DETEKCJA ZAJĘTOŚCI MIEJSC PARKINGOWYCH Z WYKORZYSTANIEM YOLOV3

PREPRINT

Jakub Motyka
Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska
Warszawa
jakub.motyka2.stud@pw.edu.pl

Dawid Sygocki
Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska
Warszawa
dawid.sygocki.stud@pw.edu.pl

26 maja 2023

Słowa kluczowe Detekcja obiektów · YOLOv3 · Miejsca parkingowe · PANet

1 Wprowadzenie

Nasz projekt skupia się na wykrywaniu zajętości miejsc parkingowych. W klasycznej odsłonie ten powszechny problem wymagał dedykowanych rozwiązań takich jak bramki biletowe lub czujniki w miejscach parkowania. Nowym rozwiązaniem pojawiającym się na rynku jest wykorzystanie do analizy materiału kamer CCTV głębokich sieci neuronowych. Oznacza to oszczędność: kamery takie często są już i tak zainstalowane w celu zapewnienia bezpieczeństwa

Zdecydowaliśmy się na implementację sieci w architekturze YOLOv3 [Redmon and Farhadi, 2018]. Jest to sieć jednoprzebiegowa (tj. wykrywająca obiekty bez wcześniejszej predykcji obszarów zainteresowań) cechująca się dobrym stosunkiem jakości wyników do szybkości uczenia i działania.

2 Zbiór danych

Do uczenia i testowania modelu wykorzystaliśmy zbiór danych PKLot [De Almeida et al., 2015] – jeden z najpopularniejszych zbiorów etykietowanych zdjęć miejsc parkingowych.

Zbiór zawiera 12'416 obrazów w formacie JPEG o wymiarach 1280x720. Pochodzą one z trzech kamer, z czego dwie zamontowane były w obrębie tego samego parkingu, ale prezentują go z różnych perspektyw. Zdjęcia przedstawiają różne warunki oświetleniowe i atmosferyczne oraz, oczywiście, różny stopień zajętości miejsc.

W skład zbioru PKLot wchodzi łącznie 695'851 etykiet w formacie XML, z czego 48,5% to miejsca zajęte.

Liczba finalnie używanych obrazów to 12'142, co wynika z błędów w etykietach. Obrazy są odrzucane, jeżeli nie zawierają co najmniej jednej prawidłowej adnotacji. Etykiety przetwarzane są do formatu Pascal VOC, który opisuje każdy prostokąt jako współrzędne środka, szerokość, wysokość oraz klasę, a następnie są one zapisane pod tą samą nazwą co obraz, lecz z innym rozszerzeniem. Pozwala to na szybkie wczytanie obrazów zakwalifikowanych jako poprawne i uzyskanie z nich danych potrzebnych do uczenia.

Dane nie były podzielone na zbiór uczący, walidacyjny i testowy, dlatego dokonaliśmy takiego podziału samodzielnie w stosunku 8:1:1 za pomocą funkcji `random_split`. Obecny podział jest losowy.

3 Analiza literatury

Przegląd literatury pomógł nam w wyborze architektury sieci, zbioru danych oraz metryki pozwalającej określić jakość rozwiązania. Istotne z naszego punktu widzenia artykuły przedstawione są pokrótce poniżej:

- [Lisboa de Almeida et al., 2022] stanowi systematyczny przegląd zagadnienia detekcji miejsc parkingowych, podsumowując technologie i zbiory danych wykorzystywane w istniejących pracach oraz porównując otrzymane wyniki,
- [Ding and Yang, 2019] rozszerza sieć YOLOv3 o czwarty poziom skali cech (7x7 dla wejścia o rozmiarze 448x448); wykorzystany zbiór danych to podzbiór PKLot – PUCPR; ramki kotwiczne (*anchor boxes*) zostały dopasowane do zbioru danych za pomocą algorytmu centroidów,
- [Nguyen et al., 2023] porównuje wyniki różnych wariantów YOLOv5 na pełnym zbiorze PKLot, uzyskując mAP na poziomie 99,6-99,7%; w porównaniu wskazane są również inne prace wykorzystujące sieci YOLOv3, Faster R-CNN i Mask R-CNN.

3.1 Projekty wzorcowe

Implementując sieć YOLOv3, bazowaliśmy na następujących istniejących projektach:

- [Redmon and Farhadi, 2018] (oryginalna implementacja twórcy YOLOv3),
- [v iashin, 2023],
- [Lornatang, 2023].

4 Implementacja podstawowa (*baseline*)

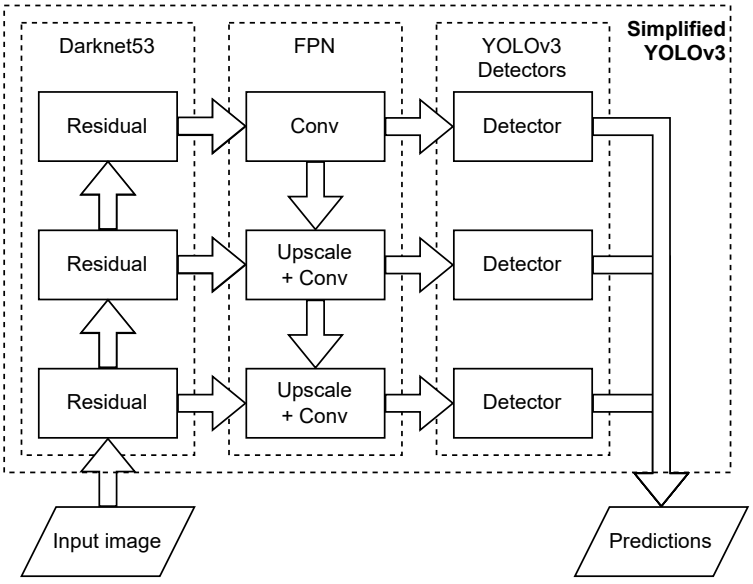
W ramach projektu dokonaliśmy implementacji standardowej sieci YOLOv3 dla wejścia o rozmiarze 416x416 z użyciem języka Python, bibliotek uczenia maszynowego: PyTorch, PyTorch Lightning i pomocniczych bibliotek do obliczeń i wizualizacji: NumPy, PIL, MatPlotLib.

Procedurą uczenia objęliśmy tylko szyjkę i głowice detekcyjne modelu. W kręgosłupie sieci wykorzystaliśmy wagi uzyskane w ramach treningu na zbiorze ImageNet (dostarczone przez autora YOLOv3 [Redmon and Farhadi, 2018]).

Zachowaliśmy zgodność wszelkich hiperparametrów z wzorcową optymalizacją.

4.1 Omówienie architektury

Bardzo uogólniony schemat YOLOv3 przedstawiony jest na rysunku 1. Model składa się z kręgosłupa (Darknet53), szyjki (FPN) oraz 3 głowic do detekcji (YOLOv3).



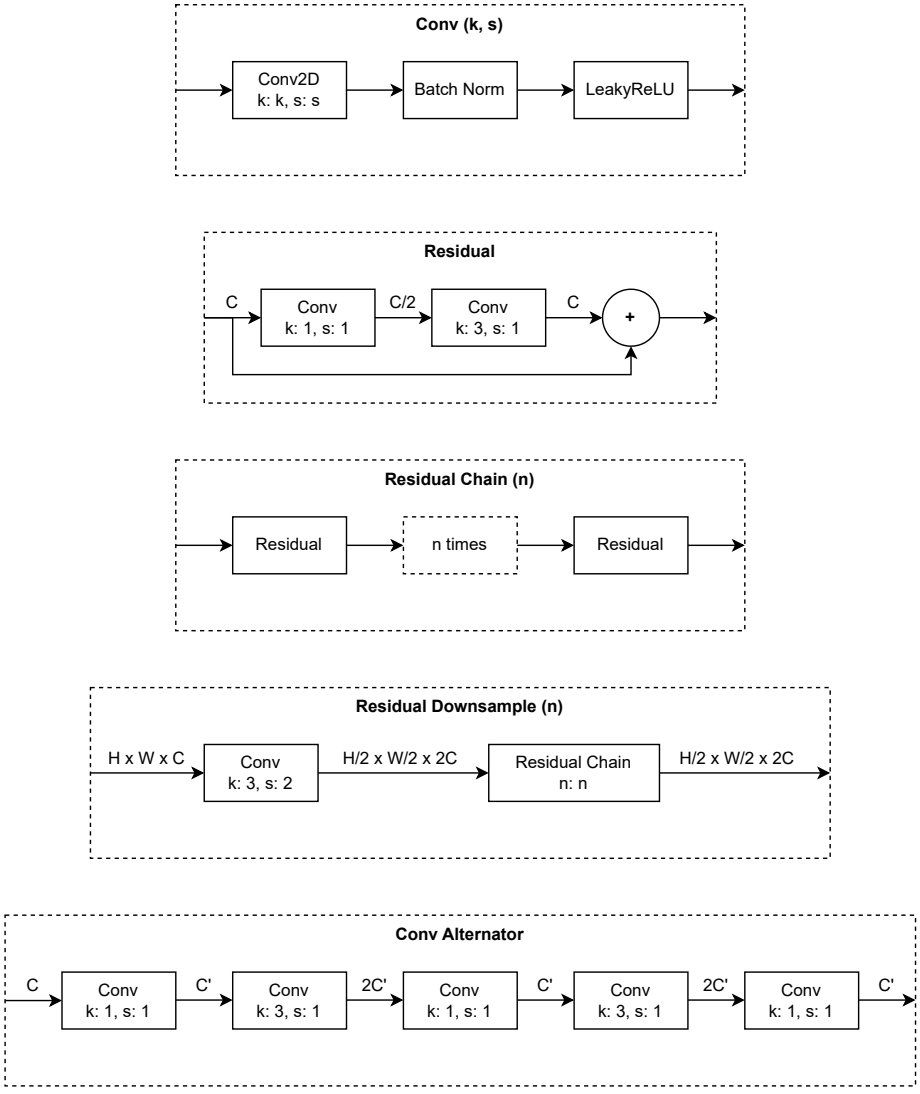
Rysunek 1: Bardzo uproszczony diagram architektury modelu detekcji obiektów YOLOv3

Warstwa kręgosłupa (rysunek 3) odpowiada za ekstrakcję cech z obrazu wejściowego. Korzystamy z 3 wyjść kręgosłupa, aby wykorzystać przydatne cechy w różnych rozdzielczościach, a więc i o różnym stopniu przetworzenia.

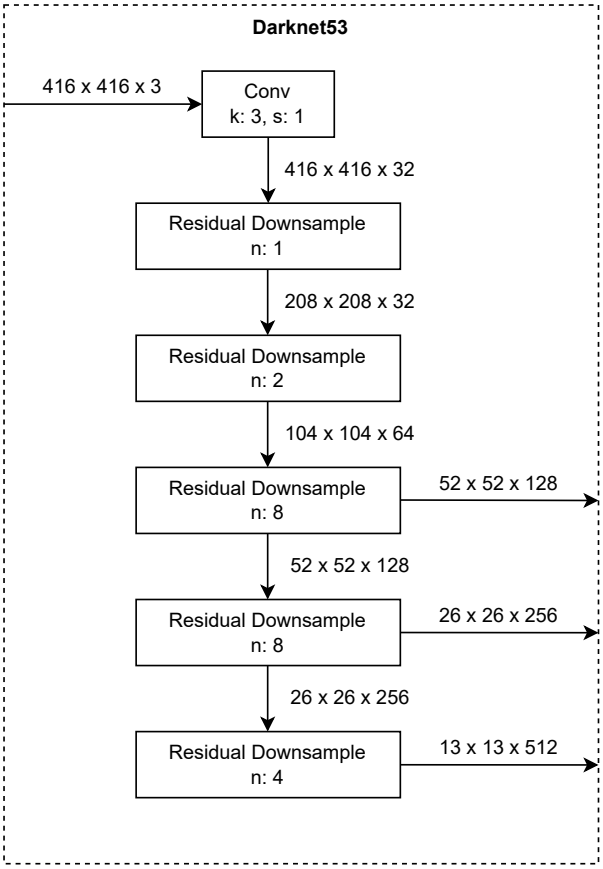
Szyjka modelu (rysunek 4) pozwala na propagację cech z późniejszych warstw do wcześniejszych poprzez skalowanie w górę cech o mniejszej rozdzielczości i ich konkatenacje z mniej bogatymi cechami o wyższej rozdzielczości. Finalnie cechy o 3 rozdzielczościach trafiają do 3 głowic detektora.

Głowice detektora (rysunek 5) generują predykcje. Każdy detektor dla każdego „piksela" (komórki) cech generuje 3 predykcje, odpowiadające 3 ramkom kotwicznym (dostarczonym dla każdej skali jako hiperparametr).

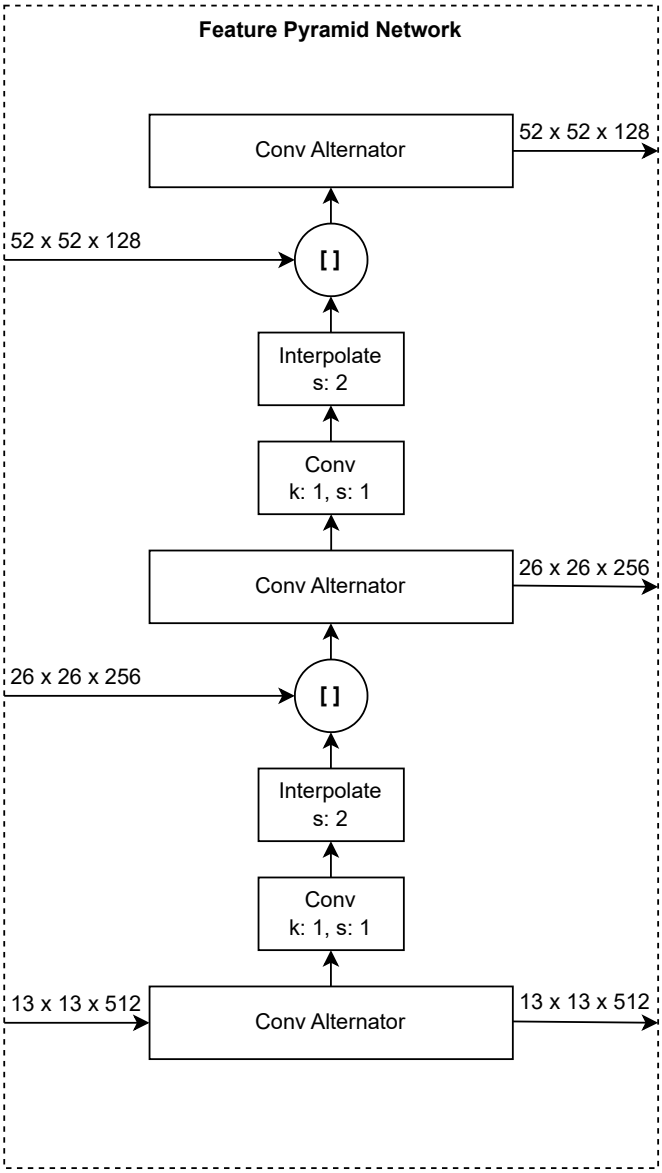
Całość modelu YOLOv3 podsumowana jest na rysunku 6. Szczegółowe diagramy wykorzystują bloki pomocnicze zdefiniowane na rysunku 2.



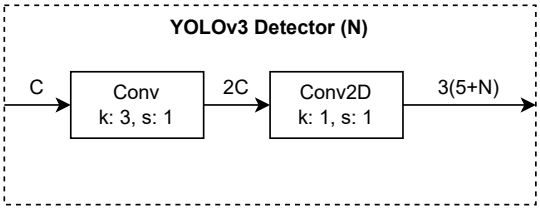
Rysunek 2: Bloki składowe modelu



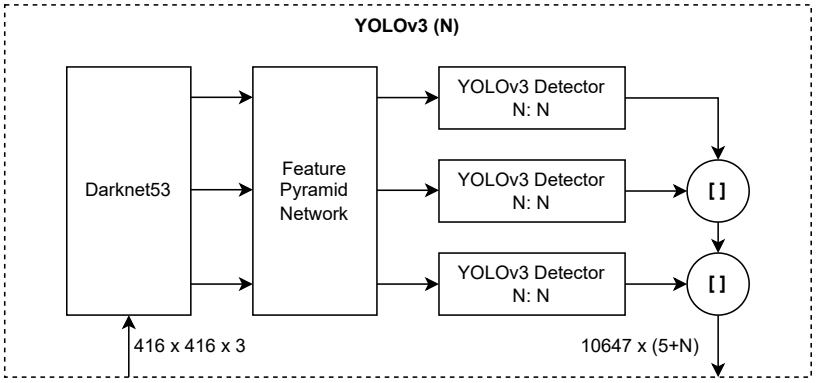
Rysunek 3: Diagram architektury kręgosłupa Darknet53 modelu YOLOv3



Rysunek 4: Diagram architektury szyjki FPN modelu YOLOv3



Rysunek 5: Diagram architektury głowicy detekcyjnej modelu YOLOv3



Rysunek 6: Diagram architektury modelu detekcji obiektów YOLOv3

4.2 Wyjście z modelu

Nieprzetworzone wyjście z modelu (predykcje detektora) składa się z trzech zestawów siatek o różnej skali detekcji. Każda komórka siatki zawiera trzy wektory (dla trzech różnych rozmiarów ramek kotwicznych) oznaczające wykryte ramki okalające. Zastosowanie kilku rozmiarów kotwic pozwala na wykrycie kilku obiektów o środku w zbliżonych punktach. W szczególności mogą być to obiektu o różnych proporcjach (np. jeden wydłużony w pionie, drugi w poziomie).

Pojedyncza ramka okalająca ma postać wektora o wymiarze $4 + 1 + N$, gdzie cztery pierwsze wymiary opisują pozycję środka ramki (x, y) oraz jej wymiary (szerokość, wysokość), piąty to stopień pewności co do obecności obiektu w danym miejscu (tzw. *objectness*), natomiast pozostałe N wymiarów opisuje pewność dla każdej z N klas.

4.2.1 Przetwarzanie wyjścia

Dla wybranego rozmiaru wejścia model YOLOv3 zwraca zawsze stałą liczbę ramek okalających: $(52 \cdot 52 + 26 \cdot 26 + 13 \cdot 13) \cdot 3 = 10647$. Aby możliwe było np. zaznaczenie na źródłowym obrazie wykrytych obiektów, konieczne jest pozbycie się nadmiarowych informacji.

Ramki ze wszystkich trzech skali łączone są w jedną listę, a następnie:

1. odsiewane są ramki o wartości *objectness* poniżej progu,
2. spośród nakładających się na siebie ramek wybierane są te o największej pewności (algorytm *Non-maximum Suppression*),
3. nakładające się na siebie ramki tej samej klasy są scalane ze sobą za pomocą średniej ważonej.

4.3 Funkcja straty

Funkcja straty YOLOv3 stanowi sumę następujących składników:

- błędu średniokwadratowego pozycji i rozmiarów ramek (tylko dla wykrytych obiektów – powyżej określonego progu *objectness*),
- binarnej entropii skrośnej stopnia pewności wykrycia obiektu,
- binarnej entropii skrośnej pewności poszczególnych klas (również tylko dla wykrytych obiektów).

Widać zatem, że bierze pod uwagę wszystkie trzy elementy nieprzetworzonego wyjścia z modelu.

Aby obliczyć wartość straty, konieczne jest dostarczenie wartości docelowych wygenerowanych na podstawie etykiet. W uproszczeniu jest to procedura odwrotna do przetwarzania wyjścia z modelu w procesie predykcji.

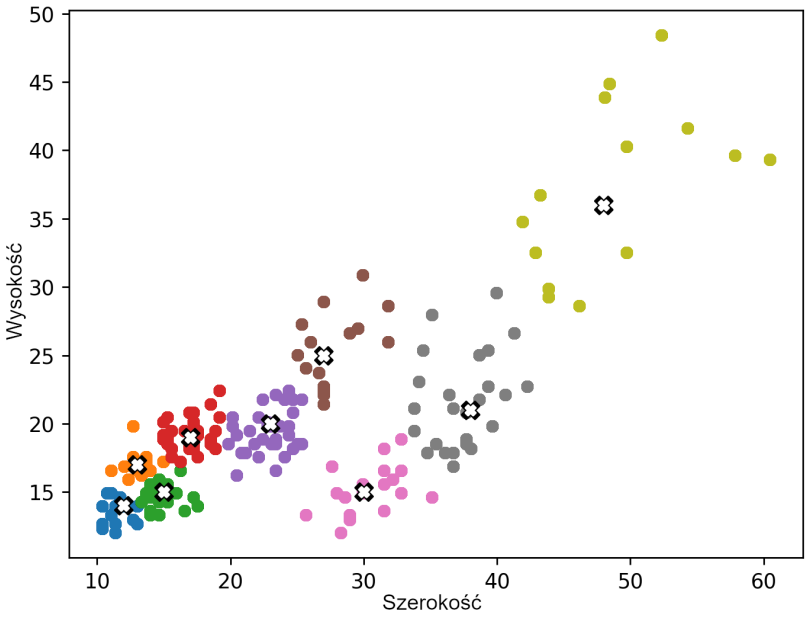
5 Usprawnienia

5.1 Dopasowanie *anchor boxów* do zbioru danych

Jedną z metod polepszenia jakości detekcji YOLOv3 jest dostosowanie wielkości ramek kotwicznych do danych, na których model działa. W tym celu należy przetworzyć rozmiary ramek okalających z etykiet używanego zbioru danych algorytmem k-średnich (dla $k = 9$). Uporządkowane punkty stanowią nowy zestaw kotwic dla modelu. Należy pamiętać, że najmniejsze *anchory* przeznaczone są dla detektora z największą rozdzielczością (który najlepiej wykrywa małe obiekty).

Wyznaczone przez nas *anchor boxy* dla zbioru PKLot (rysunek 7) wylistowane są poniżej:

1. 12x14, 13x17, 15x15,
2. 17x19, 23x20, 27x25,
3. 30x15, 38x21, 48x36.



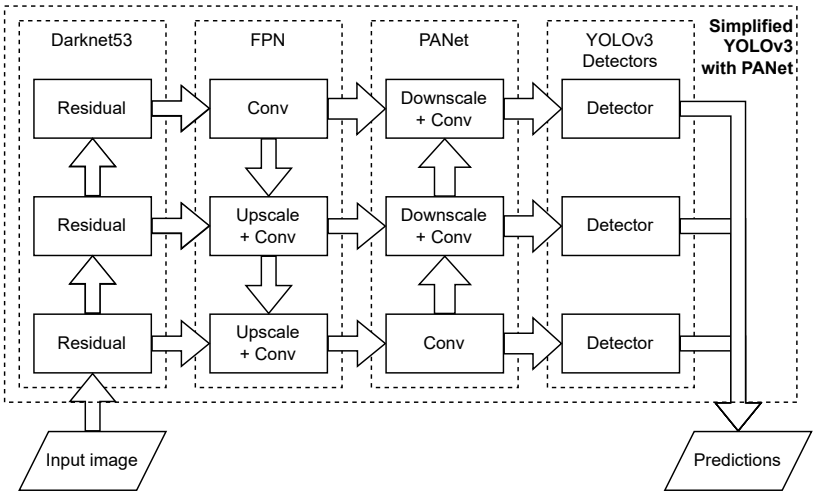
Rysunek 7: Wizualizacja wyniku klasteryzacji przeprowadzonej na rozmiarach ramek okalających zbioru PKLot

5.2 Szyjka PANet

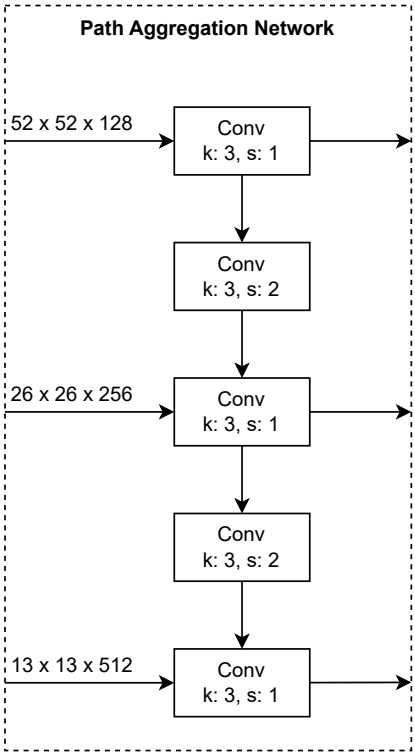
W celu poprawienia jakości sieci, szyjka modelu została wyposażona w dodatkowa warstwę *Path Aggregation Network* (PANet) pomiędzy oryginalną szyjką (FPN) a detektorami. Miała ona stworzyć tzw. skrót (*shortcut*), czyli potencjalnie krótszą ścieżkę pomiędzy oryginalnym obrazem a głowicą detektora, co pozwala na lepsze określenie położenia obiektu. Warstwa została stworzona na podstawie opisującej ją pracy [Liu et al., 2018]. Architektura warstwy przedstawiona jest na rysunku 9.

Podczas implementacji planowaliśmy sugerować się implementacją YOLOv4 opisaną w [Bochkovskiy et al., 2020]. Model okazał się być silnie dostosowany do problemu i momentami odbiegał od konwencji, dlatego zrezygnowaliśmy z tego pomysłu na rzecz obecnej, prostszej implementacji.

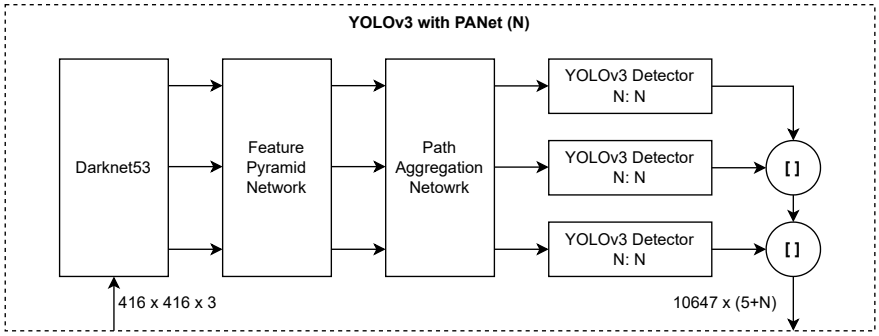
Finalna postać modelu zaprezentowana jest w formie uproszczonej na rysunku 8 i w formie pełnej na rysunku 10.



Rysunek 8: Bardzo uproszczony diagram architektury modelu detekcji obiektów YOLOv3 z dołożoną szyjką PANet



Rysunek 9: Diagram architektury szyjki PANet



Rysunek 10: Diagram architektury modelu detekcji obiektów YOLOv3 z dołożoną szyjką PANet

6 Wyniki

Tabela 1 prezentuje wyniki uzyskane dla:

1. podstawowej implementacji (model *Baseline*) trenowanej przez 85 epok,
2. implementacji z wyznaczonymi dla zbioru danych kotwicami (model *Anchors*) trenowanej przez 84 epoki,
3. implementacji z szyjką PANet (model *PANet*) trenowanej przez 57 epok (liczba epok była mniejsza ze względu na dłuższy czas uczenia większego modelu).

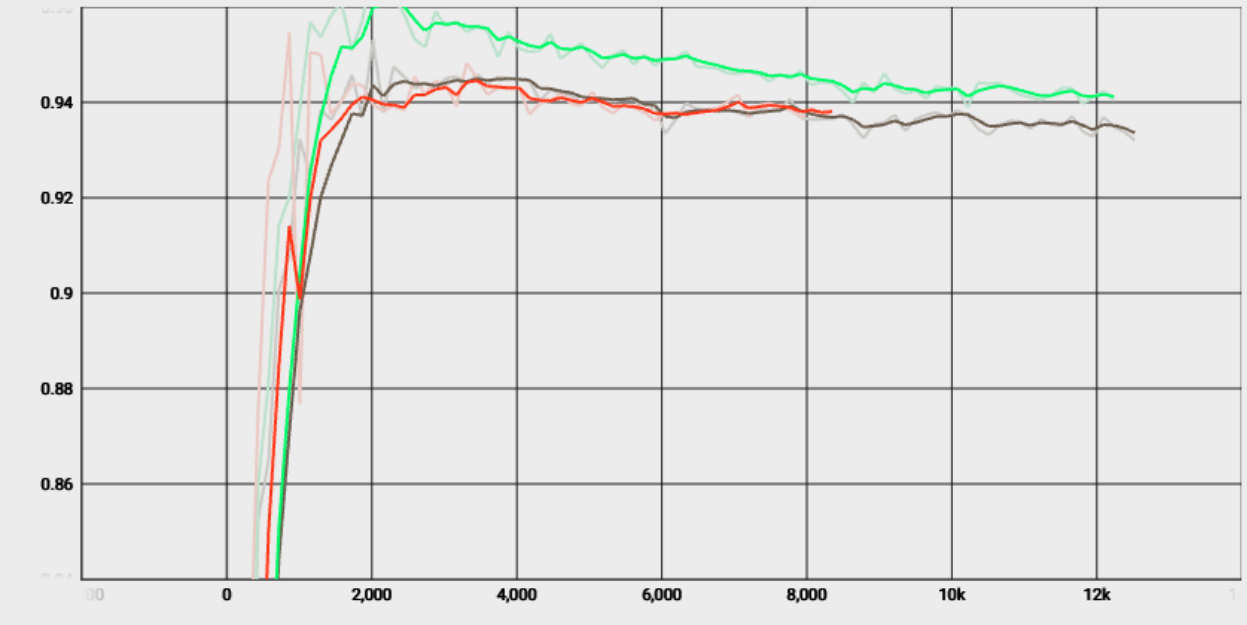
Tabela 1: Wyniki testów wyuczonego modelu YOLOv3

	Model		
Metryka	Baseline	Anchors	PANet
$mAP_{@0.5}$	0.94	0.94	0.94
$mAP_{@0.75}$	0.94	0.94	0.94
$mAP_{@0.5:0.95}$	0.84	0.85	0.85

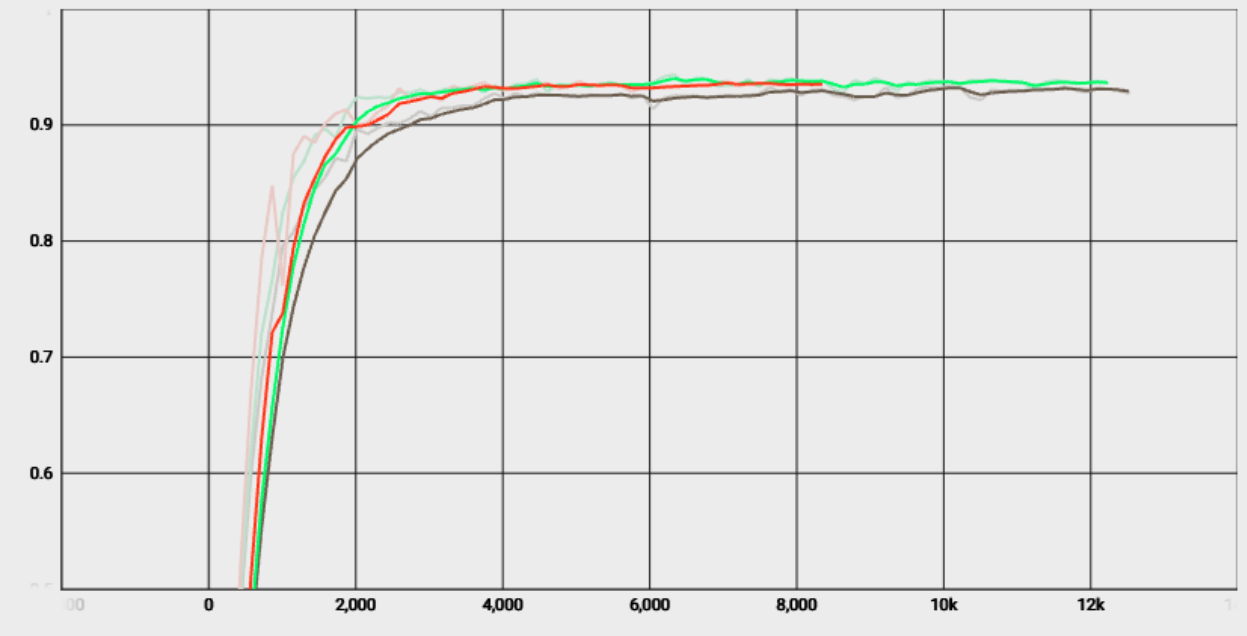
6.1 Porównanie modeli w czasie

Ponieważ finalne wyniki (po długotrwałym uczeniu) są zbliżone do siebie, poniżej przedstawiamy wykresy wartości mAP na przestrzeni kolejnych epok.

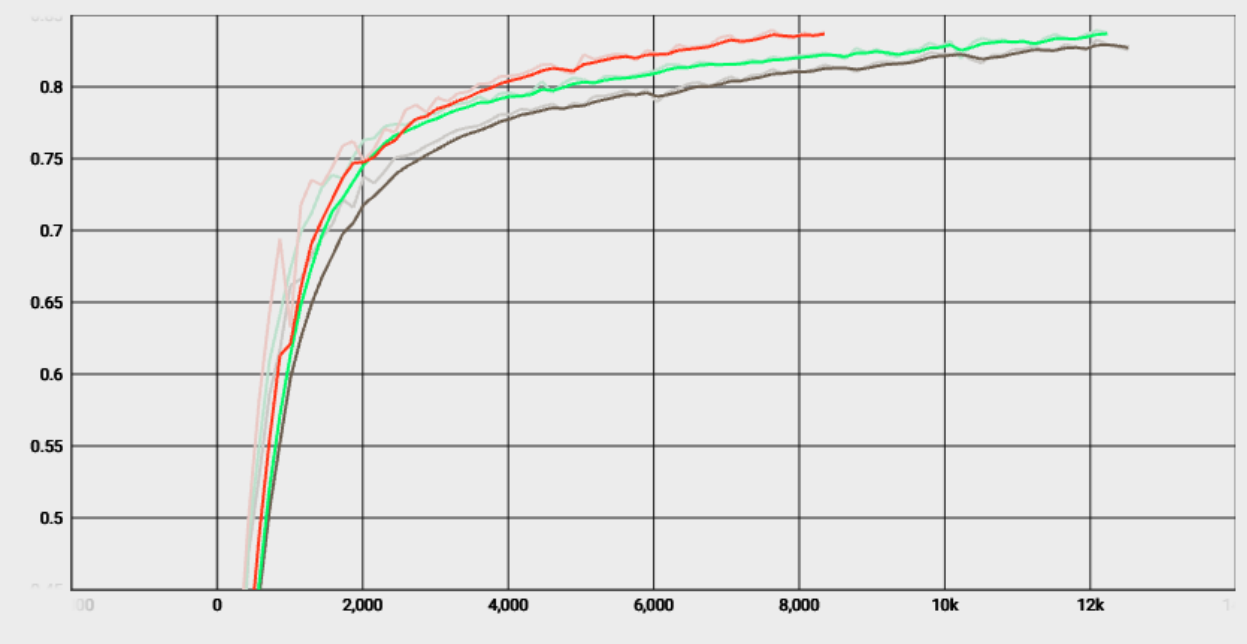
- Kolor szary - model *Baseline*
- Kolor zielony - model *Anchors*
- Kolor czerwony - model *PANet*



Rysunek 11: Wykres $mAP_{@0.5}$ od ilości obrazów uczących



Rysunek 12: Wykres $mAP_{@0.75}$ od ilości obrazów uczących



Rysunek 13: Wykres $mAP_{@0.5:0.95}$ od ilości obrazów uczących

6.2 Wnioski

- Wyniki są bardzo dobre. Niestety, jest duża szansa, że model się przeucza ze względu na niewielką różnorodność danych uczących.
- Dostosowanie rozmiarów *anchor boxów* do danych pozwoliło szybciej uzyskać wyższe wartości mAP, ale na dłuższą metę uzyskane wyniki były zbliżone do oryginalnych. Powodem może być wystarczająco dobre dopasowanie oryginalnych rozmiarów kotwic do naszych danych.

- Dodanie warstwy PANet spowolniło uczenie pojedynczej epoki modelu, ale znacznie polepszyło wyniki jakie osiąga. Już dla ok. 75% epok (w porównaniu do innych modeli) uzyskaliśmy wyniki porównywalne do oryginalnego modelu.
- Model *Anchors* ma największy przestrzał w $mAP_{@0.5}$ w okolicy 2000 obrazów uczących, a następnie utrzymuje się najwyżej.
- Model *PANet* uzyskuje najlepsze wyniki $mAP_{@0.5:0.95}$ zaczynając od 3000 obrazów uczących i stale utrzymuje się nad resztą. Ponieważ jest to powszechnie używana metryka, można powiedzieć, że model ten sprawdza się najlepiej ze wszystkich, które testowaliśmy.

Literatura

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

Paulo RL De Almeida, Luiz S Oliveira, Alceu S Britto Jr, Eunelson J Silva Jr, and Alessandro L Koerich. Pklot–a robust dataset for parking lot classification. *Expert Systems with Applications*, 42(11):4937–4949, 2015.

Paulo Ricardo Lisboa de Almeida, Jeovane Honório Alves, Rafael Stubs Parpinelli, and Jean Paul Barddal. A systematic review on computer vision-based parking lot management applied on public datasets. *arXiv e-prints*, pages arXiv–2203, 2022.

Xiangwu Ding and Ruidi Yang. Vehicle and parking space detection based on improved yolo network model. In *Journal of Physics: Conference Series*, volume 1325, page 012084. IOP Publishing, 2019.

Duy-Linh Nguyen, Xuan-Thuy Vo, Adri Priadana, and Kang-Hyun Jo. Yolo5pklot: A parking lot detection network based on improved yolov5 for smart parking management system. 2023.

v iashin. Websiteyolo: The back-end for the yolov3 object detector running as a webapp. Dostęp zdalny (26.05.2023): `https://github.com/v-iashin/WebsiteYOLO`, 2023.

Lornatang. Yolov3-pytorch: Pytorch implements yolov3.good performance, easy to use, fast speed. Dostęp zdalny (26.05.2023): `https://github.com/Lornatang/YOLOv3-PyTorch`, 2023.

Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation, 2018.

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.