AML homework 2

Introduction

There are 3 different datasets that were created for this homework:

1. Dataset that I manually annotated from raw old pictures
2. Dataset that was annotated automatically using Grounded DINO from raw images (and then applying SAM for segmentating)
3. Dataset that was automatically converted from founding boxes from the HW1 to segmentation. (with a help of SAM)

I provide 2 jupiters notebooks for training the first Dataset (annotated by hand)just for reference, the main solution - jupiter notebooks that were trained on the 3rd dataset

Step 1: Taking pictures

For this Assignment I simply used the previous pictures. Nothing new

Step 2: Annotating

First dataset

As I said in introduction - there are different sets that were annotated. For the manual one - I decided to classify 4 types of items in vending machines. Simply repeated the procedure of cliking on the needed item, fixing the highlighted area by adding new points and then choose an appropriate class.

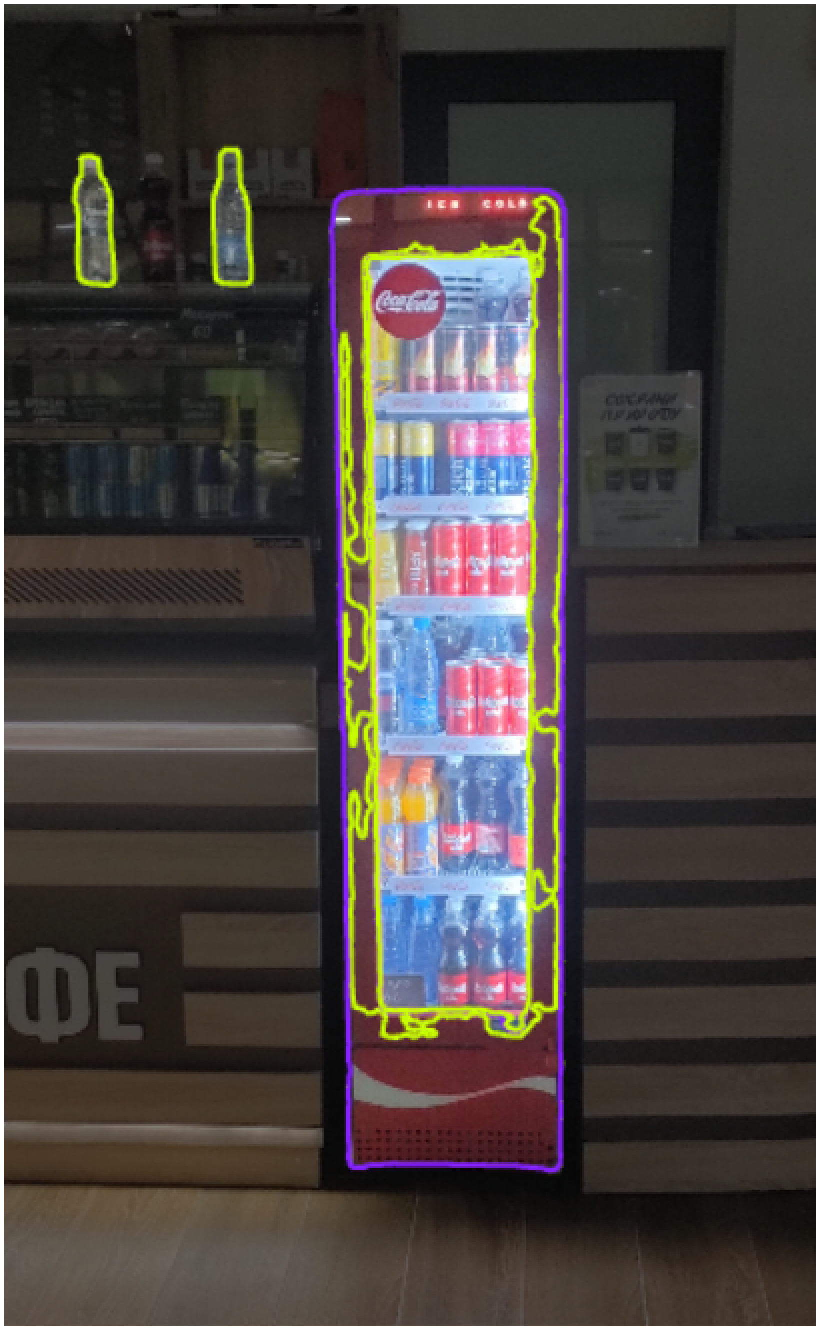
In general, nothing complicated, but monotonous and boring process

Second dataset

I found that on 21st of April Roboflow uploaded a very helpful article on how to aitomatically annotate your dataset for segmentation. So I checked it out (even after I already trained the models for the 1st dataset).

Second dataset was annotated from raw data with help of grounded DINO model, so it has some nuances.

- You have to provide names for your classes, so the DINO model could find objects of interests. Because of that it’s hard to annotate something really specific like 2 different types of corals. So you can use only some general words like “dog”, “person”, “shoe” e.t.c
- Even for small thresholds the model struggles to find some objects, so you will have to add some objects manually
- Low thresholds also implies that sometimes model will incorrectly highlite something wrong, in my case it highlighted the whole vending machine as an object. However this types of problem can be solved simply rejecting all image areas that are bigger than some other threshold, that you choose. But potentially it could lead to incorrect interpretations.



- For the same object there may be several classes. It may bring a problem when you have a class that is a subclass of another or has some intersection. In my case aluminium cans of soda are both a drink and a can(which is an intersection of 2 classes), but I wanted to treat them differently.

Overall I was satisfied with the speed of annotating data (and the ML approach to the problem), but the images didn't have a lot of class instances, so there was not enough data for training model. Therefore I came to the 3rd and final dataset.

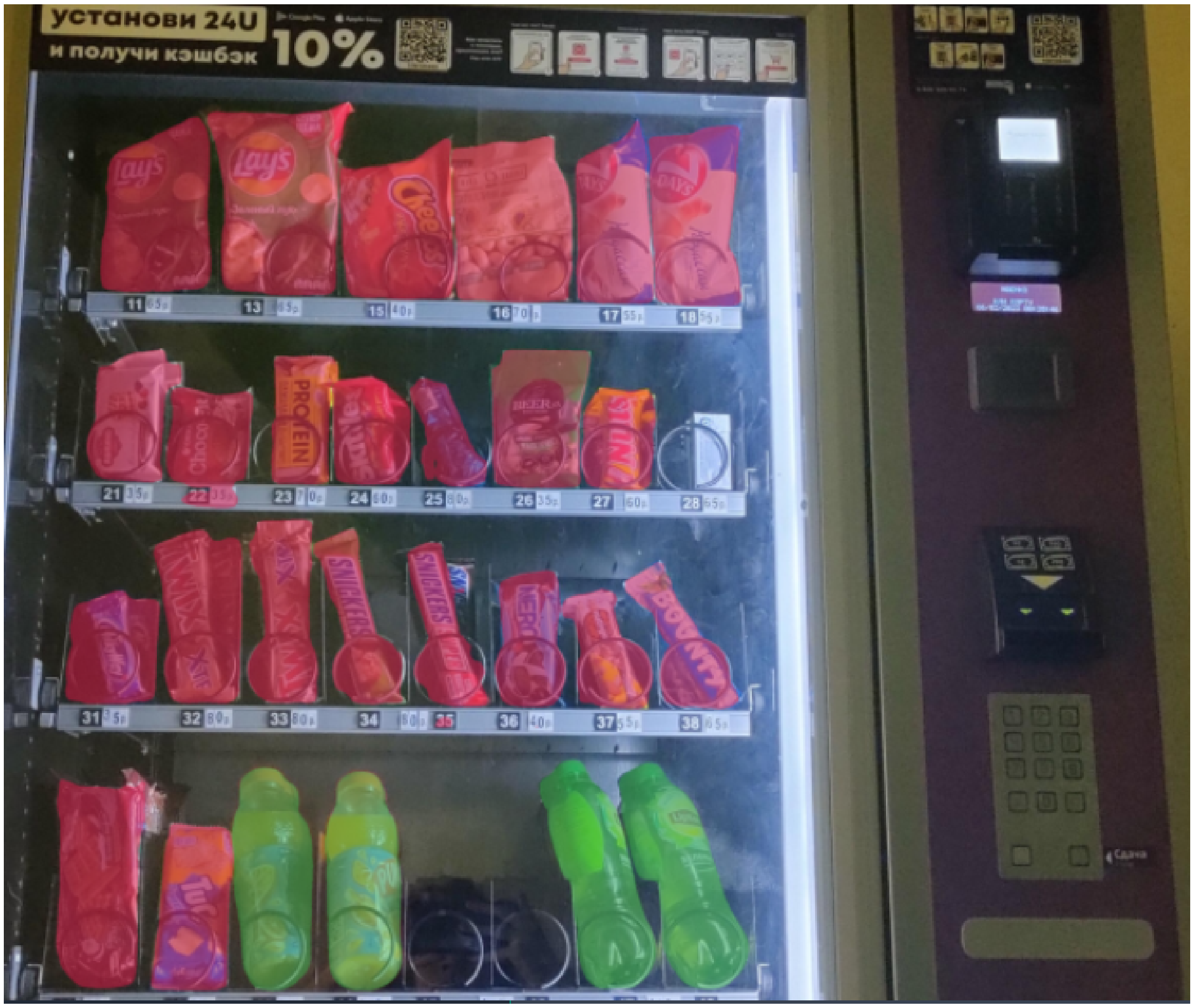
Third dataset

This dataset was made in the same jupiters notebook.

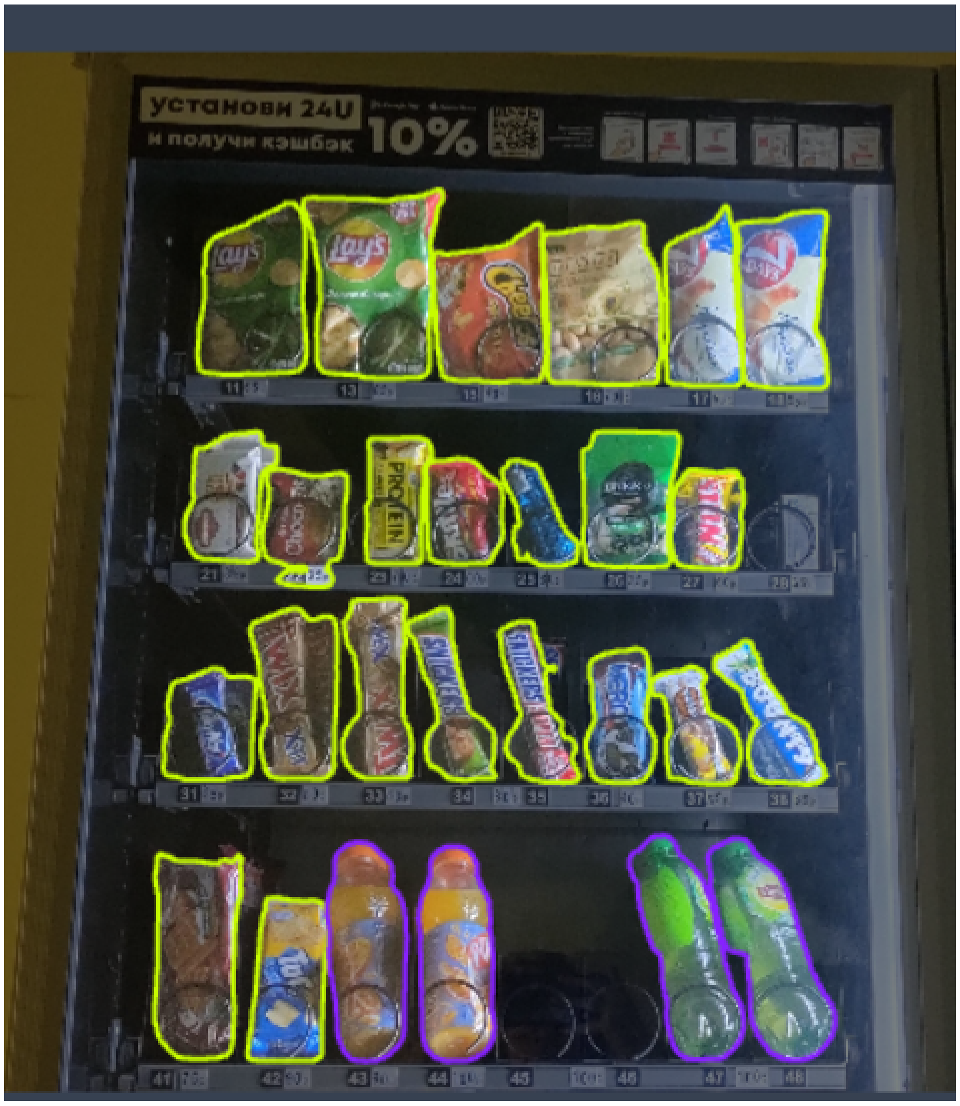
I just accessed my dataset from HW1 and used SAM for converting finding boxes to segmentation polygons.

Note that in the HW1 I had just 2 classes: food and drink, in contrast with 2 previous datasets

Picture of an annotated image in jupiters notebook



Picture of the same annotated image in roboflow



There are some problems with annotations - sometimes a polygon is larger then a real object, so for improvement of training models I could adjust that. However I overall satisfied with them and do not want to spend more time on that - therefore I leave it as it is.

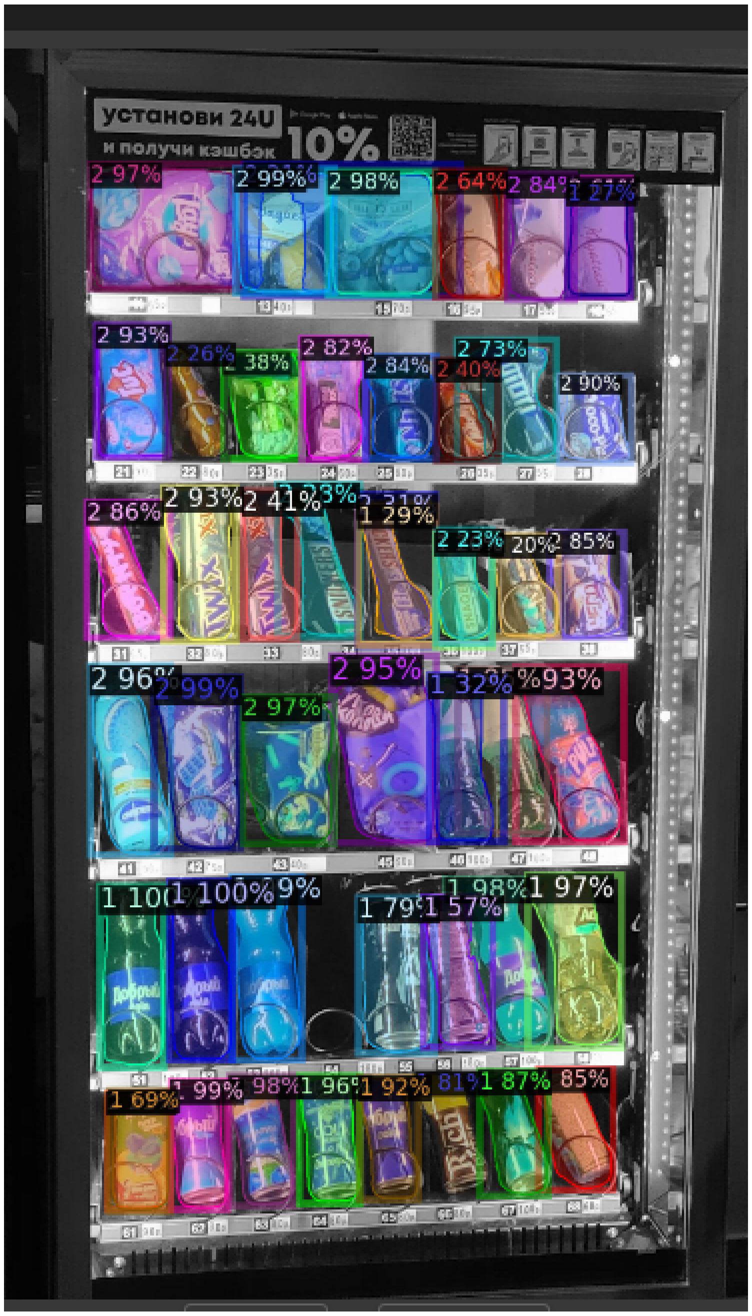
Example of the problem (I zoomed picture)

Step 3: training a Mask RCNN model using detectron2

Here it’s as simple as in the first assignment. I adjusted some parameters like Patience for early-stopping and overall number of epochs, so I don’t have to vait too much as I trained Masc RCNN for the first dataset, which took almost an hour in google collab on GPU.

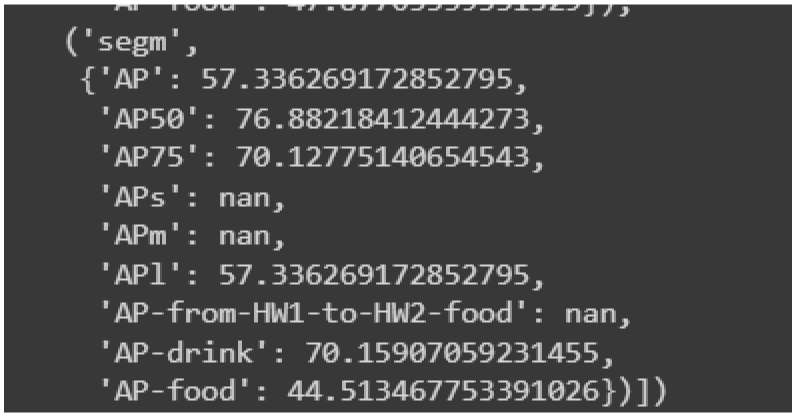
The .ipunb file can be found in the github

Here is a test image after the training of model



From this we can tell that model is quite sure about plastic bottle drinks and large packets of food, little less sure about caned soda and quite unsure about small food like candy bars (also their masks not always perfectly match the objects - sometimes it takes the surrounding background)

The issue of not perfectly matching the actuall item can be seen also in AP values

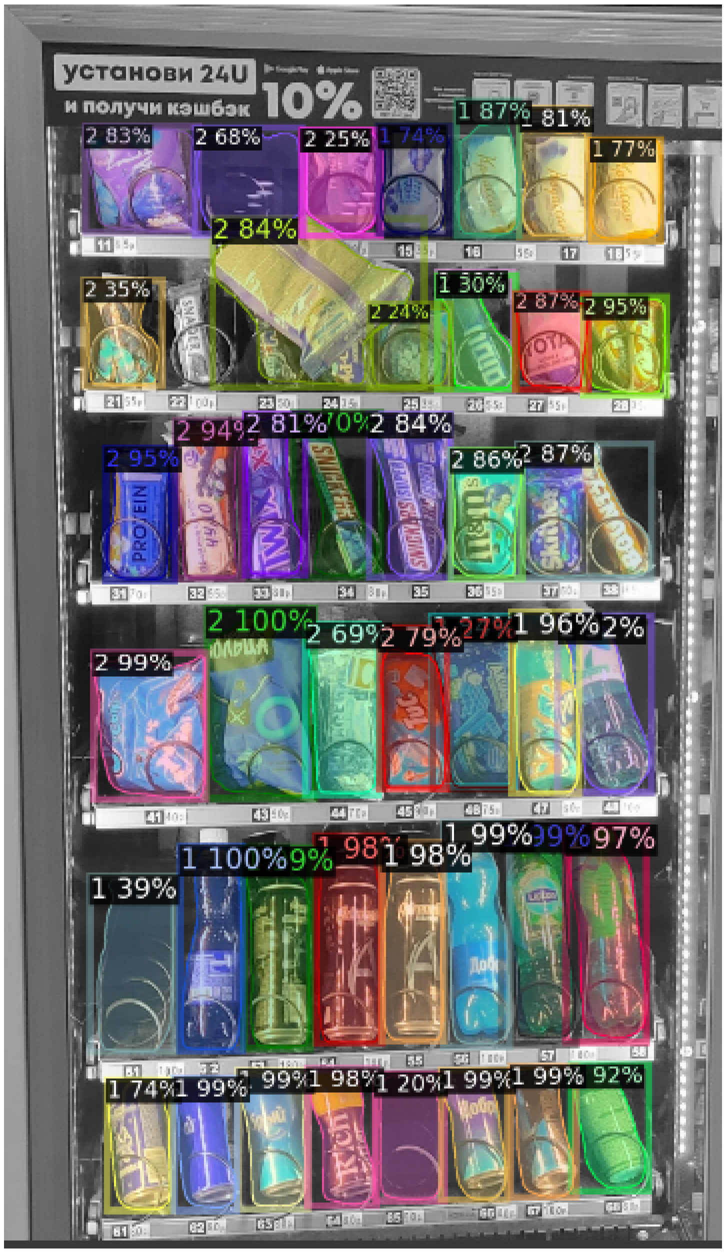


```
  ('segm',
   {'AP': 57.336269172852795,
    'AP50': 76.88218412444273,
    'AP75': 70.12775140654543,
    'APs': nan,
    'APm': nan,
    'APl': 57.336269172852795,
    'AP-from-HW1-to-HW2-food': nan,
    'AP-drink': 70.15907059231455,
    'AP-food': 44.513467753391026})])
```

for AP75 it’s required to hav IoU at least 75%, but if the model takes a lot of background - it won’t be the case. This issue may come from the dataset, since I didn’t manually correct problem where SAM took some background or another objects in a mask for an instance. So we can improve this problem if we adjust the masks by hand.

There is also another problem that model expects an item in an empty cell, however the percentage is not that high, so we can deal with it by moving the threshold

Finally some the model can correctly find object boundaries, however it misinterprets its class, like top right food on the picture, that was classified as a drink. That will affect AP later



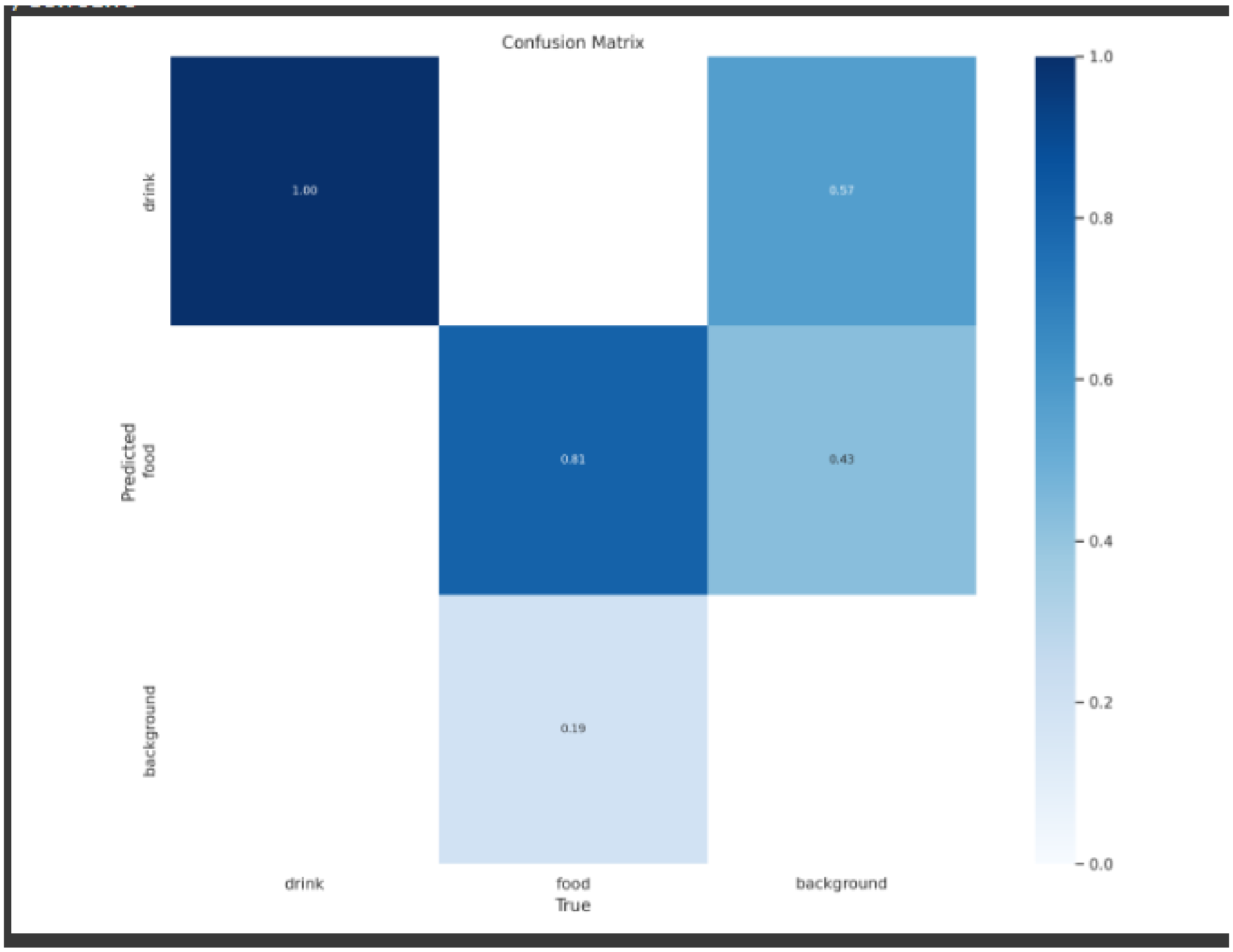
Step 4: Train Yolov8 the smallest size for segmentation

The smallest size of yolov8 for segmentation - nano. So I use yolov8n-seg for the 3rd dataset.

On the first dataset I did not succed to train small model on small number of epochs, so I tried to use the largest for 50 epoch which gave me some results (before that it didn't predict anything).

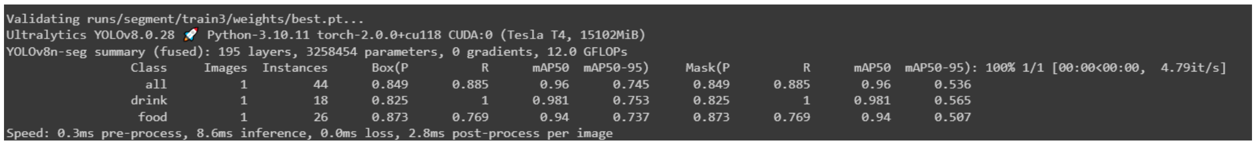
For the final solution (3rd dataset) I tried to train the smallest model (yolov8n-seg) on 200 epochs and left it as it is

Here is the confusion matrix



From this matrix we can say, that model can find all the drinks, struggles sometimes with food. and sometimes chooses places where there more background rather than actual item.

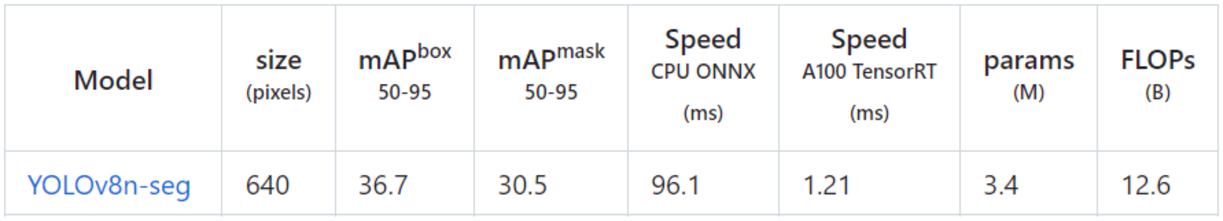
Here is the result of validation of the best model state



```
Validating runs/segment/train3/weights/best.pt...
Ultralytics YOLOv8.0.28 🚀 Python-3.10.11 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8n-seg summary (fused): 195 layers, 3258454 parameters, 0 gradients, 12.0 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.79it/s]
                   all          1         44      0.849      0.885       0.96      0.745      0.849      0.885       0.96      0.536
                 drink          1         18      0.825          1      0.981      0.753      0.825          1      0.981      0.565
                  food          1         26      0.873      0.769       0.94      0.737      0.873      0.769       0.94      0.507
Speed: 0.3ms pre-process, 8.6ms inference, 0.0ms loss, 2.8ms post-process per image
```

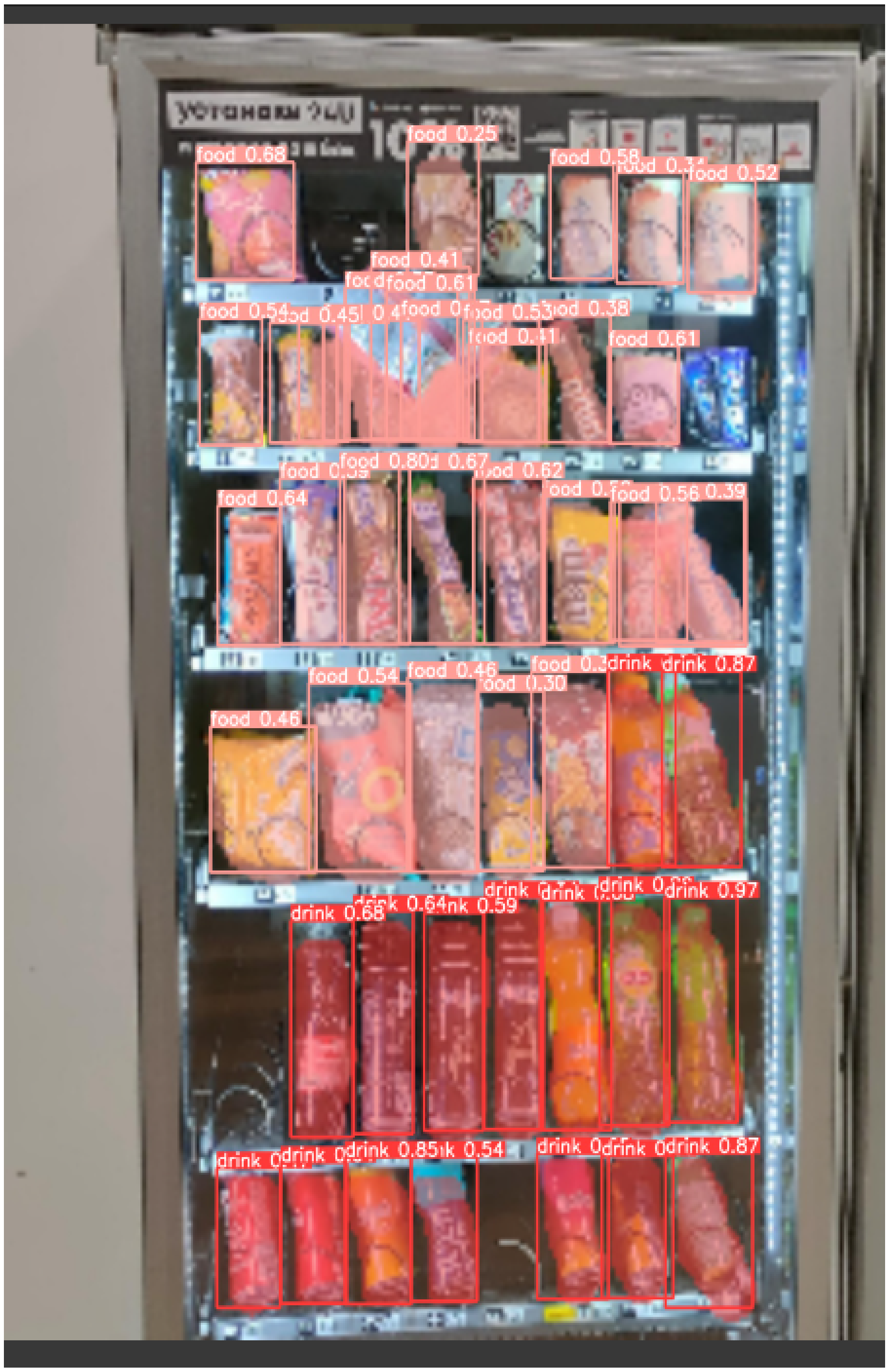
mAP50 of Mask for each class is greater than 90% which means, that only each 10th prediction has less than 50% of image area filled with not the instance of the predicted class.

mAP50-95 is actually better comparing with documentation on github for the COCO val2017 dataset.



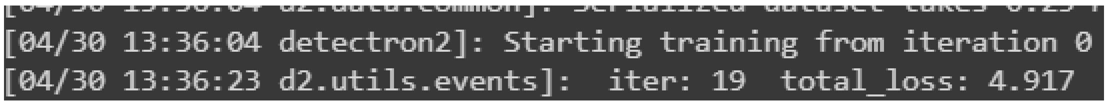
Model	size (pixels)	mAP^{box} 50-95	mAP^{mask} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n-seg	640	36.7	30.5	96.1	1.21	3.4	12.6

From the prediction on test images we can see which difficulties occur. Sometimes it matches different candy bar to one mask, the edge case with falling crisps - where model tried to mark it, but instead marked the empty space below. Those fails can be resolved by simply removing the edge cases + manually adjust the masks where several items highlighted as one (I was talking about that in th step 2)

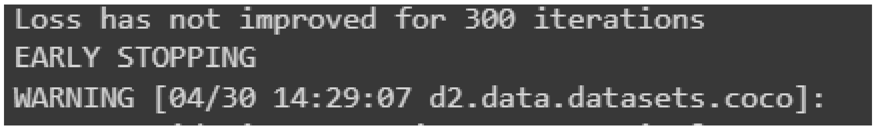


Step 5: Comparison

First comparison in time : I spend almost an hour to train the Mask RCNN(~53 minutes)

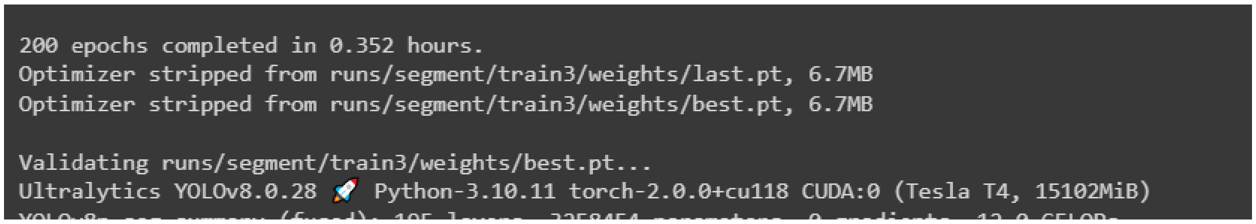


```
[04/30 13:36:04 d2.data.common]: Serialized dataset takes 0.23
[04/30 13:36:04 detectron2]: Starting training from iteration 0
[04/30 13:36:23 d2.utils.events]:  iter: 19  total_loss: 4.917
```



```
Loss has not improved for 300 iterations
EARLY STOPPING
WARNING [04/30 14:29:07 d2.data.datasets.coco]:
```

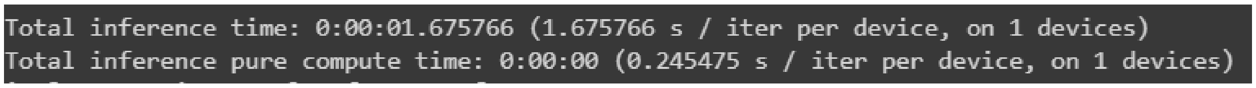
in comparison with for yolov8n (~21 minutes)



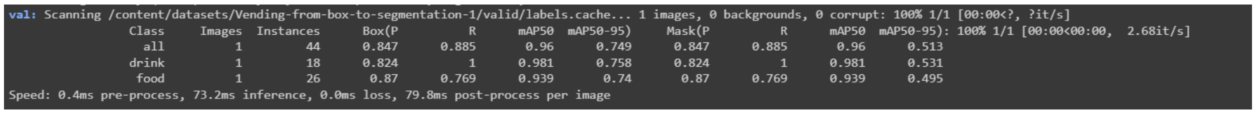
```
200 epochs completed in 0.352 hours.
Optimizer stripped from runs/segment/train3/weights/last.pt, 6.7MB
Optimizer stripped from runs/segment/train3/weights/best.pt, 6.7MB

Validating runs/segment/train3/weights/best.pt...
Ultralytics YOLOv8.0.28 🚀 Python-3.10.11 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
```

Inference time for test data:



```
Total inference time: 0:00:01.675766 (1.675766 s / iter per device, on 1 devices)
Total inference pure compute time: 0:00:00 (0.245475 s / iter per device, on 1 devices)
```

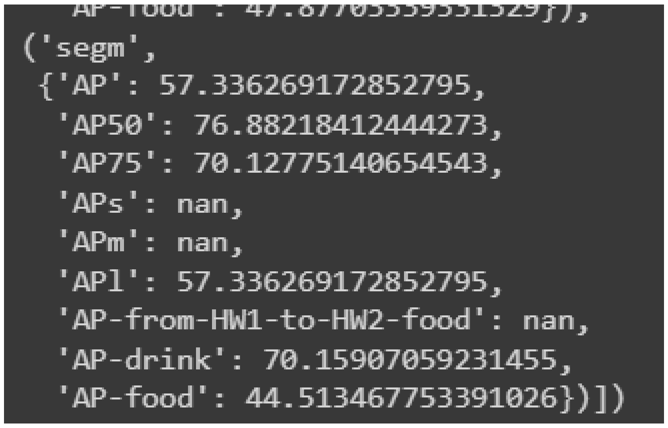


```
val: Scanning /content/datasets/Vending-from-box-to-segmentation-1/valid/labels.cache... 1 images, 0 backgrounds, 0 corrupt: 100% 1/1 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.68it/s]
                   all          1         44      0.847      0.885       0.96      0.749      0.847      0.885       0.96      0.513
                 drink          1         18      0.824          1      0.981      0.758      0.824          1      0.981      0.531
                  food          1         26       0.87      0.769      0.939       0.74       0.87      0.769      0.939      0.495
Speed: 0.4ms pre-process, 73.2ms inference, 0.0ms loss, 79.8ms post-process per image
```

Second - in size: yolov8n is definitely smaller than Mask RCNN

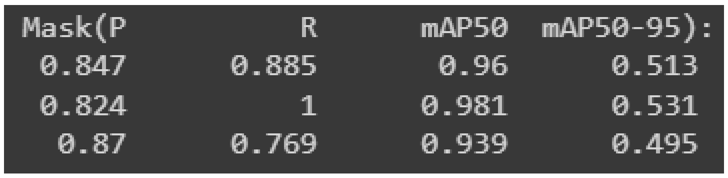
Average Precision

Mask RCNN



```
('segm',
 {'AP': 57.336269172852795,
  'AP50': 76.88218412444273,
  'AP75': 70.12775140654543,
  'APs': nan,
  'APm': nan,
  'APl': 57.336269172852795,
  'AP-from-HW1-to-HW2-food': nan,
  'AP-drink': 70.15907059231455,
  'AP-food': 44.513467753391026})])
```

yolov8n



```
Mask(P          R       mAP50  mAP50-95):
 0.847      0.885        0.96      0.513
 0.824          1       0.981      0.531
  0.87      0.769       0.939      0.495
```

The problem with Mask RCNN being worse than yolov8 on AP50 not because it cannot find the actual place of an object, but because it choses the wrong class. I would suggest to decompose current classes to multiple others, so there less dissimilar objects in the same class.

Also it seems that yolov8n tries to choose segments that are smaller than real ones, while RCNN take larger (with some background)

Here is an example of yolov8 and RCNN