RAPORT Z WYKONANIA PROJEKTU

Porównanie modeli detekcji obiektów

Jakub Wiercimak, Krzysztof Polak, Tymoteusz Widlarz

1. Początek pracy

Próby uruchomienia gotowych modeli z pochodzących z keras.io. Zbiorem danych, na którym trenowane były interesujące nas modele był ImageNET. Niestety ogromna liczba klas dostępnych w w/w zbiorze uniemożliwia znalezienie innego zbioru danych, który będzie posiadał każdą z nich. Z tego powodu porzuciliśmy Keras.

2. Wybór modeli do testowania

Użyliśmy modeli dostępnych na stronie tfhub [1]. Ze względu na charakter projektu (porównanie) wykorzystaliśmy modele wytrenowane na takim samym zbiorze danych (COCO). Wybrane przez nas modele to:

- ssd+mobilenet_v2 [2]
- Centernet+resnet [3]
- Centernet+hourglass [4]
- faster_rcnn+resnet [5]

Zrezygnowaliśmy z testowania YOLO, ponieważ chcieliśmy zachować jednolity format danych wychodzących.

3. Przygotowanie danych do testowania

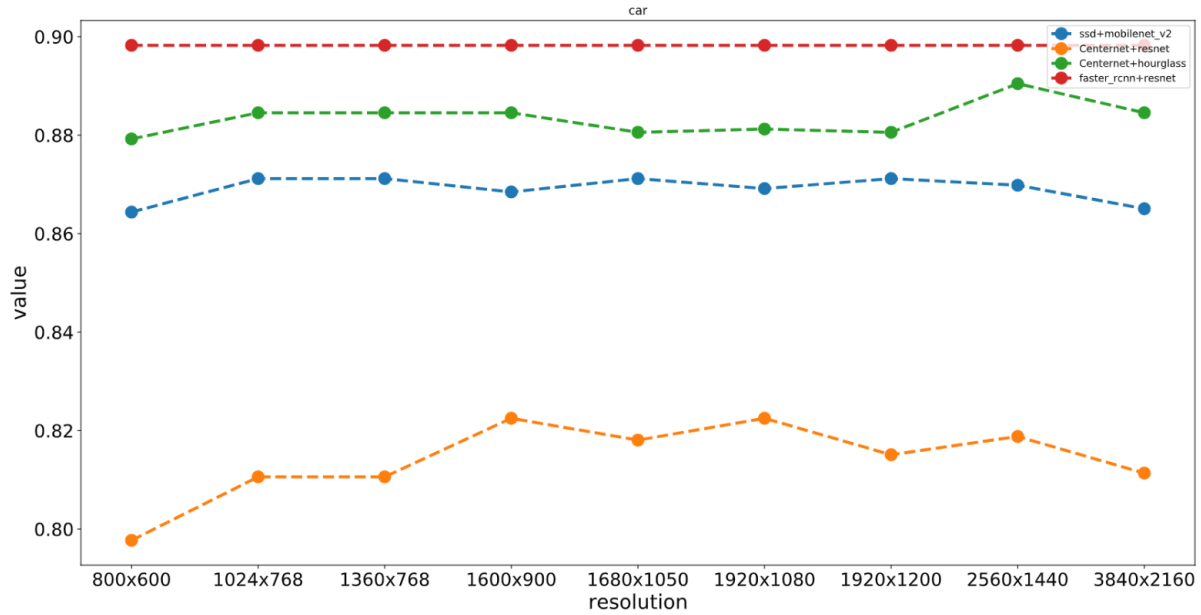
Znaleźliśmy duży zbiór danych Open Images [6]. Ręcznie sprawdziliśmy czy wszystkie klasy z COCO mają swoje odpowiedniki w w/w zbiorze. Niestety nie znaleźliśmy zdjęć dla klasy ‘hair brush’. Pobraliśmy zdjęcia dla każdej z klas korzystając z narzędzia pozwalające na wybór klasy zdjęć oraz ich ilości [7]. Następnie przygotowaliśmy mapowanie nazw pochodzących z Open Images do COCO. Utworzyliśmy metodę, która pozwala na zmianę rozdzielczości zdjęć zachowując stosunek wysokości do szerokości oryginalnego zdjęcia.

4. Testowanie modeli na przygotowanych danych

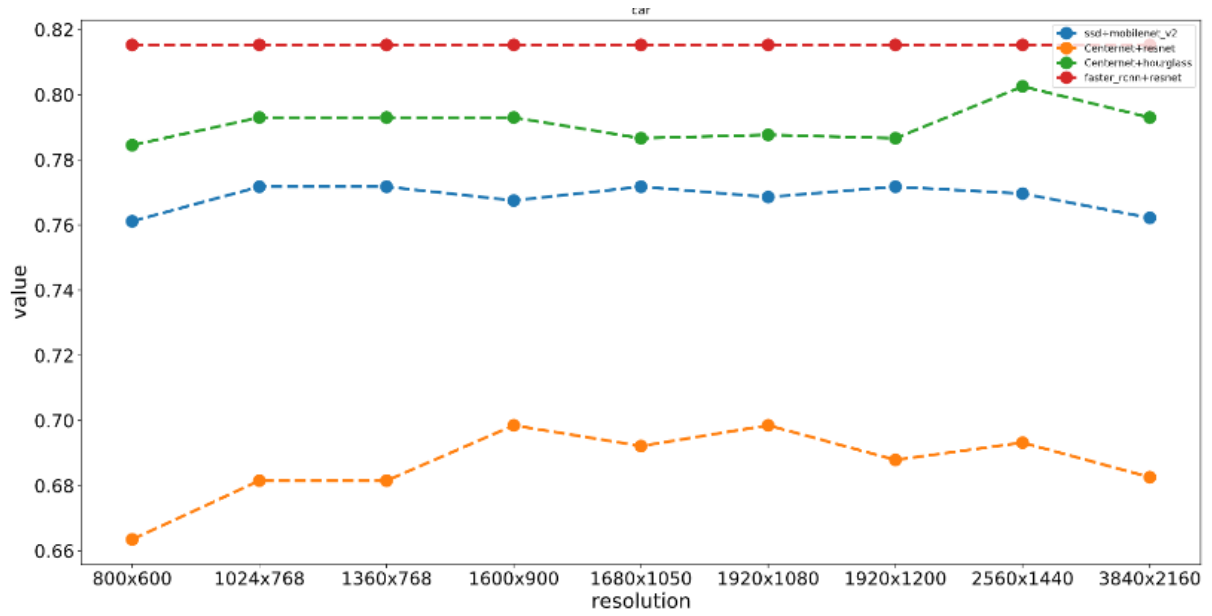
Stworzyliśmy funkcje które pozwalały na przetestowanie każdego z modeli na naszych danych. Dla każdego modelu funkcja ta zwracała macierze pomyłek dla każdej wybranej przez nas rodzielczości zdjęć. Testy obejmowały około 100 zdjęć na klasę. Klas było 90.

5. Analiza rezultatów

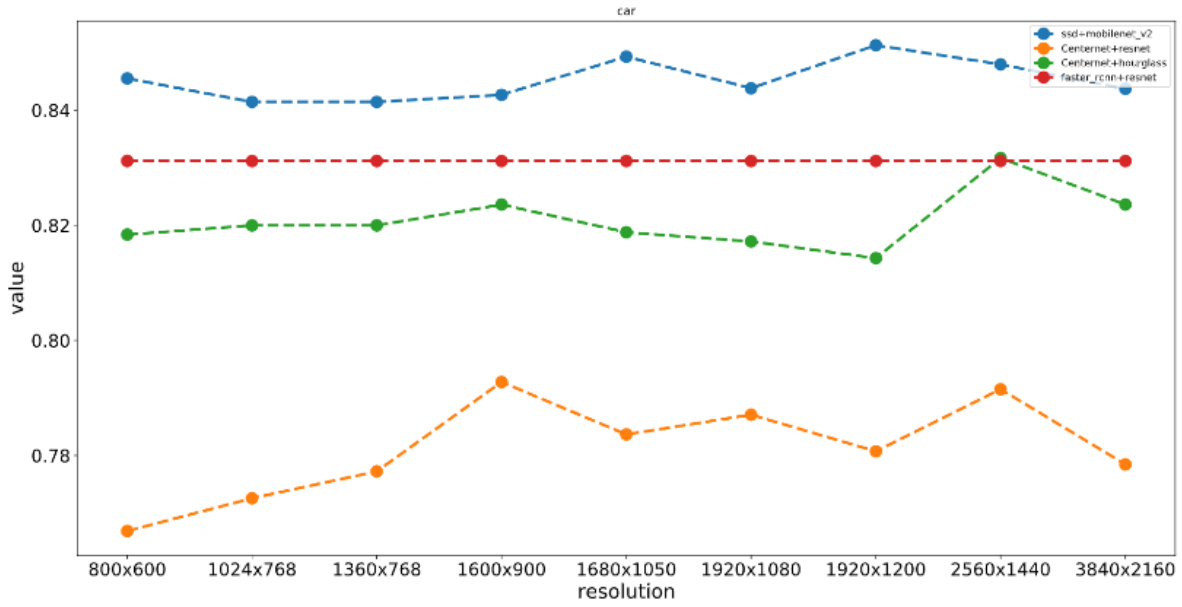
Na podstawie macierzy pomyłek wyliczyliśmy parametry takie jak F1 score, precision, recall, normalized score. Następnie przygotowaliśmy wykresy obrazujące w/w parametry dla danych klas i modeli w funkcji rozdzielczości oraz porównanie klas wewnątrz danego modelu przy danej rozdzielczości. Poniżej przykładowe wykresy. Pozostałe rezultaty dostępne są w repozytorium github dot. projektu [8].



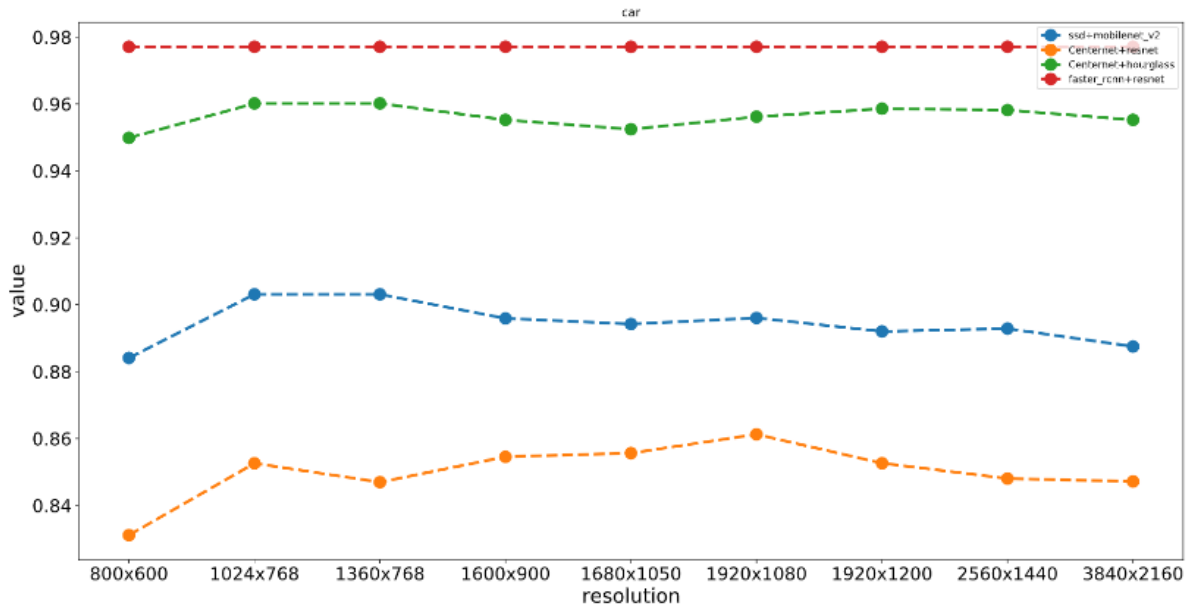
Rys 1. F1 Score



Rys 2. Normalized Score



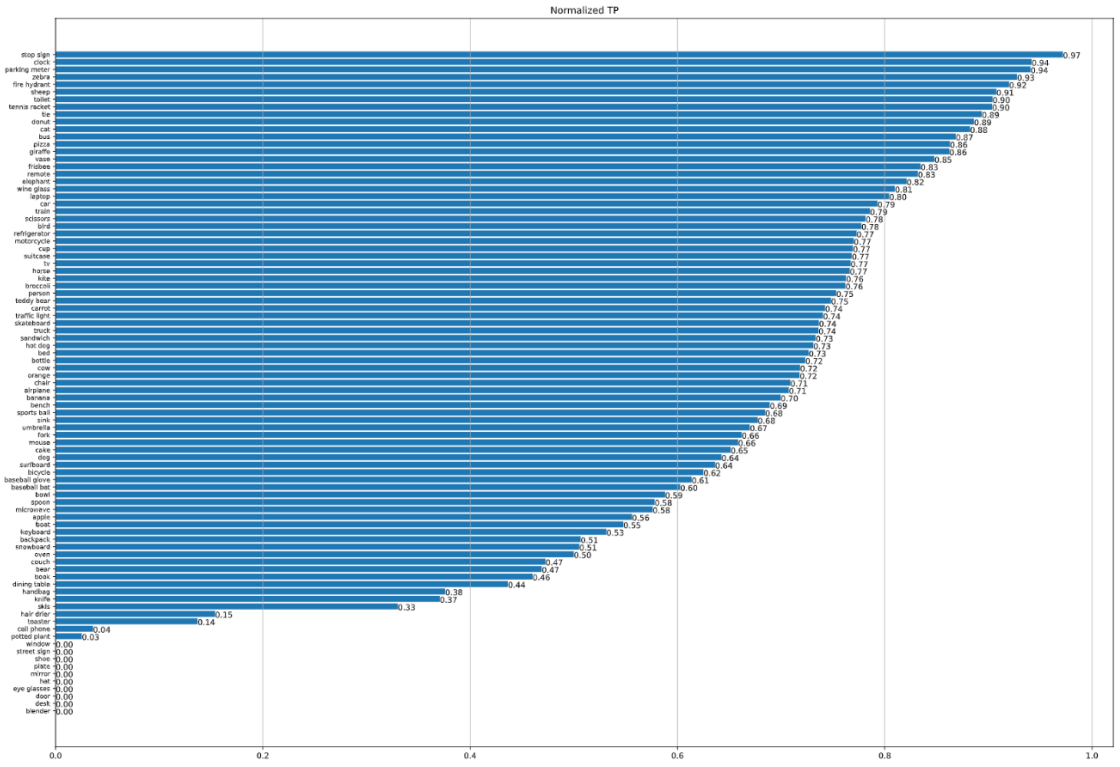
Rys. 3 Precision



Rys. 4 Recall

Wnioski dotyczące rys.1-4

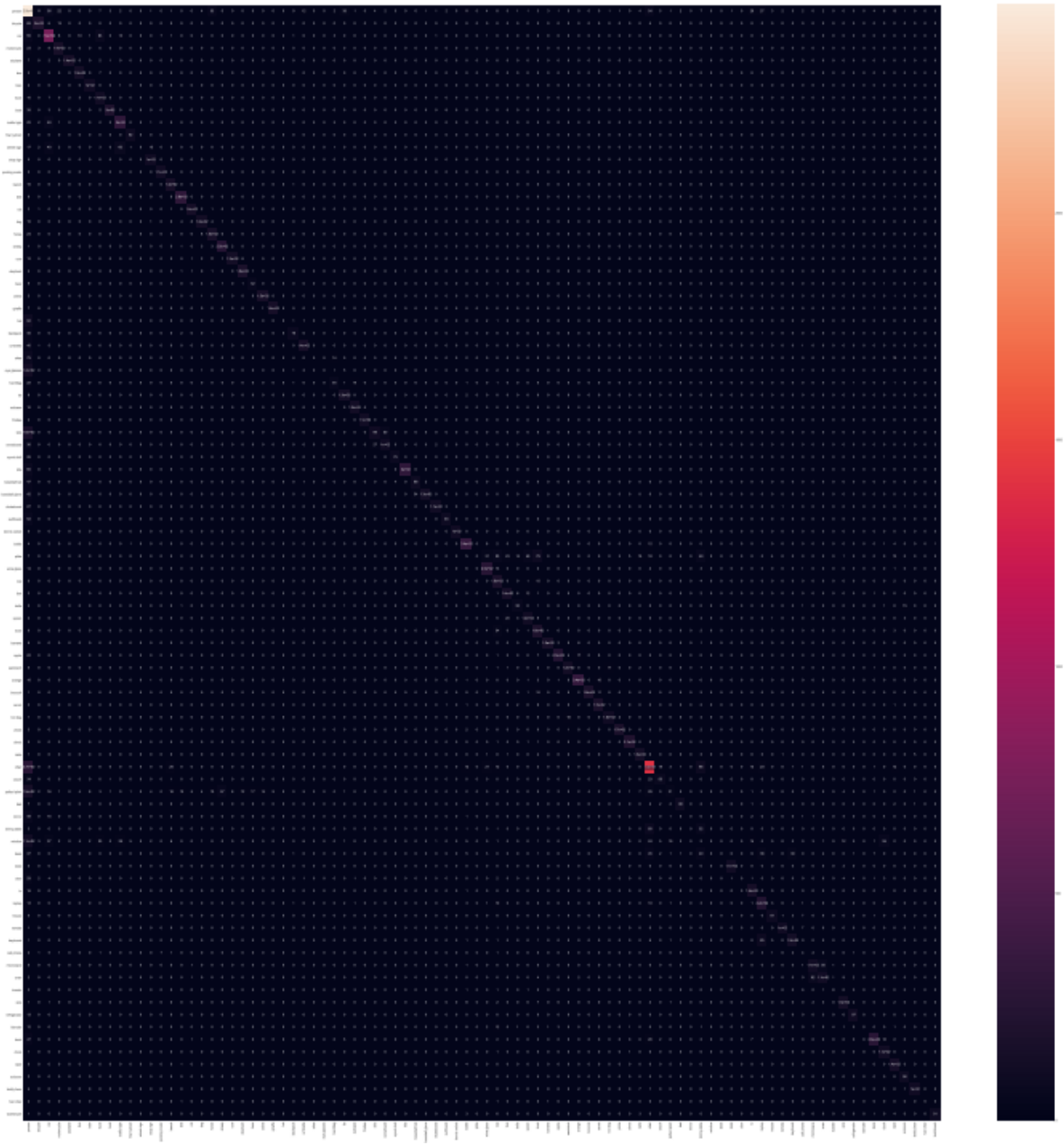
F1 Score, Normalized Score, Precision oraz Recall wskazują na nieznaczną poprawę rezultatów wraz ze wzrostem rozdzielczości. Faster_rcnn+resnet okazał się być niewrażliwy na zmianę rozdzielczości. Różnice były niewielkie. Może to wynikać ze sposobu zmiany rozdzielczości. Po analizie wykresów można porównać skuteczność modeli, od najlepszej do najgorszej: faster_rcnn+resnet, Centernet+hourglass, ssd+mobilnet_v2, centernet+resnet. Kolejność ta jest zgodna ze stopniem złożoności modelu oraz prędkością wykrywania obiektów. Wykresy dotyczące Precision wskazują na dominację ssd+mobilnet_v2 w tym zakresie. Pozostałe modele zachowują się zgodnie z resztą miar.



Rys. 5 Normalized score dla hourglass

Wnioski dotyczące rys. 5

Skuteczność wykrywania poszczególnych klas była bardzo podobna dla każdego z modeli. Zauważyliśmy że kilka klas w tym ‘window’, ‘mirror’ oraz ‘eye glasses’ nie były rozpoznawane przez żaden z testowanych modeli. Inne zaś takie jak stop sign są rozpoznawane przez każdy model z największą skutecznością. Może to wynikać z wykorzystanych danych jak i tego, że np. zamiast lustra rozpoznawane z większą wiarygodnością jest to co się w nim odbija. Obiekty słabo bądź nie wykrywane często występują również za innymi obiektami, przez co model nie jest w stanie ich rozróżnić. Natomiast znaki stop sign są zawsze bardzo podobne niezależnie od zbioru danych. W przeciwieństwie do bardziej ogólnej klasy ‘street sign’ czy luster.



Rys.6 Macierz pomyłek dla Hourglass+Centernet dla rozdzielczości 800x600

Wnioski dotyczące rys. 6

Wizualizacja macierzy w raporcie ze względu na liczbę klas jest mało czytelna. Lepszej jakości wizualizacja dostępna jest na GitHub [8]. Mapa termiczna wskazuje na przekątne co wynika z większości wykryć zgodnie z poszukiwaną klasą. Znacznie wybija się klasa ‘person’ ze względu na dużą liczbę obiektów tego typu w pozostałych zdjęciach (dotyczących innych klas). Warto zauważyć że wysoka liczba wykryć (na przekątnej) niekoniecznie musi się wiązać z dobrą dokładnością dla danej klasy. Przykładami są klasy ‘chair’ oraz ‘person’, które znajdują się na środku osi y wykresu Rys.5. Klasy te mają dużą liczbę wykryć oraz stosunkowo równie dużą liczbę pomyłek (krzesło jest brane za człowieka). Może to wynikać z danych z których korzystaliśmy, na krześle często mógł siedzieć człowiek. Idealną sytuacją jest gdy liczby większe od zera występują jedynie na przekątnej.

6. Przypisy

[1] https://tfhub.dev/s?module-type=image-object-detection
[2] https://tfhub.dev/tensorflow/ssd_mobilenet_v2/fpnlite_640x640/1
[3] https://tfhub.dev/tensorflow/centernet/resnet50v1_fpn_512x512/1
[4] https://tfhub.dev/tensorflow/centernet/hourglass_512x512/1
[5] https://tfhub.dev/tensorflow/faster_rcnn/resnet101_v1_640x640/1
[6] https://storage.googleapis.com/openimages/web/index.html
[7] https://github.com/EscVM/OIDv4_ToolKit
[8] https://github.com/tymekw/ML-project