, czy na wybranym obszarze może znajdować się potencjalny obiekt, RPN generuje współrzędne regionu. Wymiary generowanych regionów są ogólnie określone [26]

Zastosowane w projekcie parametry modułu RPN pokazano w poniższej Tabeli 4.

Tabela 5. Parametry modułu RPN w Faster R-CNN

Rozmiar kanału wejściowego	256
Rozmiar kanału pośredniego	256
Krok (o ile przesuwa się okienko)	[4, 8, 16, 32]

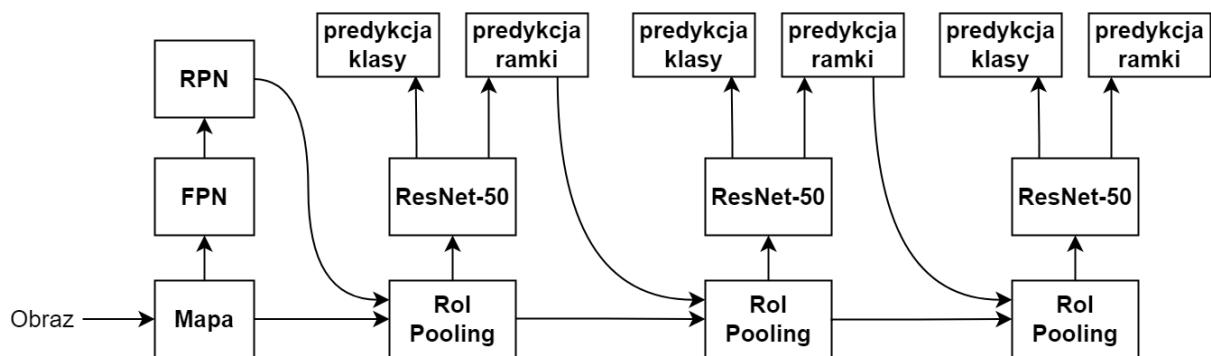
Na wyjściu otrzymujemy współrzędne regionów przekazywane do kolejnej warstwy sieci Faster R-CNN, czyli Region of Interest Pooling.

Region of Interest Pooling (RoI Pooling) – komponent służący do skalowania obszarów wygenerowanych przez RPN. Ponieważ moduł piramidowy FPN przekazuje wykryte cechy do RPN w różnej skali (Rys. 12) potrzebny jest komponent skalujący wygenerowane regiony odpowiednio do jednolitego rozmiaru. RoI Pooling skaluje otrzymane regiony jednocześnie

skalując odpowiednio obszar mapy reprezentacji cech z modułu FPN, na którym oznaczony został dany region. W tym wypadku rozmiar wyjściowy przeskalowanych regionów RoI to 7x7 pikseli.

Kolejnym etapem jest przekazanie regionów z warsty RoI Pooling do głównego komponentu, czyli splotowej sieci neuronowej. Zaimplementowany w projekcie model Faster R-CNN wykorzystuje w tym celu wstępnie wytrenowaną sieć ResNet-50 [27], której dodatkowa konfiguracja nie jest potrzebna. Rozmiary warstw i liczbę parametrów sieci ResNet-50 można znaleźć w artykule [27]. Sieć ResNet-50 jest wytrenowana do przewidywania zarówno klas obiektów, jak i współrzędnych ramki obiektów. Prawdopodobieństwo wystąpienia danej klasy jest generowane przez funkcję „Softmax” [28]. Klasa z najwyższym prawdopodobieństwem jest następnie przekazywana na wyjściu jako końcowa etykieta klasy. Współrzędne ramki są przewidywane przy pomocy regresji i wyznaczają obramowanie obiektu.

Poniżej przedstawiono schemat zaimplementowanej w projekcie sieci Cascade R-CNN (Rys.15):



Rys. 15. Schemat przedstawiający działanie sieci Cascade R-CNN, gdzie „Mapa” to warstwa sieci neuronowej zakodowana jako mapa reprezentacji cech

Zastosowana w projekcie sieć Cascade R-CNN wykorzystuje takie same komponenty jak w Faster R-CNN. Różnicą jest kaskadowa architektura sieci. Taka modyfikacja powoduje, że sieć wielokrotnie oblicza i poprawia współrzędne ramki. Dzięki temu Cascade R-CNN powinno wyznaczać dokładniejsze obramowanie obiektu. Konfiguracja poszczególnych modułów sieci nie została zmieniona.

5.3.2. Algorytmy jednokrokowe

Problem braku widoczności kontekstu całego obrazka w dwukrokowych detektorach rozwiązuje propozycja sieci z użyciem detektora jednokrokowego. Kontekst jest widoczny dla sieci, ponieważ procesowane jest jednocześnie całe pole obrazu wejściowego, zamiast tylko wyznaczonych regionów.

Zaimplementowanym w niniejszej pracy modelem sieci neuronowej z jednokrokovym algorytmem detekcji jest sieć FCOS (Fully Convolutional One-Stage Object Detection) [29]. Posiada jednokrokovy model detekcji obiektów, który wykorzystuje omawiany wcześniej moduł FPN (Rys. 12). Na podstawie wygenerowanych cech kolejne warstwy sieci spłotowej przewidują pozycję i klasę dla każdego potencjalnego obiektu na obrazie. Działa to poprzez przesuwanie się filtra spłotowego po całej mapie reprezentacji cech i wykrywanie poszczególnych obiektów. Dzięki temu FCOS jest w stanie wykrywać obiekty na granicy obrazu, co pozwala na lepszą dokładność w wykrywaniu przyciętych obiektów [30]. Zastosowane w projekcie parametry modułu FPN pokazano na poniższej Tabeli 6:

Tabela 6. Parametry modułu FPN w FCOS

Rozmiar kanałów wejściowych	[256, 512, 1024, 2048]
Rozmiar kanału wyjściowego	256
Liczba wyjściowych predykcji	5

Zastosowane w projekcie parametry kolejnych warstw spłotowych pokazano na poniższej Tabeli 7:

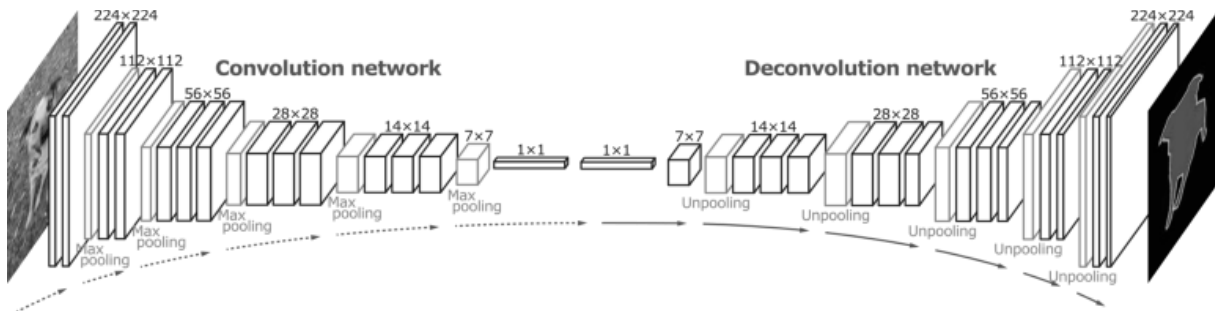
Tabela 7. Parametry warstw spłotowych w FCOS

Rozmiar kanałów wejściowych:	[2048, 1024, 512, 256]
Krok (o ile przesuwa się filtr)	[128, 64, 32, 16, 8]

5.4. Zaimplementowane sieci do segmentacji

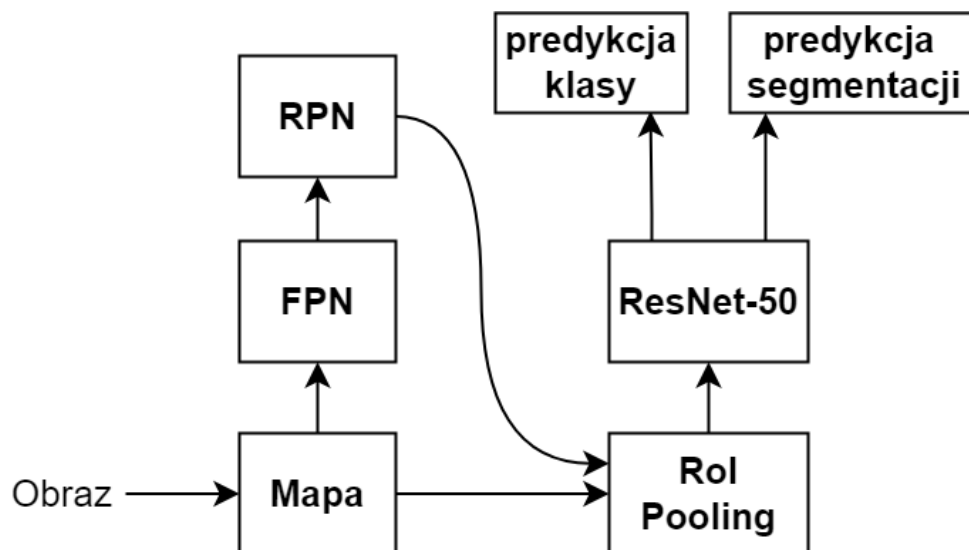
Do segmentacji używa się zazwyczaj spłotów transponowanych (ang. up-convolution), czyli odwrotności działania operacji spłotowej. Używa się ich szczególnie w sieciach spłotowych, wykorzystujących koncepcję autoenkoderów (Rys. 16). Taka architektura składa się z dwóch części: enlodera i delodera. Encoder zajmuje się kodowaniem, czyli zmniejszaniem reprezentacji obrazu oraz zwiększeniem abstrakcji danych poprzez kodowanie tylko najważniejszych cech obrazu. Dekoder natomiast zajmuje się zwiększaniem rozmiaru obrazu oraz generowaniem map segmentacji. Na tym etapie wykorzystuje się właśnie działanie

splotów transponowanych, poprzez odwrotność działania filtra w procesie dekodowania. Celem takiego zabiegu jest przywrócenie zmniejszonej reprezentacji obrazu (otrzymanej w procesie kodowania) do rozmiaru początkowego i jednocześnie lepszego dopasowania obszarów segmentacji. Poglądowy przykład działania sieci splotowej do segmentacji (autoenkoder) jest widoczny na poniższej grafice (Rys.16):



Rys. 16. Poglądowy przykład działania autoenkodera. Od lewej: architektura enkodera („convolution network”), architektura dekodera („deconvolution network”) [31]

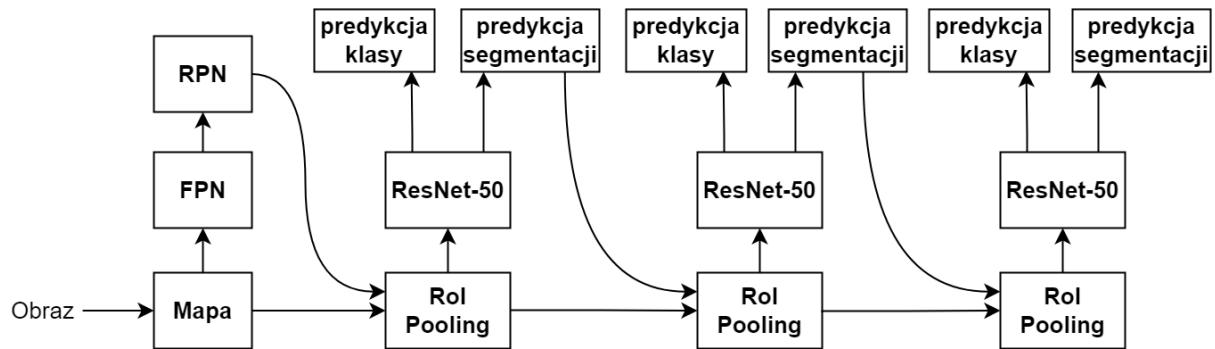
W niniejszej pracy do segmentacji instancji przetestowane zostały dwie sieci, wykorzystujące połączenie omawianych wcześniej schematów działania Faster R-CNN oraz sieci splotowa w architekturze autoenkodera. Poniżej (Rys. 17) pokazany został schemat działania pierwszej z sieci o nazwie Mask R-CNN:



Rys. 17. Schemat działania sieci Mask R-CNN

Jak widać na powyższym schemacie (Rys. 17), model Mask R-CNN wykorzystuje schemat działania modelu Faster R-CNN, przy czym wykorzystana sieć ResNet-50 działa w tym wypadku jako autoenkoder. Zastosowane w sieci warstwy dekodujące odpowiadają za predykcję obszarów segmentacji.

Podobnym schematem działania charakteryzuje się druga sieć użyta do segmentacji instancji, czyli Cascade Mask R-CNN, której schemat zamieszczono poniżej (Rys. 18). Zastosowane parametry warstw kodujących i dekodujących sieci ResNet-50 pozostają takie same jak w konfiguracji sieci Mask R-CNN.



Rys. 18. Schemat działania sieci Mask R-CNN

6. Ewaluacja wyników

Skuteczność działania modelu na rzeczywistych danych jest kluczowym aspektem oceniającym jego potencjalne zastosowanie w praktyce. Ewaluacja wyników sieci neuronowych przeprowadzana jest na zbiorze testowym, czyli na danych, których model wcześniej nie widział. Na podstawie takich danych możliwe jest określenie zdolności wytrenowanego modelu do generalizacji, czyli radzenia sobie z rozpoznawaniem rzeczywistych cech obiektów, a nie tylko tych ujranych podczas treningu. W celu pomiaru skuteczności działania sieci neuronowych w kontekście detekcji i segmentacji używa się wielu przystosowanych do tego metryk. W tym rozdziale opisane zostały metryki do ewaluacji wyników detekcji i segmentacji oraz przedstawiono zestawienie wyników otrzymanych w niniejszej pracy. Do wyznaczenia wyników posłużyły dane testowe, tj. 20 tysięcy obrazów.

6.1. Opis wybranych metryk

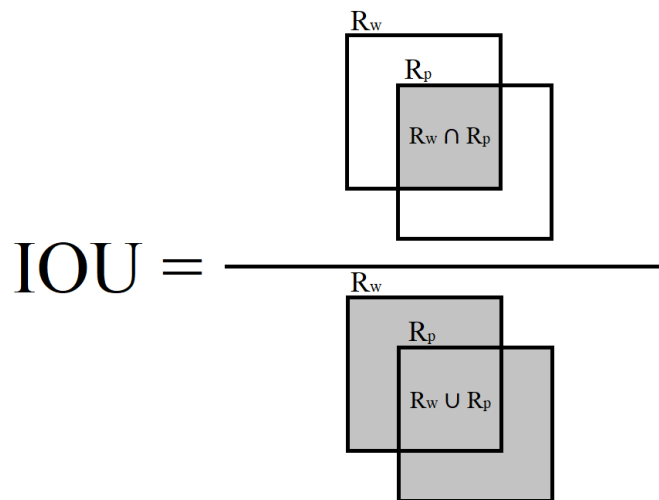
Detektory zapisują wykryty obiekt w trzech atrybutach: klasa obiektu, współrzędne ramki i przedział ufności, czyli „poziom przekonania” detektora, że wykryty obiekt należy do przypisanej klasy (generowana przez funkcję SoftMax [28], której zakres mieści się w przedziale od 0 do 1). Do ewaluacji tych predykcji potrzebne są etykiety ze zbioru danych, które określają prawidłowe atrybuty obiektu. W przypadku detekcji najczęściej mierzy się jak daleko znajdują się współrzędne wyznaczonej ramki w stosunku do prawidłowej ramki. Pomiar ten jest wykonywany niezależnie dla każdej klasy, poprzez ocenę stopnia nakładania się obszarów przewidywanych i prawidłowych.

IOU

Zakładając, że pole obszaru wyznaczone przez ramkę prawdziwą (etykietę ze zbioru) to R_p , a pole obszaru wyznaczone przez ramkę wykrytą (predykcję detektora) to R_w , perfekcyjnym dopasowaniem nazywa się stan, kiedy położenie obydwu ramek jest takie same. W rzeczywistości jednak taki stan pojawia się niezwykle rzadko, dlatego do pomiaru podobieństwa pomiędzy położeniem ramek R_p i R_w służy miara „intersection over union (IOU)”. IOU jest równe części wspólnej ramki R_p i ramki R_w podzielone przez ich sumę:

$$IOU = \frac{(R_w \cap R_p)}{(R_w \cup R_p)} \quad (4)$$

Idealne dopasowanie ma miejsce, gdy $IOU = 1$, a jeśli obydwie ramki nie przecinają się, $IOU = 0$. Im bliżej 1 zbliża się IOU, tym lepsza jest detekcja. Poniżej (Rys. 19) zobrazowano działanie miary IOU:



Rys. 19. Zobrazowanie działania miary IOU

Ponieważ detektory dokonują także klasyfikacji obiektu, mierzone są tylko ramki obiektów tej samej klasy. IOU wykorzystuje się w bardziej zaawansowanych metrykach do ewaluacji wyników detekcji, mianowicie wyznacza się próg wartości IOU, który musi zostać przekroczony przez R_w . W takich miarach próg bliższy 1 jest bardziej restrykcyjny, ponieważ wymaga niemal doskonałych detekcji. Podczas, gdy próg bliższy 0 (ale jednocześnie różny od 0) oznacza łagodniejszą metrykę, biorącą pod uwagę nawet niewielkie przecięcia między R_w , a R_p . Wartości IOU są zwykle wyrażane w procentach, a najczęściej używane wartości progowe to 50% i 75%.

Precyzja i czułość

W dziedzinie sztucznej inteligencji i uczenia maszynowego, precyzję określa się jako zdolność modelu do prawidłowego identyfikowania obiektów. Jest wyrażana w procentach i oznacza odsetek prawidłowych, pozytywnych predykcji. Czułość określa zdolność modelu do znalezienia wszystkich obiektów. Jest to odsetek prawidłowych, pozytywnych predykcji wśród wszystkich szukanych obiektów. Aby obliczyć precyzję i czułość, każda wykryta ramka musi najpierw zostać sklasyfikowana jako:

- True Positive (TP) – prawidłowa predykcja pokrywająca się z R_p
- False Positive (FP) – nieprawidłowa predykcja istniejącego obiektu lub nieprawidłowa predykcja nieistniejącego obiektu
- False Negative (FN) – niewykrycie predykcji pokrywającej się z R_p

Przytaczając twierdzenie autorów publikacji [32] badających zależności miar do detekcji definicję precyzji i czułości należy definiować w następujący sposób. Zakładając, że istnieje

zbiór danych z G etykiet oraz model, który wykrywa N detekcji, z których S jest poprawnych ($S \leq G$), koncepcje precyzji można formalnie wyrazić jako (Rys. 20):

$$\text{Precyzja} = \frac{\sum_{n=1}^S TP_n}{\sum_{n=1}^S TP_n + \sum_{n=1}^{N-S} FP_n} = \frac{\sum_{n=1}^S TP_n}{\text{liczba wszystkich predykcji}}$$

Rys. 20. Wzór równania precyzji omawiany w [32]

Upraszczając, precyzja oznacza stosunek prawidłowo wykrytych przez algorytm obiektów (TP) do liczby wszystkich predykcji algorytmu (TP + FP). Analogicznie, definicję czułości autorzy publikacji [32] przedstawiają w następujący sposób (Rys.21):

$$\text{Czułość} = \frac{\sum_{n=1}^S TP_n}{\sum_{n=1}^S TP_n + \sum_{n=1}^{G-S} FN_n} = \frac{\sum_{n=1}^S TP_n}{\text{liczba wszystkich etykiet}}$$

Rys. 21. Wzór równania czułości omawiany w [32]

Czułość można opisać jako stosunek prawidłowo wykrytych przez algorytm obiektów (TP) do liczby prawidłowych predykcji i pozostałych, niewykrytych obiektów (TP + FN).

Average Precision (AP)

Wspomniany wcześniej w tym rozdziale atrybut, przedział ufności, może być uwzględniany w obliczeniach precyzji i czułości, uznając za pozytywne detekcje tylko te, których przedział ufności jest większy, niż wartość progowa τ . Detekcje, których wartość przedziału ufności jest mniejsza niż τ , mogą zostać uznane za negatywne. Dokładnie w ten sam sposób, można ustalić wartość progową τ dla IOU, która musi zostać spełniona, aby predykcja była uznana za pozytywną. Biorąc pod uwagę wyznaczone poprzednio wzory na predykcję i czułość, autorzy publikacji [32] rozszerzyli je do postaci funkcji od τ . Wzory prezentują się w następujący sposób:

$$\text{Precyzja}(\tau) = \frac{\sum_{n=1}^S \text{TP}_n(\tau)}{\sum_{n=1}^S \text{TP}_n(\tau) + \sum_{n=1}^{N-S} \text{FP}_n(\tau)} = \frac{\sum_{n=1}^S \text{TP}_n(\tau)}{\text{liczba wszystkich } (\tau) \text{ predykcji}}$$

$$\text{Czułość}(\tau) = \frac{\sum_{n=1}^S \text{TP}_n(\tau)}{\sum_{n=1}^S \text{TP}_n(\tau) + \sum_{n=1}^{G-S} \text{FN}_n(\tau)} = \frac{\sum_{n=1}^S \text{TP}_n(\tau)}{\text{liczba wszystkich etykiet}}$$

Rys. 22. Wzór równania funkcji precyzji i funkcji czułości omawiany w [32]

Do wyznaczenia miary Average Precision (AP) obliczana jest wartość pola pod krzywą $\text{Precyzja}(\tau) \times \text{Czułość}(\tau)$ po wcześniejszej interpolacji tej krzywej, w taki sposób, aby była ona funkcją monotoniczną. Otrzymane wyniki z wszystkich testowanych obrazów są następnie uśredniane. Ostateczną uśrednioną wartością jest omawiana miara Average Precision (AP) dla zadanych progów przedziału ufnosci i IOU, które w niniejszym projekcie zostały ustawione na wartości 50% i 75%. Podsumowując, im większe AP osiągnie model, tym lepsza detekcja.

Miara AP jest jedną z dwóch głównych miar zastosowaną do ewaluacji wyników detekcji w tej pracy, z racji na jednoczesne uwzględnienie wielu czynników przy pomiarze, tj. precyzji, czułości, IOU i przedziału ufnosci. Wyszczególniono kilka rodzajów miary:

- AP – z zastosowanym uśrednieniem 10 progów IOU: od 0.5 do 0.95 (z krokiem co 0.05)
- AP50 – próg τ ustawiony na 50%
- AP75 – próg τ ustawiony na 75%
- APs – do pomiaru tylko małej wielkości obiektów:

$$\text{pole ramki} < 32 \times 32 \text{ pikseli} \quad (5)$$

- APm – do pomiaru tylko średniej wielkości obiektów:

$$32 \times 32 \text{ pikseli} < \text{pole ramki} < 96 \times 96 \text{ pikseli} \quad (6)$$

Pomiar wielkich obiektów został pominięty, ze względu na specyfikę problemu pracy i charakterystykę zbioru danych.

Average Recall (AR)

Miara AR służy do pomiaru średniej czułości dla różnych progów IOU przy detekcji. Do obliczenia metryki Average Recall nie uwzględnia się progu poziomu zaufania, a jedynie próg IOU. Jeśli próg IOU został przekroczony wszystkie dokonane predykcje w tym przypadku są

uznane na pozytywne. Do wyznaczenia miary AR bierze się pod uwagę tylko uśrednioną wartość najwyższych czułości jakie osiągnął model dla 10 progów IOU: od 0.5 do 0.95 (z krokiem co 0.05). Miara AP jest drugą główną miarą zastosowaną do ewaluacji wyników detekcji w niniejszej pracy. Wyszczególniono kilka rodzajów miary:

- AR1 – miara AR tylko dla pierwszej detekcji każdego obiektu na obrazie. Pozwala uwzględnić jak dobrze model jest w stanie znaleźć obiekt w pierwszej wygenerowanej predykcji.
- AR10 - miara AR tylko dla pierwszych 10 detekcji. Mierzy jak dobrze model znajduje obiekty w pierwszych 10 wygenerowanych predykcjach.
- AR100 - miara AR dla pierwszych 100 detekcji.
- ARs – do pomiaru tylko małej wielkości obiektów:

$$\text{pole ramki} < 32 \times 32 \text{ pikseli} \quad (5)$$

- ARm – do pomiaru tylko średniej wielkości obiektów:

$$32 \times 32 \text{ pikseli} < \text{pole ramki} < 96 \times 96 \text{ pikseli} \quad (6)$$

Celem segmentacji jest sklasyfikowanie każdego piksela na obrazie. Do ewaluacji wyników segmentacji instancji używa się takich samych jak w przypadku detekcji. Różnicą jest natomiast dostosowanie obliczenia IOU do dokładnego pola obszaru sklasyfikowanego przez model obiektu, zamiast pola prostokątnej ramki (w przypadku detekcji).

6.2. Zestawienie i prezentacja wyników

Wyniki poszczególnych sieci zostały umieszczone w poniższych tabelach. Każda tabela przedstawia osobne wyniki miar Average Precision oraz Average Recall. Pierwsza część tego podrozdziału przedstawia wyniki detekcji, następnie przedstawiono wyniki segmentacji. Każdy wynik miar został rozdzielony ze względu na poszczególne klasy. Pozwala to dokładniej przyjrzeć się zróżnicowaniu w efektywności wykrywania poszczególnych obiektów przez sieci.

6.2.1. Wyniki detekcji

Wyniki miary AP dla sieci Faster R-CNN (Tabela 8) udowadniają, że model ten wykonuje bardzo dobrą detekcję klas: „samochód”, „ciężarówka” i „bus”. Powodem takich rezultatów jest duża ilość wystąpień tych obiektów w zbiorze danych, dlatego model najlepiej zapamiętał cechy charakterystyczne dla takich obiektów. Jak widać, klasa „pociąg” posiada zerową liczbę pozytywnych predykcji, ponieważ występowała zbyt rzadko w zbiorze treningowym. Jeśli chodzi o detekcję małych obiektów, najlepiej wypada klasa „znak drogowy”. Model nauczył się bardzo dobrze rozpoznawać charakterystykę znaków drogowych, dlatego mimo małego rozmiaru tych obiektów zostają one dobrze rozpoznane na obrazie. Odwrotnie jest w przypadku klas: „rowerzysta”, „motocykl”, „rower”. Predykcja średniej wielkości obiektów wypada najlepiej dla klas: „samochód”, „znak drogowy”, „pieszy” oraz „sygnalizacja świetlna”. Inne obiekty średniej wielkości również zostały wykrywane w zadowalający sposób.

Tabela 8. Wyniki Average Precision sieci Faster R-CNN

Miara Klasa	AP	AP50	AP75	APs	APm
pedestrian	35.41	70.55	30.89	19.62	45.46
rider	26.69	53.88	23.3	8.33	31.4
car	51.89	83.21	52.89	27.48	59.28
truck	46.78	66.64	52.54	16.58	41.34
bus	48.05	65.64	54.3	13.77	38.53
train	0	0	0	0	0
motorcycle	24.65	50.45	18.24	9.37	27.74
bicycle	25.84	54.61	21.65	9.44	27.17
traffic light	24.59	59.76	15.37	21.08	45.1
traffic sign	39.12	70.13	38.86	31.04	58.5
Średnia	32.3	57.49	30.81	15.67	37.45

Wyniki miary AR dla sieci Faster R-CNN (Tabela 9) udowadniają, że model najrzadziej pomijał predykcje dla klas: „ciężarówka”, „bus” i „samochód”. Powodem takich rezultatów może być wielkość tych obiektów. Ponieważ wymienione obiekty są duże, model nie ma problemu z ich znalezieniem, dlatego miara AR osiąga wysoką wartość. Podobnie jak w przypadku miary AP, najniższe wartości przypadają dla małych obiektów. W tym przypadku najniższe ARs występuje dla klas: „rowerzysta”, „motocykl”, „rower”.

Tabela 9. Wyniki Average Recall sieci Faster R-CNN

Miara Klasa	AR1	AR10	AR100	ARs	ARm
pedestrian	13.23	39.61	44.51	32.14	54.06
rider	31.57	36.78	36.78	17.29	43.85
car	8.31	47.78	57.52	37.45	64.83
truck	40.9	63.05	63.11	37.77	60.45
bus	49.49	62.11	62.11	31.67	57.3
train	0	0	0	0	0
motorcycle	29.93	36.09	36.46	17.58	40.77
bicycle	24.41	35.04	36.06	19.49	37.52
traffic light	8.24	31.86	32.44	29.34	53.29
traffic sign	14.56	47.54	48.81	41.78	67.47
Średnia	22.06	39.99	41.78	26.45	47.95

Wyniki miary AP dla sieci Cascade R-CNN (Tabela 10) osiągają większe wartości niż pozostałe sieci. Największe wartości miary AP75 osiągają klasy: „samochód”, „ciężarówka”, „bus”. Tym samym, są one większe niż w przypadku sieci Faster R-CNN, co można uzasadnić ich odmienną architekturą. Cascade R-CNN wielokrotnie poprawia predykcje ramki, dlatego wyznaczone detekcje są dokładniejsze. Podobnie jak w poprzednim przypadku, model nie potrafi wykrywać klasy „pociąg”. Model wykrywa bardzo dobrze obiekty średniej wielkości, jednocześnie osiągając słabsze wyniki dla predykcji małych obiektów.

Tabela 10. Wyniki Average Precision sieci Cascade R-CNN

Miara Klasa	AP	AP50	AP75	APs	APm
pedestrian	37.22	70.71	34.74	20.3	47.96
rider	27.51	54.13	23.94	7.6	32.82
car	53.32	83.83	54.76	28.51	61.03
truck	47.93	66.74	54.03	17.25	42.23
bus	49.77	66.02	55.92	14.55	39.43
train	0	0	0	0	0
motorcycle	27.65	52.21	25.76	10.24	29.98
bicycle	27.18	54.36	23.41	8.81	28.68
traffic light	25.85	60.84	17.29	22.15	47.24
traffic sign	40.8	70.94	41.44	32.27	60.81
Średnia	33.72	57.98	33.13	16.17	39.02

Wyniki miary AR dla sieci Cascade R-CNN (Tabela 10) osiągają lepsze wyniki, niż pozostałe sieci. Model posiada wysoką czułość na detekcję obiektów klasy: „bus”, „samochód”, „ciężarówka” i „znak drogowy”. Wyniki czułości dla małych obiektów są zadowalające. W odróżnieniu od pozostałych modeli, Cascade R-CNN nie pomija małych obiektów tak często jak pozostałe sieci. Obiekty średniej wielkości podlegają bardzo dobrej detekcji, o czym świadczą wysokie wyniki miary ARm osiągnięte przez omawiany model.

Tabela 11. Wyniki Average Recall sieci Cascade R-CNN

Miara Klasa	AR1	AR10	AR100	ARs	ARm
pedestrian	13.94	41.41	46.12	33.06	56.13
rider	31.32	36.77	36.78	17.33	44.13
car	8.39	48.78	58.67	38.38	66.3
truck	41.69	63.97	64	39.35	60.9
bus	50.11	62.55	62.55	33.50	56.61
train	0	0	0	0	0
motorcycle	32	37.98	38.15	17.73	42.09
bicycle	24.49	36.19	36.91	18.69	38.89
traffic light	8.59	33.18	33.77	30.60	55.06
traffic sign	14.97	49.01	50.45	43.36	69.34
Średnia	22.55	40.99	42.74	27.2	48.95

Wyniki miary AP dla sieci FCOS (Tabela 12) są najslabsze w zestawieniu wszystkich sieci do detekcji. Model osiąga dobre wyniki jedynie w przypadku miary AP50, gdzie próg IOU=0.5. Świadczy to o dobrej zdolności modelu do szybkiej, „niechlujnej” detekcji obiektów z niedokładnym oznaczeniem jego lokalizacji. Powodem takiego działania jest jednokrokowa architektura sieci, przystosowana do szybkiej pracy. Dlatego model FCOS osiąga tym samym mniejszą precyzję w detekcji. Model najgorzej radzi sobie z wykrywaniem małych obiektów, wyniki miary APs są w tym przypadku niezadowalające.

Tabela 12. Wyniki Average Precision sieci FCOS

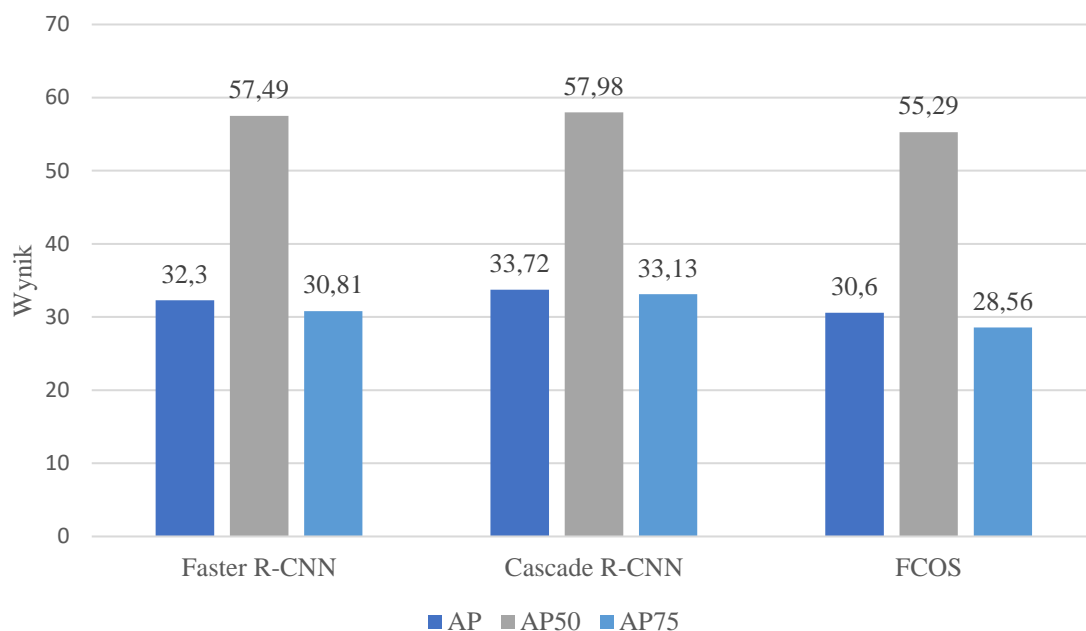
Miara Klasa	AP	AP50	AP75	APs	APm
pedestrian	33.16	67.43	28.49	17.19	44.01
rider	21.14	45.32	17.2	4.61	26.47
car	50.44	81.41	51.07	24.82	58.52
truck	45.43	63.33	50.52	12.25	38.59
bus	47.49	63.53	53.11	9.69	36.18
train	0	0	0	0	0
motorcycle	21.3	44.03	16.9	6.67	23.89
bicycle	21.9	49.24	15.64	5.34	24.1
traffic light	26.18	67.56	14.85	22.7	44.91
traffic sign	38.97	70.96	37.86	30.36	59.23
Średnia	30.6	55.29	28.56	13.36	35.59

Analizując wyniki miary AR dla sieci FCOS (Tabela 13) można wywnioskować, że model bardzo dobrze wykrywa poszczególne instancje obiektów. Model osiąga wysokie wyniki miary AR, co świadczy o tym, że nie pomija wielu obiektów podczas detekcji. Wyniki miary AR100 dla sieci FCOS przewyższają wyniki sieci Faster R-CNN. Można jednoznacznie stwierdzić, że sieć FCOS ze względu na szybkość działania poświęca jedynie precyzję detekcji, jednakże czułość w wykrywaniu wszystkich obiektów nadal pozostaje wysoka.

Tabela 13. Wyniki Average Recall sieci FCOS

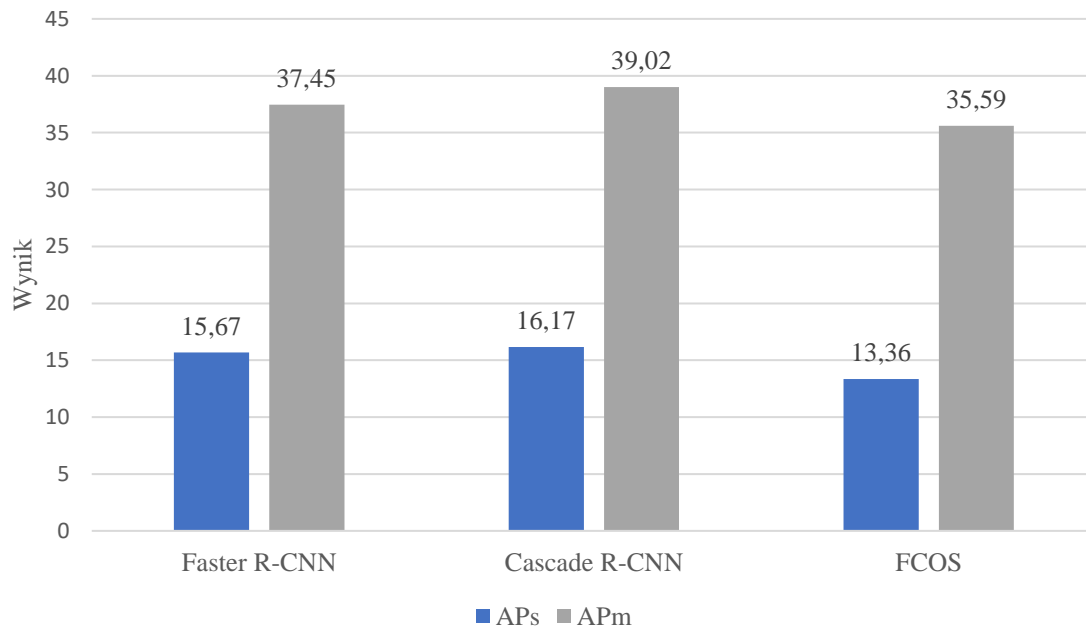
Miara Klasa	AR1	AR10	AR100	ARs	ARm
pedestrian	12.92	38.49	43.45	29.4	54.59
rider	27.87	35.23	35.32	14.29	43.81
car	8.29	46.98	56.61	34.9	65.15
truck	40.04	63.64	64.17	36.45	61.18
bus	49.93	65.64	65.67	36.58	59.87
train	0	0	0	0	0
motorcycle	28.04	36.78	37.09	15.08	41.66
bicycle	21.67	32.86	34.21	12.74	37.47
traffic light	8.44	35.41	37.44	34.9	54.58
traffic sign	14.44	48.10	51.48	44.8	69.32
Średnia	21.17	40.31	42.54	25.91	48.76

Poniższy wykres (Rys. 23) przedstawia porównanie miar AP dla wyznaczonych progów IOU, omawianych wcześniej w tym rozdziale. Jak widać, model Cascade R-CNN osiąga najwyższą dokładność w procesie wyznaczania pozycji ramki obiektu. Jest to potwierdzeniem tezy, że zastosowana kaskadowa modyfikacja architektury Faster R-CNN sprawdza się w poprawieniu wyznaczania współrzędnych ramki. Sieć FCOS, która jest detektorem jednokrokovym poświęca swoją dokładność na rzecz prostszej architektury, a tym samym większej szybkości działania i mniejszego poboru zasobów obliczeniowych. Ważnym aspektem jest znaczący spadek wyników dla zwiększonego progu IOU. Średni wynik miary AP75 dla wszystkich sieci jest 1.84 razy niższy niż wynik miary AP50.



Rys. 23. Wykres przedstawiający wyniki AP, AP50 i AP75 sieci użytych do detekcji

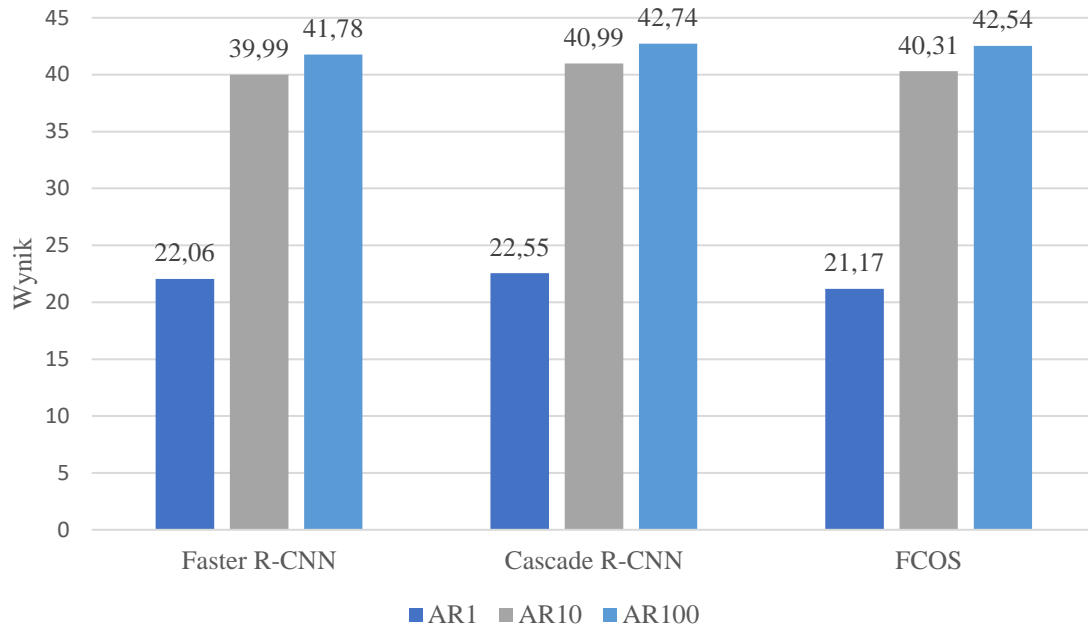
Poniższe porównanie (Rys. 24) przedstawia wyniki wykrywania obiektów w zależności od ich wielkości na obrazie. Każda z sieci osiąga lepsze wyniki w wykrywaniu większych obiektów. Sieć Cascade R-NN w tym przypadku również osiąga najlepsze wyniki.



Rys. 24. Wykres przedstawiający wyniki APs i APm sieci użytych do detekcji

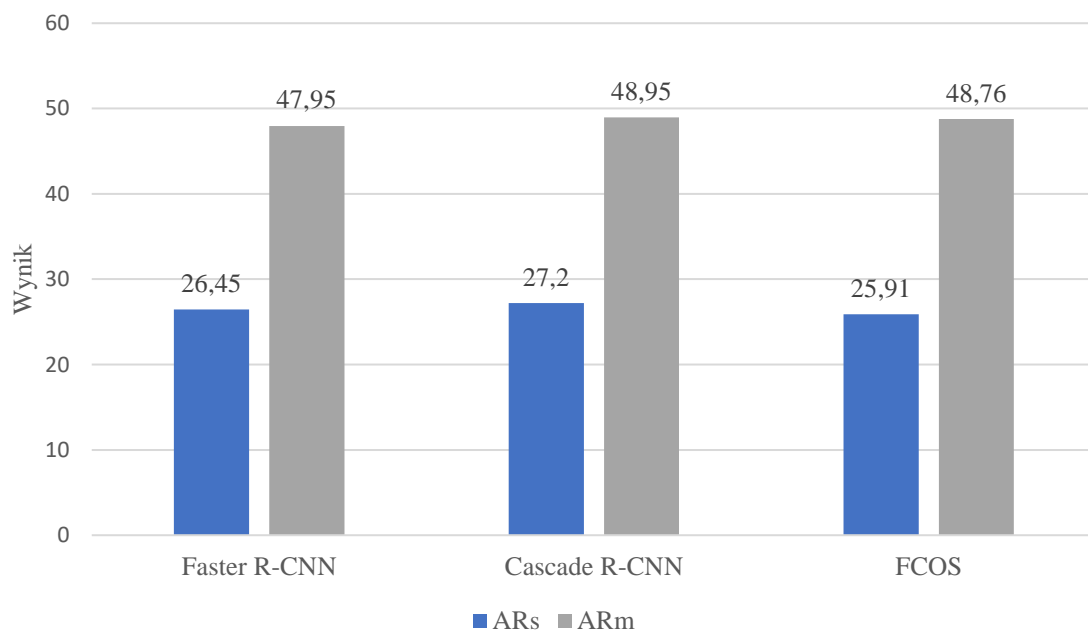
Poniżej przedstawiono (Rys. 25) porównanie w osiągniętej przez modele średniej czułości. Podobnie jak w przypadku analizy precyzji, model Cascade R-CNN osiągnął najlepsze wyniki, dopełniając przy tym wniosek, że ten model poradził sobie najlepiej w całym procesie detekcji obiektów zaimplementowanym w niniejszej pracy. Warto zauważyć, że sieć FCOS osiąga

większą wartość AR dla pierwszych 10 i 100 predykcji niż sieć Faster R-CNN. Można wywnioskować, że sieć FCOS nakładała większe kary w procesie trenowania za nieznajdowanie obiektów, niż za nieprecyzyjne znajdowanie położenia obiektów.



Rys. 25. Wykres przedstawiający wyniki AR1, AR10 i AR100 sieci użytych do detekcji

Poniżej przedstawione zostało (Rys. 26) porównanie wyników AR w zależności od wielkości obiektów na obrazie. Podobnie jak w przypadku analizy wyników AP, sieci gorzej wykrywają małe obiekty na obrazach. Model Cascade R-CNN osiągnął najlepsze wyniki.



Rys. 26. Wykres przedstawiający wyniki ARs i ARm sieci użytych do detekcji

6.2.2. Wyniki segmentacji

Wyniki miary AP dla sieci Mask R-CNN (Tabela 14) osiągają największe wartości dla dużych obiektów. Najlepiej segmentowane obiekty to klasa „samochód”. Powodem tego jest wysoka ilość wystąpień tej klasy w zbiorze treningowym. W przypadku restrykcyjnej miary AP75, klasa „samochód” również uzyskuje bardzo dobre wyniki. Najgorzej wypadają segmentacje dla małej wielkości obiektów, które osiągają gorsze wyniki, niż w przypadku detekcji. W tym przypadku, model nie jest w stanie wyznaczyć obszarów dla klas: „rower” i „rowerzysta”. Jednakże biorąc pod uwagę mniej restrykcyjną miarę AP50, wyniki segmentacji (która wymaga dokładnego zaznaczenia obszaru obiektu) są bardzo wysokie. Można uznać, że segmentacja „przybliżonego” obszaru obiektów wypada bardzo dobrze dla sieci Mask R-CNN. Zważając na trudność zadania jakim jest segmentacja obiektów, można uznać, że wyniki sieci Mask R-CNN przy segmentacji obiektów klasy: „pieszy”, „samochód”, „ciężarówka”, „bus” i „motocykl” osiągają bardzo wysokie wartości.

Tabela 14. Wyniki Average Precision sieci Mask R-CNN

Miara Klasa	AP	AP50	AP75	APs	APm
pedestrian	30.2	59.13	28.41	19.71	48.94
rider	8.3	24.04	1.46	5.75	10.77
car	45.93	67.94	49.56	22.66	62.43
truck	26.36	36.61	30.81	9.71	25.58
bus	28.84	38.37	33.52	2.82	22.88
train	0	0	0	0	0
motorcycle	12.93	35.66	7.67	6.99	17.88
bicycle	6.26	19.97	0.6	3.57	10.4
Średnia	19.85	35.22	19	10.17	24.86

Wyniki miary AR dla sieci Mask R-CNN (Tabela 15) sugerują bardzo wysoką czułość na detekcję obiektów klasy „samochód”. Najtrudniej modeli jest wykryć klasę „rower” i „rowerzysta”. Zakłada się, że modele do segmentacji wykorzystujące architekturę autoenkodera tracą dużą część informacji podczas kodowania, dlatego małe obiekty są trudne do wykrycia. Analizując wyniki miary ARs można wywnioskować, że taki problem może być rozwiązany poprzez zwiększenie liczności wystąpienia szukaney klasy w zbiorze danych. Model Mask R-CNN bardzo dobrze wychwytuje małe obiekty klasy „samochód”, które są najliczniejsze. Jednakże problem pozostaje w rozpoznawaniu pozostałych, mniej licznych klas.

Tabela 15. Wyniki Average Recall sieci Mask R-CNN

Miara Klasa	AR1	AR10	AR100	ARs	ARm
pedestrian	14.89	34.64	35.78	26.8	53.46
rider	10.44	10.44	10.44	6.67	14.62
car	7.8	44.09	50.07	29.65	67.12
truck	26.66	34.05	34.05	17.08	35.02
bus	29.41	35.21	35.21	7.75	31.58
train	0	0	0	0	0
motorcycle	17.87	17.87	17.87	8.5	23.5
bicycle	7.5	9.9	9.9	6.88	14.86
Średnia	14.32	23.27	24.16	14.76	30.02

Wyniki miary AP dla sieci Cascade Mask R-CNN (Tabela 16) osiągają mniejsze wartości w zestawieniu obydwu sieci do segmentacji. Model uzyskuje gorsze wyniki dla miar: AP i AP50 oraz delikatnie wyższe dla miary AP75. Podobnie jak w przypadku sieci Mask R-CNN, omawiany model słabo radzi sobie z segmentacją małych obiektów. Pozostałe wyniki można uznać za bardzo dobre, szczególnie w przypadku segmentacji klas: „pieszy” i „samochód”.

Tabela 16. Wyniki Average Precision sieci Cascade Mask R-CNN

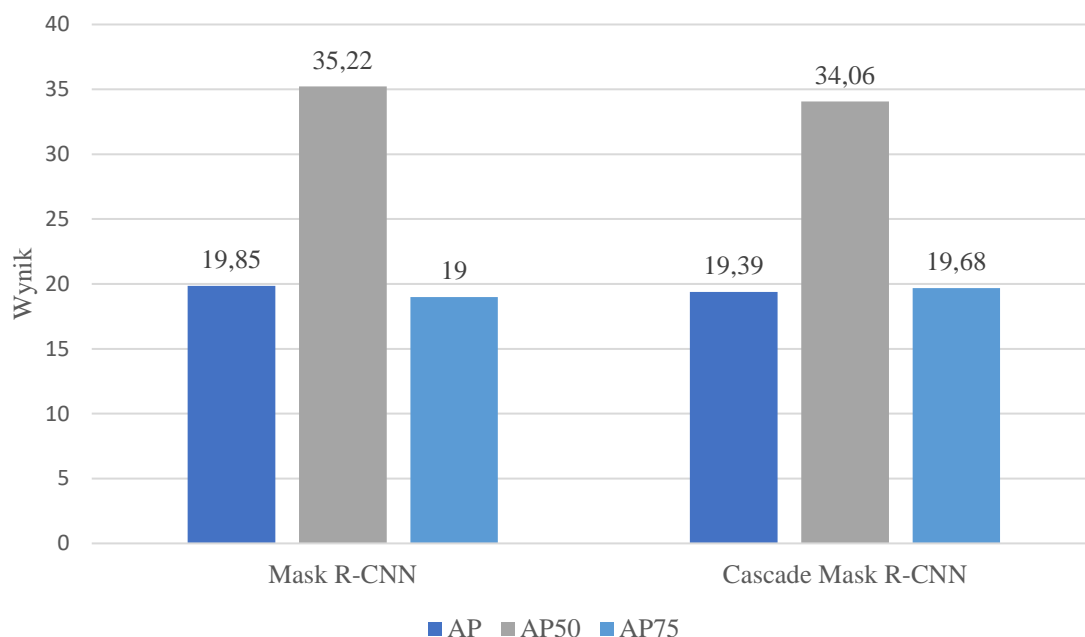
Miara Klasa	AP	AP50	AP75	APs	APm
pedestrian	28.25	55.56	26.37	17.15	47.93
rider	8.07	24.05	5.28	4.98	11.36
car	43.67	65.5	47.7	21.53	60.05
truck	26.35	36.39	30.95	8.89	24.37
bus	30.46	39.67	34.56	5.5	21.46
train	0	0	0	0	0
motorcycle	13.12	33.71	12.28	1.63	21.62
bicycle	5.2	17.6	0.35	4.78	7.72
Średnia	19.39	34.06	19.68	9.21	24.31

Wyniki miary AR dla sieci Cascade Mask R-CNN (Tabela 16) są mniejsze niż dla sieci Mask R-CNN. Model Cascade Mask R-CNN osiąga zadowalającą czułość w przypadku rozpoznawania klas: „samochód”, „ciężarówka”, „bus” i „pieszy”. Reszta klas jest ciężka do rozpoznania dla omawianego modelu.

Tabela 17. Wyniki Average Recall sieci Cascade R-CNN

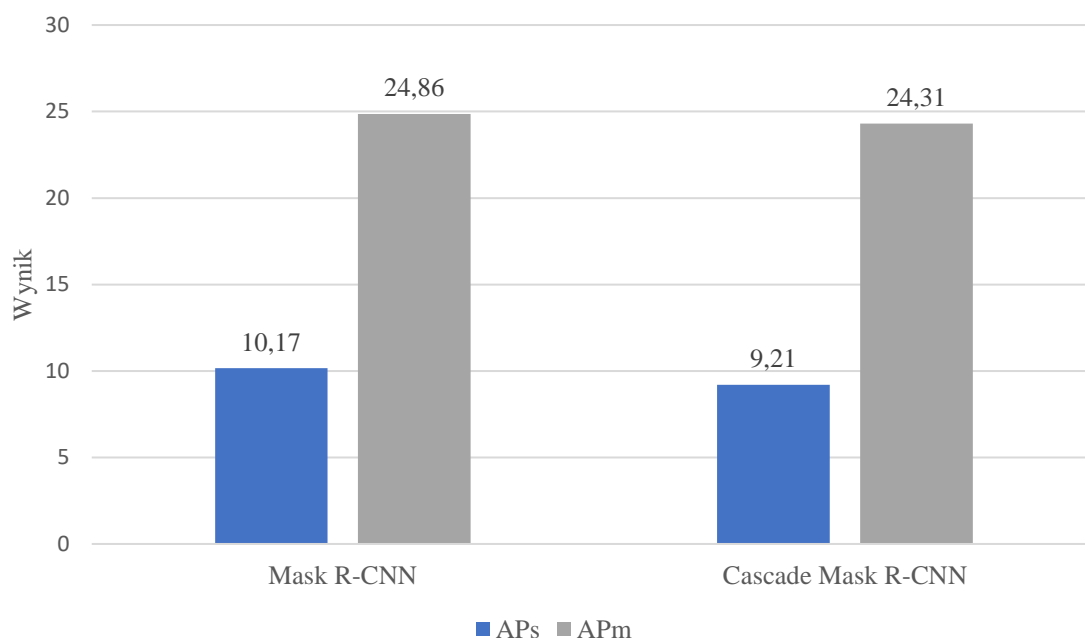
Miara Klasa	AR1	AR10	AR100	ARs	ARm
pedestrian	14.23	32.37	32.91	23.24	51.73
rider	10	10	10	6.67	12.31
car	7.34	43.74	49.08	27.89	66.67
truck	26.05	32.3	32.38	14.03	30.23
bus	30.32	34.84	34.84	6.99	27.11
train	0	0	0	0	0
motorcycle	14.9	15.53	15.53	3	25
bicycle	6.99	7.7	7.7	5.63	11.71
Średnia	13.73	22.06	22.8	12.5	28.09

Porównując wyniki (Rys. 27) miary AP w procesie segmentacji można wywnioskować, że obie sieci działają podobnie. Jednakże porównując jedynie miarę AP75, która wymaga dokładnych predykcji, model Cascade Mask R-CNN radzi sobie lepiej. Powodem tego może być wielokrotny proces segmentacji i jednocześnie dokładniejsze wyznaczanie szukanych obszarów na dużych obiektach. W przypadku analizy wyników miary AP i AP50 widoczne rezultaty mogą wyjaśniać różnice w budowie testowanych sieci. Cascade Mask R-CNN poprzez wielokrotny proces kodowania traci informację o położeniu obiektów, a tym samym sieć może ostatecznie słabo wykrywać małe obiekty.



Rys. 27. Wykres przedstawiający wyniki AP, AP50 i AP75 sieci użytych do segmentacji

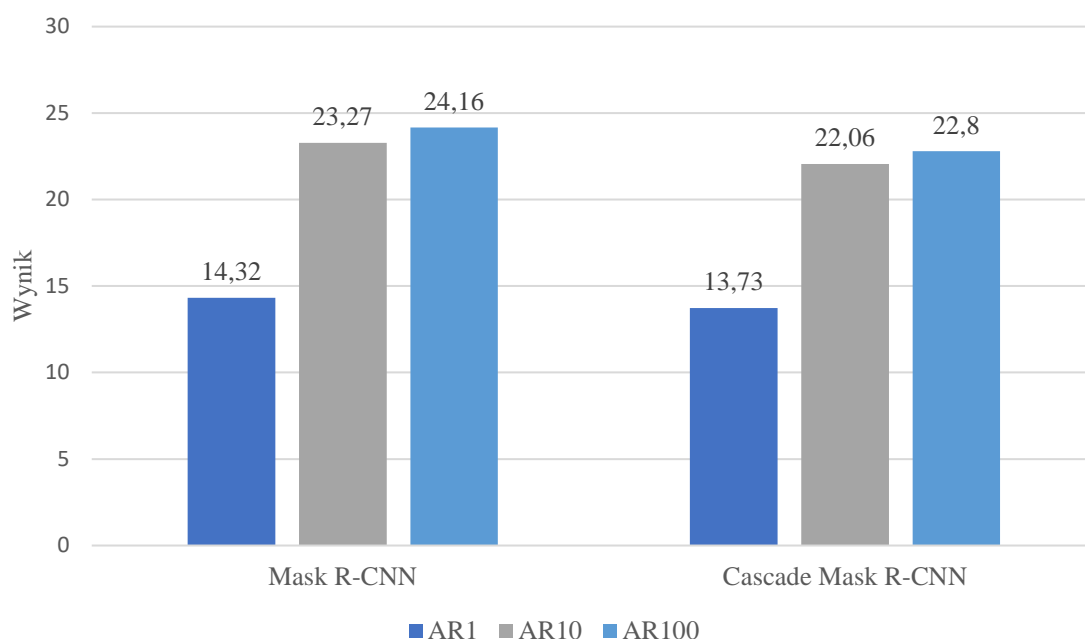
Jak widać na poniższym wykresie (Rys. 28), sieć Cascade Mask R-CNN radzi sobie gorzej z wykrywaniem małych i średnich obiektów. Jest to ponowne uzasadnienie wady, jaką posiada model wykorzystujący wielokrotny proces kodowania informacji, który prowadzi do utraty informacji o szczegółach.



Rys. 28. Wykres przedstawiający wyniki APs i APm sieci użytych do segmentacji

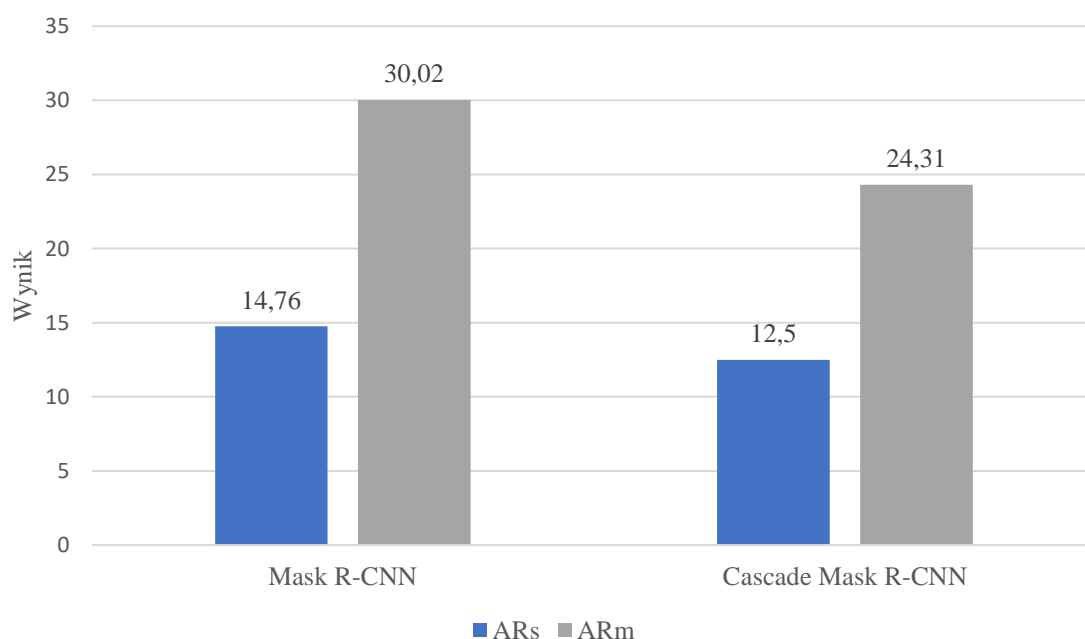
Poniższe porównanie (Rys.28) miar czułości pozwala utwierdzić się w przekonaniu, że prostsza sieć Mask R-CNN wypada lepiej w zadaniu segmentacji obiektów. Ponieważ sieć

Mask R-CNN posiada tylko jeden autoenkoder nie traci informacji o małych obiektach, dlatego jej predykcje zaznaczają większą ilość obiektów.



Rys. 29. Wykres przedstawiający wyniki AR1, AR10 i AR100 sieci użytych do segmentacji

Poniższy wykres (Rys.28) jest ostatecznym uzasadnieniem wcześniejszych wniosków. Sieć Cascade Mask R-CNN traci informację o małych i średnich obiektach, przez co ostatecznie wypada gorzej w zestawieniu. Sieć Mask R-CNN osiąga znacznie lepsze wyniki w procesie segmentacji.



Rys. 30. Wykres przedstawiający wyniki ARs i ARm sieci użytych do segmentacji

7. Wnioski

Sieci neuronowe są powszechnie stosowane w dziedzinie wizji komputerowej dopiero od niedawna, dlatego ważne jest ciągłe badanie ich możliwości w konkretnych, praktycznych zastosowaniach. W niniejszej pracy zrealizowane zostały poszczególne zadania potrzebne do pomiaru efektywności działania sieci neuronowych w rozpoznawaniu obiektów na jezdni. Począwszy od dokładnej analizy architektury i parametrów modeli, w pracy zaimplementowano pięć wybranych sieci neuronowych. Trzy modele zostały wytrenowane i przetestowane do zadania detekcji obiektów na jezdni, natomiast pozostałe dwa do segmentacji. W pracy dokonano analizy statystycznej zbioru danych, co pozwoliło na lepsze określenie jego charakterystyki. Wybrany zbiór danych przedstawia odpowiednie scenerie drogowe, takie jak: miasta, autostrady, tunele lub parkingi. Dzięki temu zbiór danych można określić jako odpowiedni do realizowanego zadania. Dane treningowe zostały przygotowane, tak aby zoptymalizować proces uczenia się sieci neuronowych. Uzasadniono dobór parametrów treningowych. W pracy udowodniono również adekwatność zastosowanych metryk do pomiaru wyników detekcji i segmentacji. Do ewaluacji uzyskanych wyników użyto miar uwzględniających wiele czynników przy pomiarze (precyzja, czułość, IOU i przedział ufności). Otrzymane rezultaty zostały przedstawione w tabelach, tak aby zaprezentować wyniki każdej z sieci. Następnie dokonano analizy i porównania wszystkich wyników względem siebie. Pozwoliło to określić, jaki model sieci radzi sobie najlepiej w wyznaczonym zadaniu. Podsumowując, sieć Cascade R-CNN osiągnęła najlepsze wyniki w detekcji, a sieć Mask R-CNN w segmentacji. Taki schemat pracy pozwolił na kompletne wdrożenie i zbadanie działania inteligentnego systemu wykrywania obiektów na jezdni. Każdy z wyznaczonych etapów docelowych został zrealizowany w wyczerpujący sposób.

Niniejsza praca stanowi solidny punkt wyjścia w dalszych badaniach nad możliwościami wykorzystania inteligentnych systemów wykrywania obiektów na jezdni. Należy jednak pamiętać, że tego typu systemy nie są jeszcze doskonałe i mogą mieć pewne ograniczenia, takie jak trudności z rozpoznawaniem obiektów w trudnych warunkach, np. mgła czy zła pogoda, a także z wykrywaniem obiektów małej wielkości. Dalsze badania i rozwój są konieczne, aby poprawić dokładność i niezawodność tych systemów. Obiecującym kierunkiem przyszłych badań może być próba połączenia wielu modeli sieci do jednoczesnego, równoległego działania, w celu uzyskania jeszcze lepszych rezultatów.

8. Literatura

- [1] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, R. Qu., "A survey of deep learning-based object detection." IEEE Access, 7:128837–128868, 2019, doi: 10.1109/ACCESS.2019.2939201.
- [2] "BDD100K A Diverse Driving Dataset for Heterogeneous Multitask Learning", <https://www.bdd100k.com> [Data uzyskania dostępu: 22 stycznia, 2023].
- [3] "BDD100K Documentation | Data Download", <https://doc.bdd100k.com/download.html> [Data uzyskania dostępu: 22 stycznia, 2023].
- [4] Y. Fisher, C. Haofeng, W. Xin, X. Wenqi, C. Yingying, L. Fangchen, V. Madhavan, T. Darrell "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 2633-2642, doi: 10.1109/CVPR42600.2020.00271.
- [5] F. Yu, V. Koltun and T. Funkhouser, "Dilated Residual Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 636-644, doi: 10.1109/CVPR.2017.75.
- [6] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 3213-3223, doi: 10.1109/CVPR.2016.350.
- [7] "Scalabel User Guide | Scalabel Format", <https://doc.scalabel.ai/format.html> [Data uzyskania dostępu: 22 stycznia, 2023].
- [8] "BDD100K Documentation | Label Format", <https://doc.bdd100k.com/format.html> [Data uzyskania dostępu: 22 stycznia, 2023].
- [9] „Python 3.11.1 documentation”, <https://docs.python.org/3> [Data uzyskania dostępu: 22 stycznia, 2023].
- [10] "Conda — conda documentation", <https://docs.conda.io/en/latest/> [Data uzyskania dostępu: 22 stycznia, 2023].
- [11] "Google Colab", <https://research.google.com/colaboratory/faq.html> [Data uzyskania dostępu: 22 stycznia, 2023].
- [12] "CUDA Toolkit 12.0 Downloads - NVIDIA Developer", <https://developer.nvidia.com/cuda-downloads> [Data uzyskania dostępu: 22 stycznia, 2023].
- [13] "PyTorch", <https://pytorch.org> [Data uzyskania dostępu: 22 stycznia, 2023].
- [14] "Welcome to MMDetection's Documentation! — MMDetection 2.27.0 documentation", <https://mmdetection.readthedocs.io/en/latest/> [Data uzyskania dostępu: 22 stycznia, 2023].
- [15] "Scalabel Website | A scalable open-source web annotation tool", <https://www.scalabel.ai> [Data uzyskania dostępu: 22 stycznia, 2023].
- [16] "SGD — PyTorch 1.13 documentation", <https://pytorch.org/docs/stable/generated/torch.optim.SGD.html#sgd> [Data uzyskania dostępu: 22 stycznia, 2023].

- [17] "Momentum and Learning Rate Adaptation", <https://www.willamette.edu/~gorr/classes/cs449/momrate.html> [Data uzyskania dostępu: 22 stycznia, 2023].
- [18] J. -a. Kim, J. -Y. Sung and S. -h. Park, "Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition," 2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), Seoul, Korea (South), 2020, pp. 1-4, doi: 10.1109/ICCE-Asia49877.2020.9277040.
- [19] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81.
- [20] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.
- [21] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.
- [22] "Github | SysCV/bdd100k-models", <https://github.com/SysCV/bdd100k-models> [Data uzyskania dostępu: 22 stycznia, 2023].
- [23] "fasterrcnn_resnet50_fpn — Torchvision main documentation", https://pytorch.org/vision/main/models/generated/torchvision.models.detection.fasterrcnn_resnet50_fpn.html#torchvision.models.detection.fasterrcnn_resnet50_fpn [Data uzyskania dostępu: 22 stycznia, 2023].
- [24] T. -Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 936-944, doi: 10.1109/CVPR.2017.106.
- [25] "Understanding Fast R-CNN and Faster R-CNN for Object Detection.", <https://towardsdatascience.com/understanding-fast-r-cnn-and-faster-r-cnn-for-object-detection-adbb55653d97> [Data uzyskania dostępu: 22 stycznia, 2023].
- [26] "Region Proposal Network (RPN) — Backbone of Faster R-CNN", <https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9> [Data uzyskania dostępu: 22 stycznia, 2023].
- [27] "ResNet-50: The Basics and a Quick Tutorial", <https://datagen.tech/guides/computer-vision/resnet-50/> [Data uzyskania dostępu: 22 stycznia, 2023].
- [28] "Softmax function – Wikipedia", https://en.wikipedia.org/wiki/Softmax_function [Data uzyskania dostępu: 22 stycznia, 2023].
- [29] Z. Tian, C. Shen, H. Chen and T. He, "FCOS: Fully Convolutional One-Stage Object Detection," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 9626-9635, doi: 10.1109/ICCV.2019.00972.
- [30] "FCOS- Anchor Free Object Detection Explained | LearnOpenCV", <https://learnopencv.com/fcos-anchor-free-object-detection-explained/> [Data uzyskania dostępu: 22 stycznia, 2023].

- [31] “Semantic Segmentation using Fully Convolutional Neural Networks | by Wilbur de Souza”, <https://medium.com/@wilburdes/semantic-segmentation-using-fully-convolutional-neural-networks-86e45336f99b> [Data uzyskania dostępu: 22 stycznia, 2023].
- [32] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. da Silva, “A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit,” *Electronics*, vol. 10, no. 3, p. 279, RJ, Brazil, Jan. 2021, doi: 10.3390/electronics10030279.

9. Spis rysunków

Rys. 1. Schemat przedstawiający różnicę między uczeniem maszynowym, a uczeniem głębokim. Proces możliwy do wytrenowania oznaczony jest przerywaną linią.	8
Rys. 2. Schemat przedstawiający architekturę sieci neuronowej.	9
Rys. 3. Schemat przedstawiający proces trenowania sieci neuronowej.....	10
Rys. 4. Przykład działania segmentacji semantycznej [2].....	12
Rys. 5. Przykład działania segmentacji instancji [2].....	13
Rys. 6. a) zdjęcie referencyjne, b) trenowane na „Cityscapes”, c) trenowane na „BDD100k”, d) adnotacja [4].....	16
Rys. 7. Przykładowe zdjęcia zbioru BDD100K oznaczonymi adnotacjami	17
Rys. 8. Sposób definiowania współrzędnych w zbiorze BDD100K [8].	18
Rys. 9. Porównanie optymalizatora SGD bez momentum oraz z momentum. Środek płaszczyzny przedstawia optymalne minimum funkcji kosztu [17].	22
Rys. 10. Schemat porównujący architekturę sieci R-CNN, Fast R-CNN i Faster R-CNN.....	24
Rys. 11. Schemat przedstawiający działanie zaimplementowanej sieci Faster R-CNN, gdzie „Mapa” to warstwa sieci neuronowej zakodowana jako mapa reprezentacji cech.	24
Rys. 12. Schemat przedstawiający działanie modułu FPN, gdzie W_x to kolejne warstwy splotowe, P_x to predykcje cech [24].	25
Rys. 13. Schemat przedstawiający zmniejszenie reprezentacji obrazu w FPN. Filtr (np. MaxPooling) koduje w kolejnej warstwie tylko najbardziej znaczące wartości pikseli, dzięki temu wyszczególnione zostają istotne cechy na obrazie [25].	25
Rys. 14. Schemat przedstawiający działanie przesuwne okienka w RPN. Po określeniu, czy na wybranym obszarze może znajdować się potencjalny obiekt, RPN generuje współrzędne regionu. Wymiary generowanych regionów są ogólnie określone [26].	26
Rys. 15. Schemat przedstawiający działanie sieci Cascade R-CNN, gdzie „Mapa” to warstwa sieci neuronowej zakodowana jako mapa reprezentacji cech.	27
Rys. 16. Poglądowy przykład działania autoenkodera. Od lewej: architektura enkodera („convolution network”), architektura dekodera („deconvolution network”) [31]......	29
Rys. 17. Schemat działania sieci Mask R-CNN.....	29
Rys. 18. Schemat działania sieci Mask R-CNN.....	30
Rys. 19. Zobrazowanie działania miary IOU	32
Rys. 20. Wzór równania precyzji omawiany w [32]......	33
Rys. 21. Wzór równania czułości omawiany w [32]......	33
Rys. 22. Wzór równania funkcji precyzji i funkcji czułości omawiany w [32].	34
Rys. 23. Wykres przedstawiający wyniki AP, AP50 i AP75 sieci użytych do detekcji	41
Rys. 24. Wykres przedstawiający wyniki APs i APm sieci użytych do detekcji.....	41
Rys. 25. Wykres przedstawiający wyniki AR1, AR10 i AR100 sieci użytych do detekcji	42
Rys. 26. Wykres przedstawiający wyniki ARs i ARm sieci użytych do detekcji.....	42
Rys. 27. Wykres przedstawiający wyniki AP, AP50 i AP75 sieci użytych do segmentacji....	46
Rys. 28. Wykres przedstawiający wyniki APs i APm sieci użytych do segmentacji	46
Rys. 29. Wykres przedstawiający wyniki AR1, AR10 i AR100 sieci do segmentacji	47
Rys. 30. Wykres przedstawiający wyniki ARs i ARm sieci użytych do segmentacji.....	47

10. Spis tabel

Tabela 1. Dystrybucja klas w zbiorze do detekcji obiektów	15
Tabela 2. Dystrybucja scenarii w zbiorze do detekcji obiektów	16
Tabela 3. Dystrybucja atrybutów pory dnia w zbiorze do detekcji obiektów	17
Tabela 4. Parametry modułu FPN w Faster R-CNN	26
Tabela 5. Parametry modułu RPN w Faster R-CNN.....	26
Tabela 6. Parametry modułu FPN w FCOS	28
Tabela 7. Parametry warstw splotowych w FCOS	28
Tabela 8. Wyniki Average Precision sieci Faster R-CNN	36
Tabela 9. Wyniki Average Recall sieci Faster R-CNN	37
Tabela 10. Wyniki Average Precision sieci Cascade R-CNN.....	38
Tabela 11. Wyniki Average Recall sieci Cascade R-CNN.....	38
Tabela 12. Wyniki Average Precision sieci FCOS.....	39
Tabela 13. Wyniki Average Recall sieci FCOS	40
Tabela 14. Wyniki Average Precision sieci Mask R-CNN	43
Tabela 15. Wyniki Average Recall sieci Mask R-CNN	44
Tabela 16. Wyniki Average Precision sieci Cascade Mask R-CNN.....	44
Tabela 17. Wyniki Average Recall sieci Cascade R-CNN	45