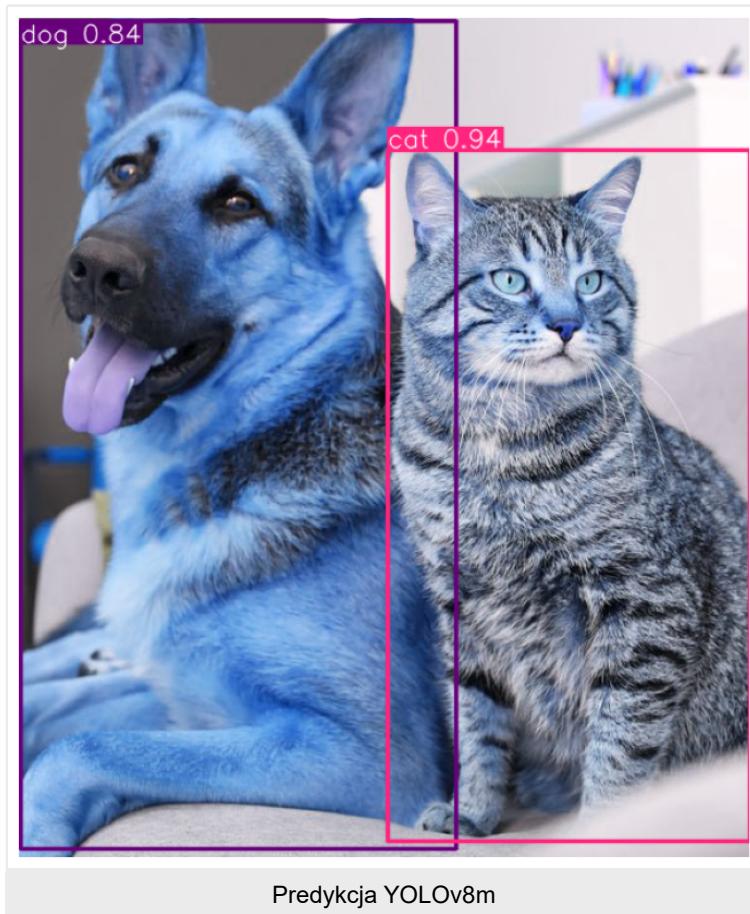


Zadanie 04

Krzysztof Dziechciarz

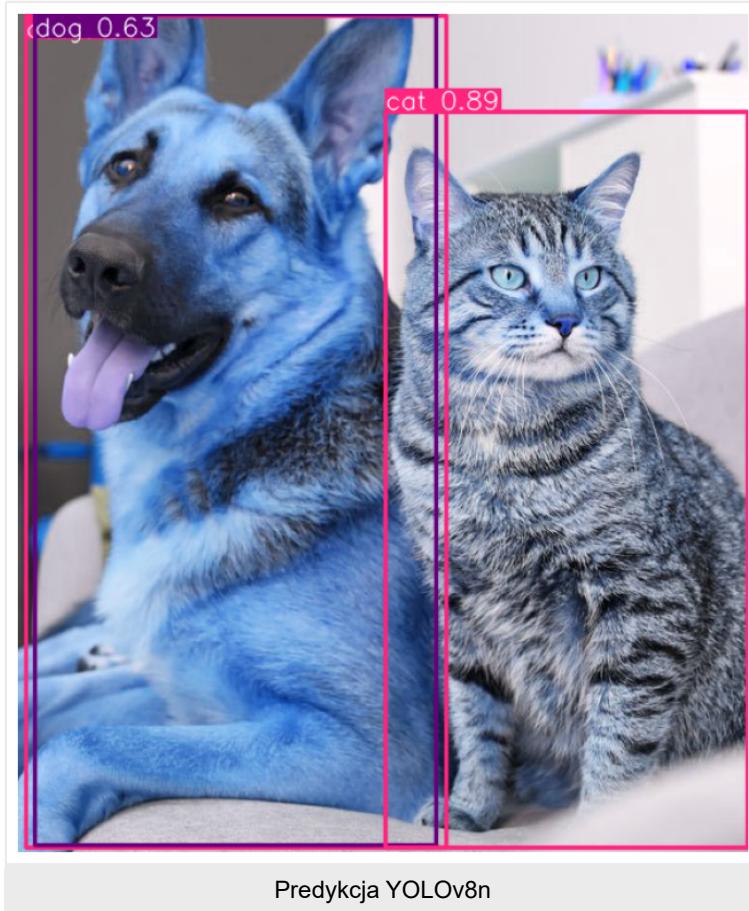
1. Test działania YOLOv8

Pobrałem i przetestowałem 2 rozmiary YOLOv8: yolov8m i yolov8n. Wersja medium dała następujące wyniki:



```
Object: cat, Confidence: 0.94, Coords: tensor([[291, 104, 577, 650]], dtype=torch.int32)
Object: dog, Confidence: 0.84, Coords: tensor([[ 1,   2, 345, 656]], dtype=torch.int32)
```

A wersja nano:



Predykcja YOLOv8n

```
Object: cat, Confidence: 0.89, Coords: tensor([[290, 77, 576, 659]]), dtype=torch.int32)
Object: dog, Confidence: 0.63, Coords: tensor([[ 13, 0, 330, 657]]), dtype=torch.int32)
Object: cat, Confidence: 0.56, Coords: tensor([[ 6, 1, 338, 659]]), dtype=torch.int32)
```

Zamiany kanałów wyświetlanych kolorów nie wykonywałem, bo nie jest to w tym przypadku istotne. Jak widać wersja medium radzi sobie bardzo dobrze, natomiast wersja nano psa załabelowała jednocześnie jako pies i kot z bardzo podobnym stopniem pewności.

2, 3 - Pobranie danych treningowych

Na początku korzystając z pliku `oidv7-class-descriptions-boxable.csv` odnalazłem ID klasy *Food*. Następnie korzystając z biblioteki pandas przefiltrowałem pliki `validation-annotations-bbox.csv` i `oidv6-train-annotations-bbox.csv`, tak aby otrzymać opisy obrazów zawierających interesującą mnie klasę. Na podstawie przefiltrowanych plików, stworzyłem pliki `.txt` zawierające opis bounding boxów zawierających klasę food w każdym z obrazów. Stworzyłem pliki `.txt` zawierające listę ID obrazów w odpowiednim formacie, zarówno dla zbioru treningowego jak i walidacyjnego. Udało się pobrać 3335 obrazów treningowych i 406 obrazów walidacyjnych.

4. Plik yaml

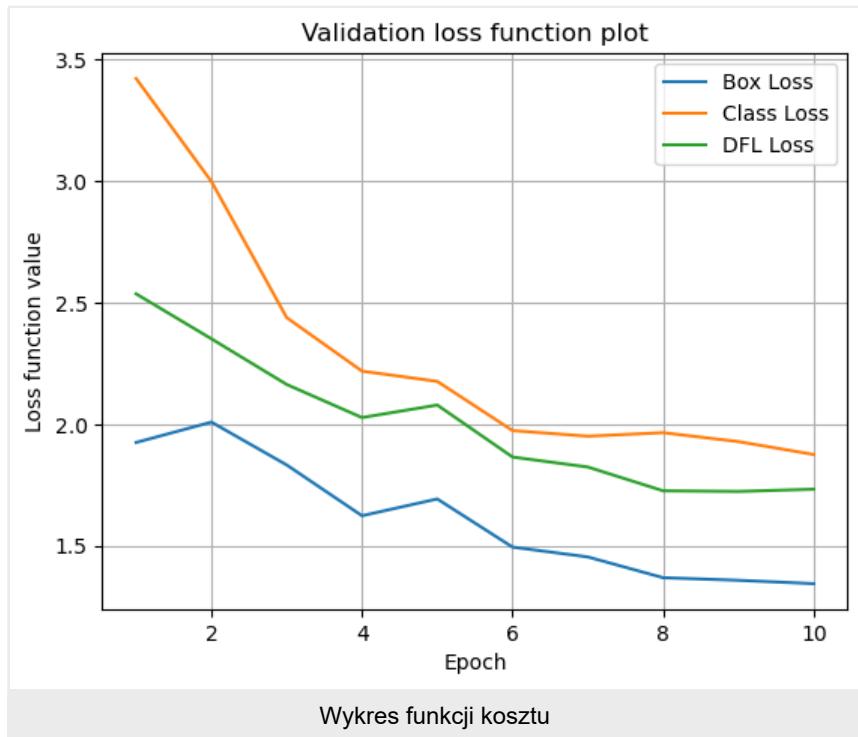
Plik `data.yaml` utworzyłem ręcznie, wzorując się na plikach dostępnych w dokumentacji YOLO:

```
path: D:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod
train: dataset/train/images
val: dataset/val/images
test:

names:
  0: Food
```

5. Trening

Wytrenowałem model na stworzonym zbiorze danych. Trening trwał 10 epok i na moim laptopie zajął niecałe 3h.



6. Testowanie modelu

Obrazy do testów pobrałem z grafiki google. Wyniki nie są niestety satysfakcyjne bo z 12 obrazów zawierających jedzenie, model znalazł je tylko na jednym.

```

image 1/12 d:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod\test_images\img_1.jpg: 448x640 (no detections),
217.5ms
image 2/12 d:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod\test_images\img_10.jpg: 384x640 (no detections),
275.2ms
image 3/12 d:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod\test_images\img_11.jpg: 448x640 (no detections),
283.0ms
image 4/12 d:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod\test_images\img_12.jpg: 480x640 (no detections),
356.0ms
image 5/12 d:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod\test_images\img_2.jpg: 384x640 (no detections),
209.0ms
image 6/12 d:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod\test_images\img_3.jpg: 640x448 (no detections),
130.3ms
image 7/12 d:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod\test_images\img_4.jpg: 640x480 (no detections),
106.5ms
image 8/12 d:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod\test_images\img_5.jpg: 384x640 (no detections),
83.7ms
image 9/12 d:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod\test_images\img_6.jpg: 320x640 1 Food, 77.9ms
image 10/12 d:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod\test_images\img_7.jpg: 448x640 (no detections),
86.6ms
image 11/12 d:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod\test_images\img_8.jpg: 448x640 (no detections),
69.4ms
image 12/12 d:\__STUDIA__\__INFA__\7_sem\MRO\lab04\kod\test_images\img_9.jpg: 448x640 (no detections),
79.9ms
Speed: 3.6ms preprocess, 164.6ms inference, 0.5ms postprocess per image at shape (1, 3, 448, 640)

```

Przykładowe obrazy:



7. Cenzura

Aby zblurować wykryte jedzenie, wyciągam z predykcji dane o bounding-boxach i w ich miejscu używam filtra GaussianBlur z biblioteki Python Imaging Library:

