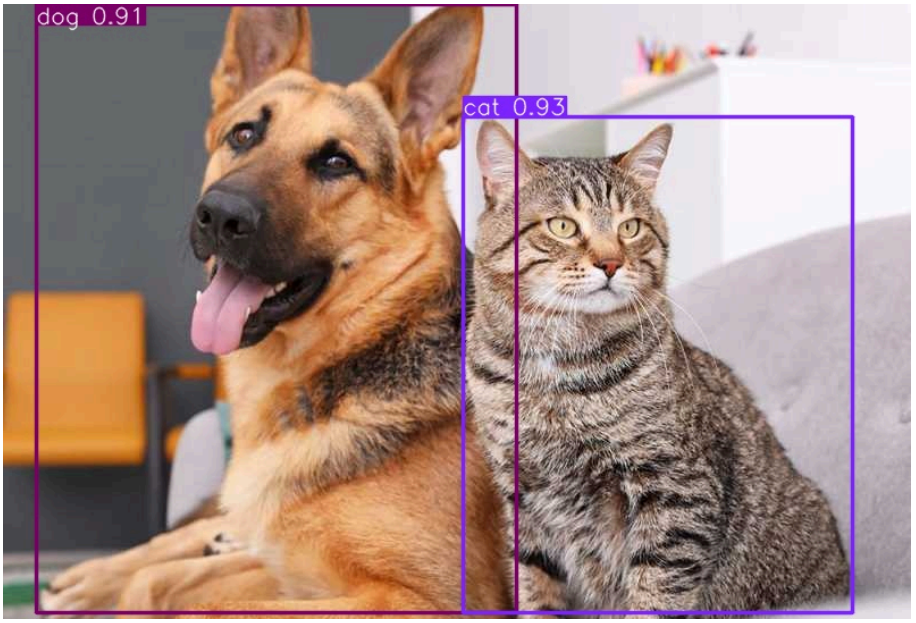
Zadanie 4.

Zadanie wykonałem korzystając z biblioteki ultralytics dostarczającej proste API to korzystania z różnych wersji modelu YOLO. Po zainstalowaniu biblioteki, przetestowałem działanie modelu YOLOv8m na obrazie przedstawiającym kota i psa. Uzyskany wynik zamieściłem poniżej.

```
from ultralytics import YOLO
# Make sure that the model works correctly
YOLO("yolov8m.pt").predict(source="./catdog.jpg", save=True, conf=0.5)
```



Następnie pobrałem zbiór danych treningowych jako podzbiór obrazów z kategoriami “Food” ze zbioru Open Images Dataset V7. Najpierw pobrałem pliki CSV zawierające opisy klas oraz listę zdjęć zawierającą ID, bounding box oraz prawidłową klasę.

```
os.system(
"wget https://storage.googleapis.com/openimages/v6/oidv6-train-annotations-bbox.csv")
os.system(
"wget https://storage.googleapis.com/openimages/v5/validation-annotations-bbox.csv")
os.system(
"wget https://storage.googleapis.com/openimages/v7/oidv7-class-descriptions-boxable.csv")
```

Następnie korzystając z biblioteki Pandas utworzyłem pliki tekstowe zawierające listę ImageID dla obrazów z klasy “Food”, w formacie akceptowanym przez skrypt downloader.py. Utworzyłem osobne pliki dla zbioru treningowego i walidacyjnego. Dodatkowo stworzyłem odpowiednią strukturę katalogów przedstawioną poniżej oraz pliki tekstowe zawierające dla każdego zdjęcia znajdujące się na nim bounding boxy zawierające obiekt z kategorii “Food”.

Poniżej zamieściłem utworzoną strukturę katalogów odpowiednią dla późniejszego fine-tuningu modelu YOLO oraz fragment kodu tworzący listę zdjęć do pobrania i zapisujący pliki tekstowe zawierające prawidłowe etykiety tych zdjęć. Analogicznego kodu użyłem dla zbioru walidacyjnego.

```
./datasets/oiv7/
              images/
                     train/
                     valid/
              labels/
                     train/
                     valid/
```

```
labels_map = pd.read_csv("oidv7-class-descriptions-boxable.csv")
labels = labels_map.loc[labels_map["DisplayName"].isin(["Food"]), "LabelName"]

df_train = pd.read_csv("oidv6-train-annotations-bbox.csv")
df_train = df_train.loc[df_train["LabelName"].isin(labels)]

with open("image_list_train.txt", "w") as f:
    f.write("")
    for i in df_train.index:
        f.write(f"train/{df_train.loc[i, 'ImageID']}\n")

for i in df_train.index:
    xmin, xmax = df_train.loc[i, "XMin"], df_train.loc[i, "XMax"]
    ymin, ymax = df_train.loc[i, "YMin"], df_train.loc[i, "YMax"]
    xc, yc, w, h = (xmin + xmax) / 2, (ymin + ymax) / 2, xmax - xmin, ymax - ymin
    with open(f'./datasets/oiv7/labels/train/{df_train.loc[i, "ImageID"]}.txt', "a") as f:
        f.write(f"0 {xc} {yc} {w} {h}\n")
```

Skrypt dowloader.py uruchomiłem z zamieszczonymi poniżej parametrami.

```
python3 downloader.py "image_list_train.txt" --download_folder="./datasets/oiv7/images/train/"
python3 downloader.py "image_list_valid.txt" --download_folder="./datasets/oiv7/images/valid/"
```

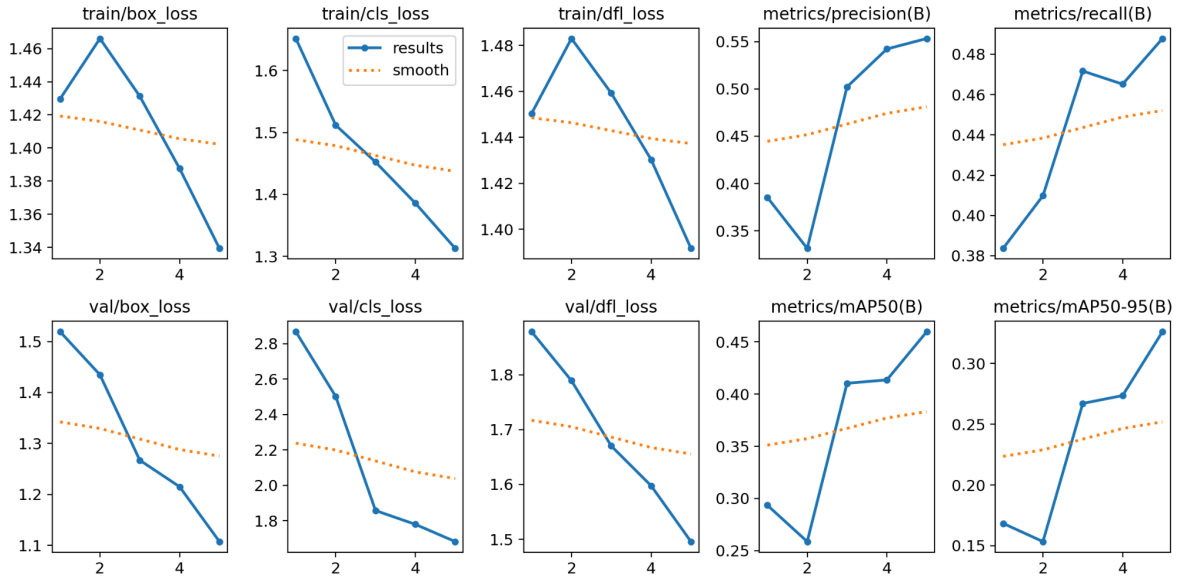
Po trwającym około 3 godzin pobieraniu ostateczny zbiór danych składał się z 25,712 zdjęć jedzenia w zbiorze treningowym oraz 1,696 zdjęć w zbiorze walidacyjnym. Następnie ręcznie stworzyłem plik dataset.yaml potrzebny do treningu, którego treść zamieściłem poniżej.

```
path: /home/barto/Projects/image-recognition-lab/lab4/datasets/oiv7
train: images/train
val: images/valid
test: # test images (optional)

names:
    0: food
```

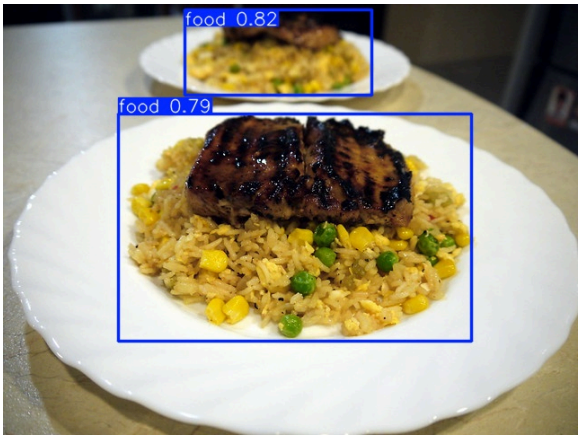
Do fine-tunowania wybrałem model YOLOv8n, który trenowałem przez 5 epok. Poniżej zamieściłem fragment kodu, który wywołuje funkcję train udostępnioną przez framework ultralytics.

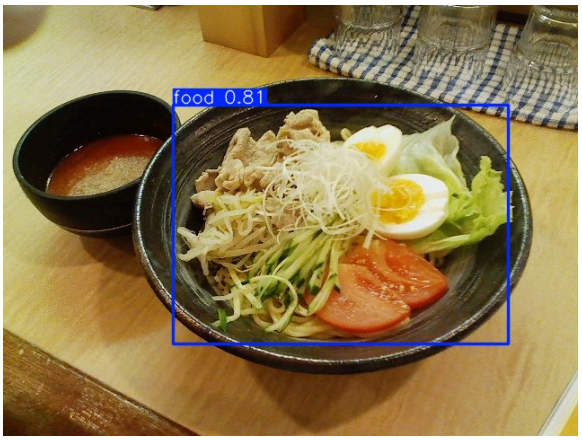
```
# Encountered issue described in https://github.com/ultralytics/ultralytics/issues/1149
# (possibly because of the GPU model -- GTX 1660 Super)
torch.backends.cudnn.enabled = False

model = YOLO("yolov8n.pt")
results = model.train(data="dataset.yaml", epochs=5, single_cls=True)
```

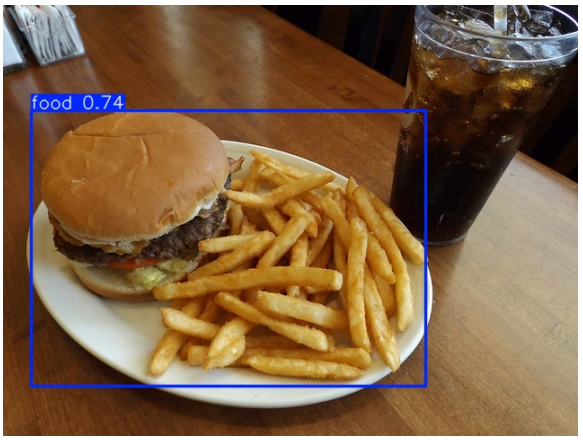
Po zakończeniu treningu funkcja ta zwraca wykresy wartości poszczególnych składowych funkcji kosztu modelu YOLO, które zamieściłem poniżej.



Wytrenowany model przetestowałem na kilku obrazach, ze zbioru testowego Open Images. Uzyskane rezultaty zamieściłem poniżej.







Do cenzurowania jedzenia na obrazach skorzystałem z biblioteki OpenCV i metody GaussianBlur.

```
for i in range(1, 5):
    results = model.predict(source=f"{TEST_DIR}/food{i}.jpg", save=True, conf=0.5)
    for r in results:
        img = r.orig_img
        for x, y, w, h in r.boxes.xywh.tolist():
            x, y, w, h = int(x - w / 2), int(y - h / 2), int(w), int(h)
            blur_img = cv2.GaussianBlur(img[y : y + h, x : x + w], (101, 101), 0)
            img[y : y + h, x : x + w] = blur_img
            cv2.imwrite(f"test_food/food{i}_blur.jpg", img)
```

Uzyskane wyniki zamieściłem poniżej.