PIVA Person Segmentation Challenge 2024.

Marina Grifell
B.S. Data Science and Engineering
Universitat Politècnica de Catalunya
marina.grifell@estudiantat.upc.edu

Ignacio Gris
B.S. Data Science and Engineering
Universitat Politècnica de Catalunya
ignacio.gris@estudiantat.upc.edu

Aïda Santacreu
B.S. Data Science and Engineering
Universitat Politècnica de Catalunya
aida.santacreu@estudiantat.upc.edu

I. Introduction

In this challenge, we will be working with the YoloV8 model to segment people in images. Particularly, we have been using the pre-trained version of the YoloV8 model for segmentation.

We have chosen YoloV8 model because it is a state-of-the-art model that, besides achieving pretty good results in segmentation tasks allows for easy fine-tuning and built-in data augmentation. Precisely, one of the most impressive features of YoloV8's data augmentation is the mosaic. This functionality allows to combine different images into a single one, which creates a diverse and complex input for the model that is used to increase generalization. Moreover, it includes built-in connections to visualization frameworks such as *ClearML* that allow for efficient tracking of each training experiment. Furthermore, YoloV8 model has been proven to achieve better results than other state-of-the-art segmentation models such as Mask R-CNN while allowing for faster inference [1]. Moreover, we have chosen the medium version by analyzing a threshold between the model's capacity and inference speed [2].

In this report, we will go through all the stages of the challenge process. From data preprocessing to prediction generation on a test dataset. We will conclude with some results and reflections that have been obtained during the project.

II. Preprocessing

A. Dataset generation

To train the model, we had available a dataset containing people images. This original dataset was structured in a *train* and a *test* section. In addition to the provided images, we have augmented the dataset by using the *Penn-Fudan* dataset. We have included this second dataset in the train folder to improve generalization. At this point, we have a *train* dataset, with all the image samples available for training, with their corresponding masks, and a *test* dataset, which contains the images that will be used for the final leaderboard.

Once we had the initial structure, we segmented the *train* folder by using a *train* (70%), *validation* (15%) and *test* (15%) split. This has allowed us to obtain realistic metrics related to the different models' performance and generalization. The *train* and *validation* splits have been used to train the different models, while the *test* dataset has been used to compute validation metrics for each of the tested models. For the definitive one, the *test* and *train* splits have been merged to create a new dataset that includes all data in training.

B. Mask preprocessing

The YoloV8 segmentation model works by comparing image predictions with labels. The YoloV8 label's format is quite specific, which is required for some mask preprocessing. The YoloV8 model requires labels to be a text file with a row for each person in the sample with the class identifier and the contour points of the mask. Particularly, we had to implement two different preprocessing stages.

1) From raw masks to visual masks: Raw masks in our dataset consist of an image whose values are all 0, with the exception of those pixels that are related to people. The values of these pixels go from 1 to N, where N is the number of people detected in the sample. With this step, we wanted to make images whose contours were visible enough to detect the contour points in the next stage. In Fig. 1

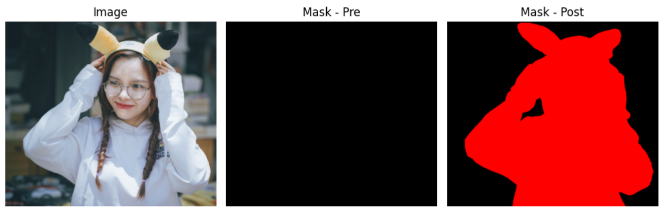

Fig. 1. Original image (left), raw masks (center) and processed mask (right)

2) From visual masks to labels: The YoloV8 model works with a specific ground truth format. We can not feed the model with the masks as labels, we need to transform the information from the mask images to plain text while representing shapes as contour points.[1]

III. Training

In this section, we will specify all the training experiments that have been done while focusing on the reasoning behind and the final model selection. We will also comment on the post-processing that has been needed to transform YoloV8's output into final mask images that can be used to compute the mIoU metric. Moreover, we will comment on the losses used during training.

[1]Moreover, PennFudan masks have been renamed so the file name of the image matches that of the mask.

A. *Losses*

To optimize the model's weights the YOLO V8 uses a loss function with four different components:

1) Bounding Box Loss: measures how well the model predicts the bounding box coordinates compared to that of the GT. Although this is a segmentation model, it outputs as well the bounding boxes.

2) Segmentation Loss: Given that the YoloV8 model includes a segmentation head that predicts a mask for each detected object, a loss that measures how well the predicted masks match the ground truth masks is used at a pixel level.

3) Classification Loss: Evaluates the predicted class probabilities for each bounding box to ensure that the model accurately classifies the detected objects. The used loss is the Binary Cross Entropy.

4) Distribution Focal Loss: Has the intention to improve object detection performance, particularly where the boundaries of object detection boxes may be unclear or difficult to predict. It addresses uncertainties in their positions [3].

IV. Fine-tunning and hyper-paramter search

The YoloV8 model comes with an already pre-trained version. Fine-tuning YoloV8's model by using its already pre-trained weights can help us to use the already learned information and adapt it to our own data.

Moreover, we have thought of using hyperparameter tuning. YoloV8 uses genetic algorithms to optimize its hyperparameters. Particularly, the model uses mutation, which helps in locally searching the hyperparameter space by applying small random changes to the already existing hyperparameters, producing new candidates for evaluation [4]. We have used this mutation algorithm to tune the hyperparameters according to our data.

Finally, we have thought about freezing some layers of the model. To be precise, we have frozen YoloV8's backbone, which consists of the first 12 layers, while allowing fine-tuning of the heads, responsible for predictions. This allows us to keep the learned feature representations while adapting predictions to our data.

Thus, the different experiments are the following.

1) Fine-tuning YoloV8 model
2) Fine-tuning YoloV8 model while freezing the backbone
3) Fine-tuning YoloV8 model by using the best obtained hyperparameters accoring to the tune method
4) Fine-tuning YoloV8 model by using the best obtained hyperparameters accoring to the tune method while freezing the backbone

V. Model selection

For each of the previously stated experiments, we performed a training during 150 epochs. For each experiment, we have obtained the mAP50:95 and mIoU metrics. These metrics have been compared among the four experiments in the *test* dataset, the one that consists of a subset of the original training data, not the one that will be used for the leaderboard. Results are shown in the following table[2].

Model	mAP50:95	mIoU
1	0.712	0.8352
2	0.845	0.8749
3	0.823	0.8719
4	0.857	0.8774

TABLE I
MAP50:95 AND MIOU RESULTS FOR TESTED EXPERIMENTS ON TEST SPLIT

We can observe that with hyperparameter tuning and backbone freezing, we have considerably improved YoloV8's baseline results. Out of these results, we can extract the following conclusions. First of all, freezing the backbone structure of the model is beneficial, as results improve about their non-freezed version both in mIoU and mAP50:95. Moreover, hyperparameter tuning is beneficial as well for the model. In particular, when implementing hyperparameter tuning on the baseline model results improve a lot. Nonetheless, when using the fine-tuned hyperparameters on the frozen model, results still increase, although not that much.

Analyzing both mAP50:95 and mIoU, it is clear that the best option is to use a fine-tuned version of the YoloV8 model starting with the pre-trained weights by using the best hyperparameters found by the mutation algorithm and freezing the backbone.

VI. Definitive training

Once the best model has been found, we have to retrain it by including the full data. As it has been said, the final chosen model has been that of the fine-tuning of the YoloV8 model by using the best hyperparameters found and freezing the backbone. This model has been trained during 300 epochs. The final obtained mAP50:95 is **0.8528**[3]. Once the final model has been trained, we have predicted the masks for the original *test* dataset that was provided.

VII. Postprocessing

YoloV8 outputs results in a specific format. This means we need to do some post-processing store results in the expected format. The generated masks are stored as (N, H, W)-shaped tensors in the *result.masks.data* variable, where N stands for the number of detected people and, by using the default YoloV8 configuration one of H or W is equal to 640. However, we need masks to have the same dimensions as the original image to correctly compare the masks. To do so, for each image, we will retrieve its dimensions H_i and W_i.

The first step of the post-processing consists in transforming the (N, H, W)-shaped tensor into a $(1, H, W)$-shaped tensor.

[2]The mAP50:95 results are the ones the ones obtained in the validation dataset, which are computing at the end of the training.

[3]In this case it makes no sense to evaluate the mIoU metric. This is because the only way of computing it is by using the validation or train datasets, which have been seen by the model and would provide over-optimistic results. We cannot compute the mIoU in the original *test* dataset because we do not have the ground truth masks

This is done by taking the maximum along the first dimension. This is done in order to generate a mask where all people's masks are overlapped, which is what we want to compute the final IoU of the image.

The next step is to transform the $(1, H, W)$-shaped tensor into a $(1, H_i, W_i)$-shaped tensor. With this mask resize we aim to obtain the predicted mask in the correct shape. This is done by converting the previous one-channel max into an image and resizing it.

At this moment we now have the final mask we want to store in the correct shape. The final step is to set all those pixels with a value greater than 0 to a high value to effectively visualize the mask. An example can be seen in Fig. 2.

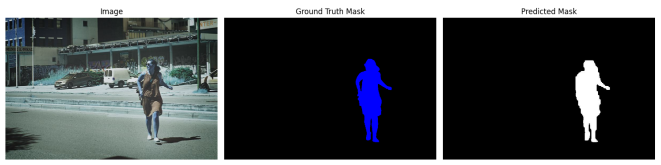

Fig. 2. Original image (left), GT mask (center) and predicted mask (right)

VIII. Results

Recall that the value of the mAP50:95 metric obtained after training the final model with all the images of the training dataset is **0.8528**. We observe a slightly lower mAP50:95 metric than the one obtained in the experimental training. This might be because of two reasons. First of all, feeding the model with more images (approximately 20% more) might make it harder for the model to learn all the variability. Moreover, by setting a patience value, we have experienced early stopping, resulting in a training that has lasted 138 epochs, instead of the 300 epochs that we expected.

After running the postprocessing pipeline, the predicted masks can be seen in the correct format and the model's performance can be also evaluated by looking at different examples of the predictions made. The model performs accurately when we evaluate images with a person that can be clearly identified. However, the resulting masks are softer than the ground truth ones. Therefore, in every prediction, there will be a default error.

When considering people that pose in such a way that there are interior regions not belonging to the person itself, the model usually fails to segment properly the instance, considering the hole part of the mask as seen in the Fig. 2 person's arm.

Additionally, when dealing with objects that are not people but have a similar shape (such as a dog), the model might fail and return a mask identifying them as a person. This can be seen for example in Fig. 3.

All the reasons above justify that the model does not have a perfect performance, resulting in a mAP value just above 0.8. However, we see highly accurate masks for most of the images. To achieve better outcomes with the more hard images, we would probably need to train a more complex model or apply alternative refinements.

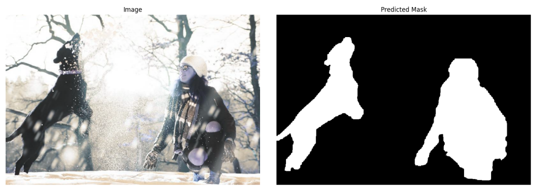

Fig. 3. Original image (left) and predicted mask (right)

IX. Conclusions about the used metric

The metric used to evaluate the results is the mean of the global IoU of each mask prediction. This metric computes the overlap between the predicted segmentation mask and the ground truth mask. It is simple and provides a straight forward interpretation of the model performance for single-object images. By averaging the IoU scores over all images it emphasizes the overall performance of the model reducing the impact of possible outliers. However, this metric only takes into account the covered area of the global mask but does not evaluate the correct separation and identification of individual objects. Thus, when the image contains overlapping objects but the predicted mask does not segment them, the global IoU may still be significantly high if the overlapping area is large enough. Additionally, when two objects are correctly identified but one of the object masks is not precisely predicted the global score of the IoU will mask this poor performance. Moreover, even is not the case, when working with different classes, the global IoU does not allow us to compute metrics per each of the classes.

Overall, this metric can effectively measure the model's performance when having single-object images but it has several limitations when used with multiple objects and more complex images. Given that most of the images in the challenge dataset are only of a single person, the mean of the IoU can be considered an accurate metric to evaluate the performance of the chosen model.

If the task would systematically include images with more than one person, probably the mAP50:95 metric would have been a better option, this is why we have been considering it as well all over the report (along the mIoU). This metric evaluates results by using different thresholds and then averaging them, resulting in a more robust metric. Moreover, it can be computed per class.

References

[1] Qeios. (2023-12-11). Comparing YOLOv8 and Mask RCNN for object segmentation in complex orchard environments [Online]. Available: https://www.qeios.com/read/ZB9SB0

[2] Ultralytics. (2023-11-12). Instance Segmentation [Online]. Available: https://docs.ultralytics.com/tasks/segment/

[3] Github. (2023-08). About the functionality of DFL Loss [Online]. Available: https://github.com/ultralytics/ultralytics/issues/4219

[4] Ultralytics. (2023-11-12). Ultralytics YOLO Hyperparameter Tuning Guide [Online]. Available: https://docs.ultralytics.com/guides/hyperparameter-tuning/introduction

[5] Youtube. (2023-04-10). Image segmentation with Yolov8 custom dataset [Online]. Availbale: https://www.youtube.com/watch?v=aVKGjzAUHz0t=591s