DEEP LEARNING

EE569

-REPORT-

“INSTANCE SEGMENTATION AND GENDER CLASSIFICATION OF PEOPLE ”

PREPARED BY:

ID: 2200208085	الطاهر عبد السلام صالح
ID: 2200208256	بدر محمد شحيم
ID: 2200208609	طلال مالك بادي

SUPERVISED BY:

د. نوري بن بركة

Table of Contents

Abstract

Abstract This project develops an instance segmentation model using Detectron2 to detect people and classify gender with the LV-MHP-V2 dataset. Parsing annotations are processed to create segmentation masks excluding hands and faces. A Mask R-CNN model is modified to include gender classification and trained with annotated data. The model is evaluated on a test set, reporting segmentation (Mask AP, AR) and classification (accuracy, precision, recall, F1-score) metrics, with visualizations demonstrating its performance

.

Introduction

This project uses Detectron2 to perform instance segmentation and gender classification on the LV-MHP-V2 dataset. Masks are generated from parsing annotations, excluding hands and faces, and a Mask R-CNN model is trained to detect people and classify their gender.

Implementation

A) Phase 1 (Setup and Data Exploration)

The environment was set up to use a local GPU for efficient training, with a compatible Python version and necessary dependencies installed for Detectron2. The LV-MHP-V2 dataset was downloaded, with annotations stored as segmentation masks that represent different body parts as binary masks or polygon coordinates. These annotations were processed into a COCO-style format for use with Detectron2, allowing the model to perform instance segmentation. The setup enabled training and evaluation of the model using the provided segmentation masks as ground truth, measuring performance through Mask Average Precision and other metrics, of course before starting we have to generate a json file that each item has a Filepath its width height and bbox with annotation (parsing annotation) path, two files are created generation_train.py and generation_val.py, now using these files to run the provided script we get the following results:

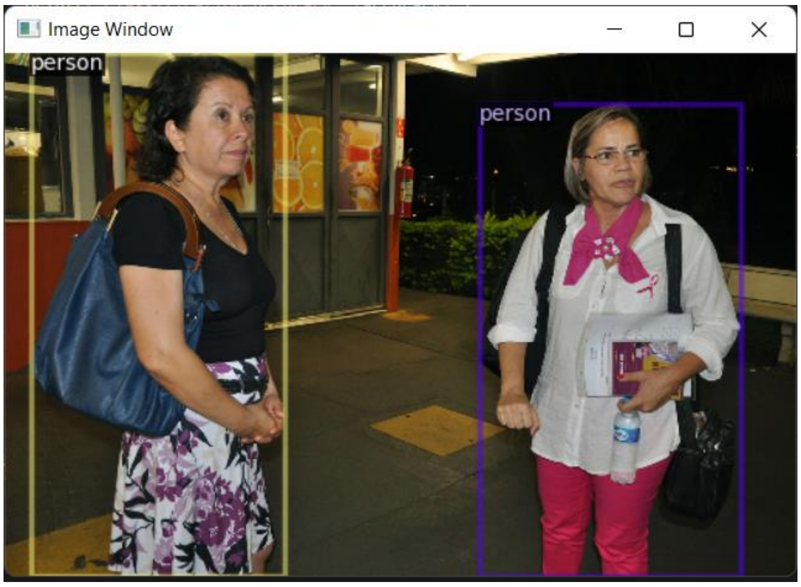
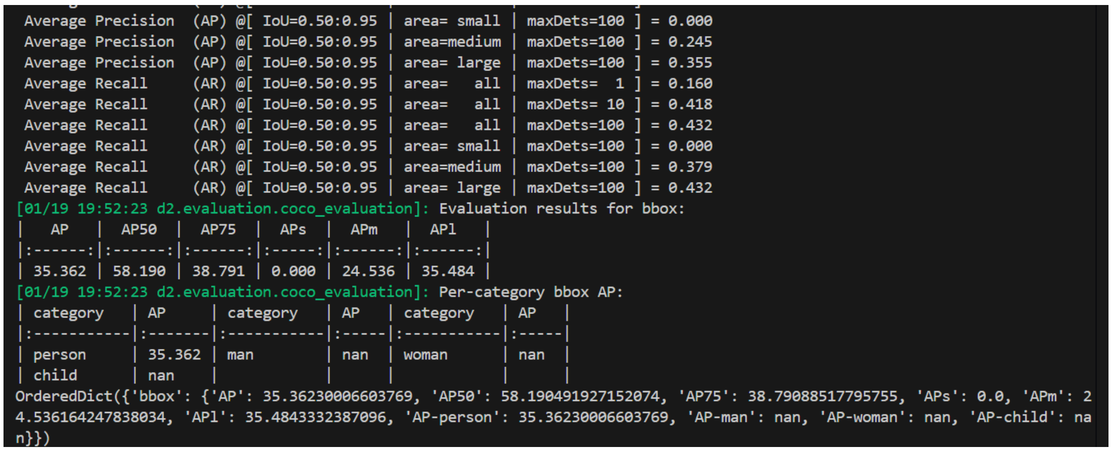


Figure 1: Visualized Output from the provided script

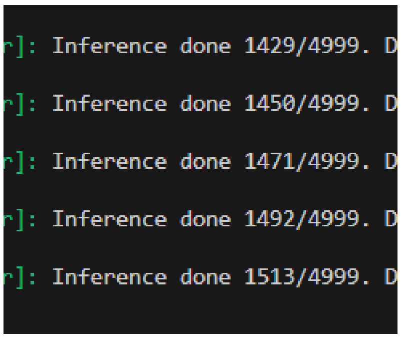


```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.245
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.355
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.160
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.418
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.432
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.379
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.432
[01/19 19:52:23 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 35.362 | 58.190 | 38.791 | 0.000 | 24.536 | 35.484 |
[01/19 19:52:23 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP     | category   | AP   | category   | AP   |
|:-----------|:-------|:-----------|:-----|:-----------|:-----|
| person     | 35.362 | man        | nan  | woman      | nan  |
| child      | nan    |            |      |            |      |
OrderedDict({'bbox': {'AP': 35.36230006603769, 'AP50': 58.190491927152074, 'AP75': 38.79088517795755, 'APs': 0.0, 'APm': 2
4.536164247838034, 'APl': 35.4843332387096, 'AP-person': 35.36230006603769, 'AP-man': nan, 'AP-woman': nan, 'AP-child': na
n}})
```

Figure 2: Evaluated Results of people

B) Phase 2 (Object Detection)

In phase 2 of the project, we utilized the Faster R-CNN model from the Detectron2 Model Zoo to establish a baseline for detecting people in the LV-MHP-V2 dataset. The pre-trained weights from the model zoo were leveraged, and the model was fine-tuned on a subset of the LV-MHP-V2 dataset using the provided bounding boxes as ground truth, after going through the images a lot of times with debugging and error I found out that one of the images were corrupted see Figure 3, after running the program the following results were outputted see Figure 3,



```
r]: Inference done 1429/4999. D
r]: Inference done 1450/4999. D
r]: Inference done 1471/4999. D
r]: Inference done 1492/4999. D
r]: Inference done 1513/4999. D
```

Figure 3: one of the images are corrupted (10059.jpg)

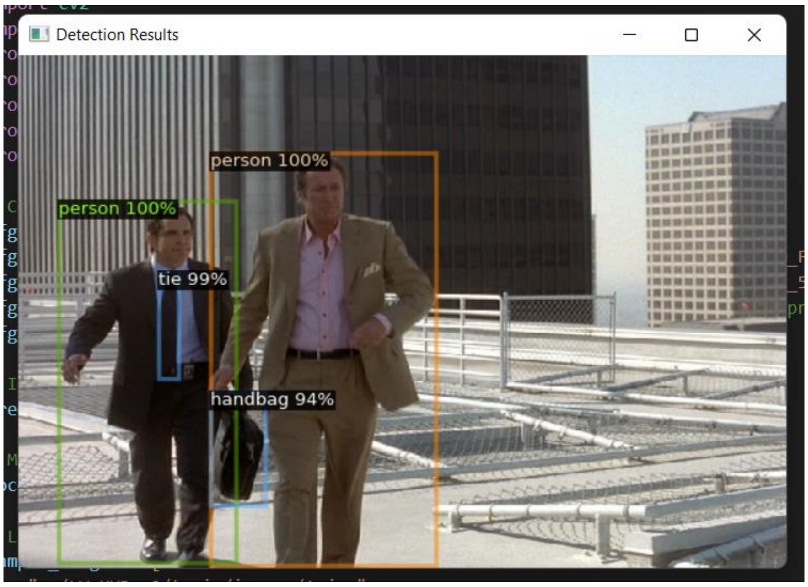
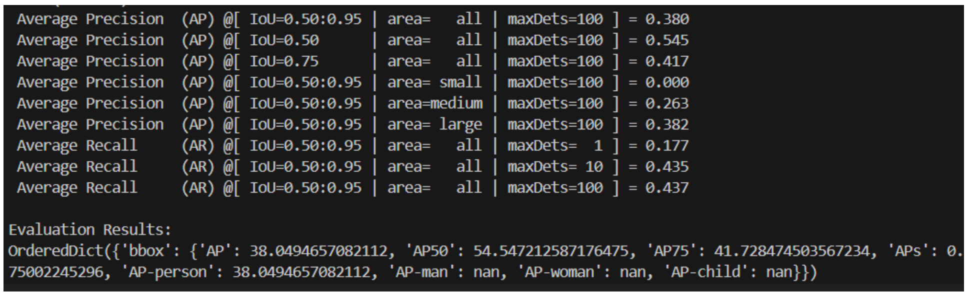


Figure 4: object detection results with model zoo



```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.380
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.545
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.417
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.263
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.382
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.177
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.435
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.437

Evaluation Results:
OrderedDict({'bbox': {'AP': 38.0494657082112, 'AP50': 54.547212587176475, 'AP75': 41.728474503567234, 'APs': 0.
75002245296, 'AP-person': 38.0494657082112, 'AP-man': nan, 'AP-woman': nan, 'AP-child': nan}})
```

Figure 5: AP metrics for objects and persons (man and woman currently not included)

C) Phase 3 (Data Preparation for Instance Segmentation and Gender Classification)

1. Data Annotation

In this section, we focus on enhancing our dataset by incorporating gender labels (male and female) for selected images using CVAT. The goal is to create a balanced dataset with an approximately equal number of male and female annotations. Our dataset will consist of around 400 images for training and 60 images for validation, maintaining an 80% to 20% split. As illustrated in Figure 5, we utilize CVAT to annotate the dataset with gender labels, providing accurate ground truth annotations essential for training our instance segmentation model for gender classification, after the process is done the cvat will give you two json files one for Train and one for Val, the segmentation field is currently empty we will fill it with it’s mask after we generate them.

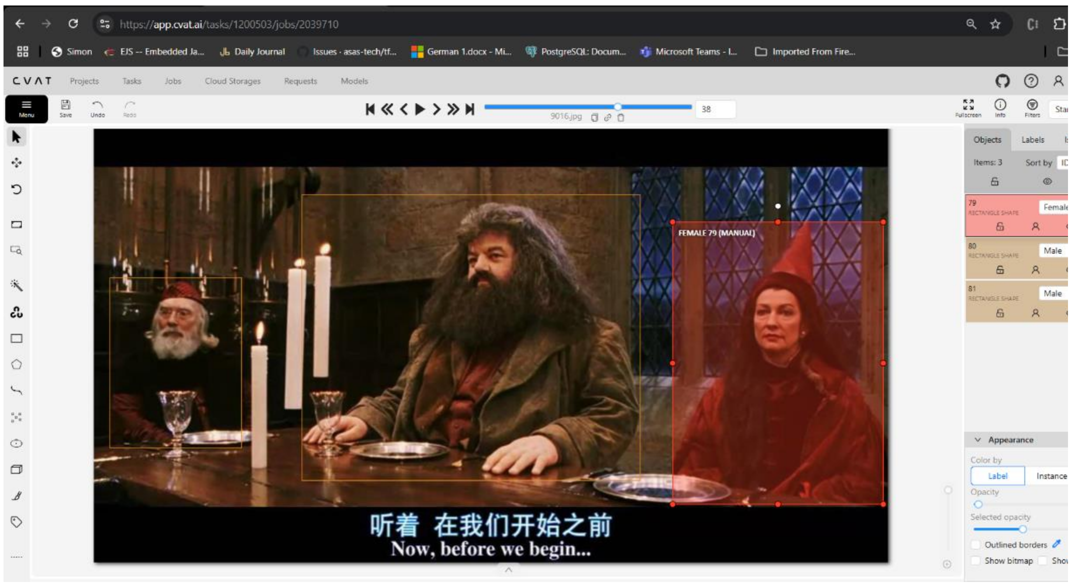
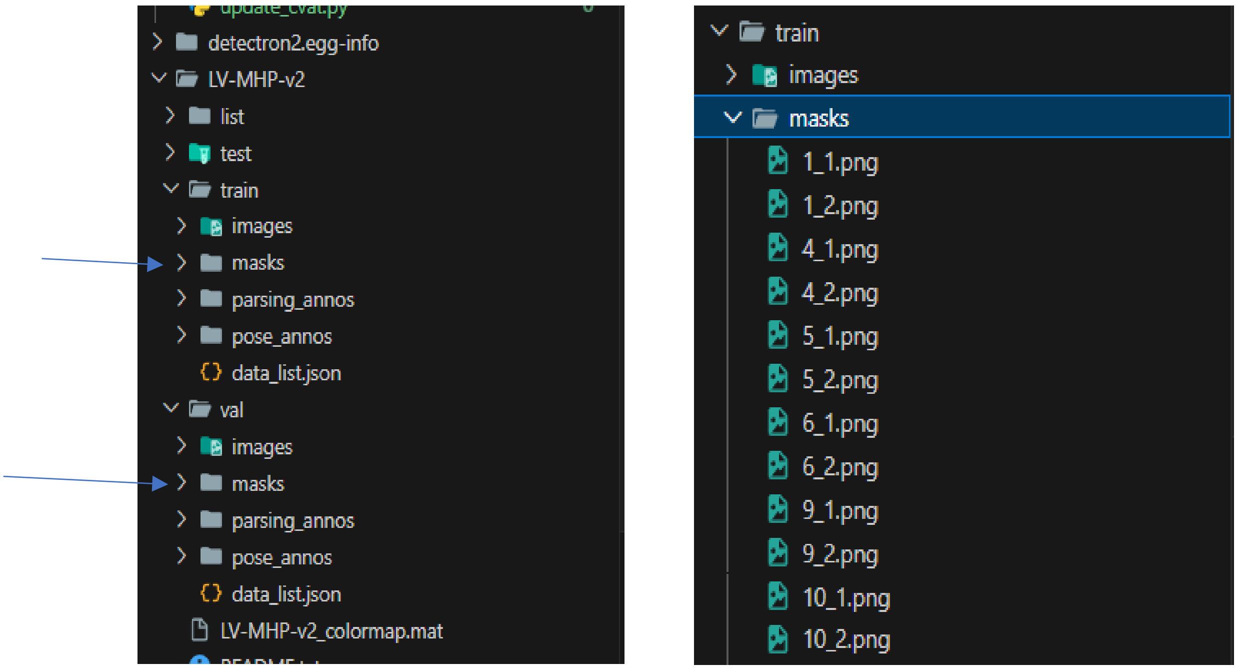


Figure 6: Annotating images with Gender

2. Mask Generation from Parsing Annotations:
to generate segmentation masks from parsing annotations, we will implement a clear and consistent naming convention for the output mask files. Each mask file will be named using the format image_name_mask_inumber.png. The first number corresponds to the original image file name (depending on whether it belongs to the training or validation set), and the second number indicates the mask index within that image. For example, an image named 1.jpeg with its first mask will be saved as 1_1.png. This structured naming ensures that each mask is easily traceable to its source image and facilitates organized dataset management, as illustrated in Figure 7.



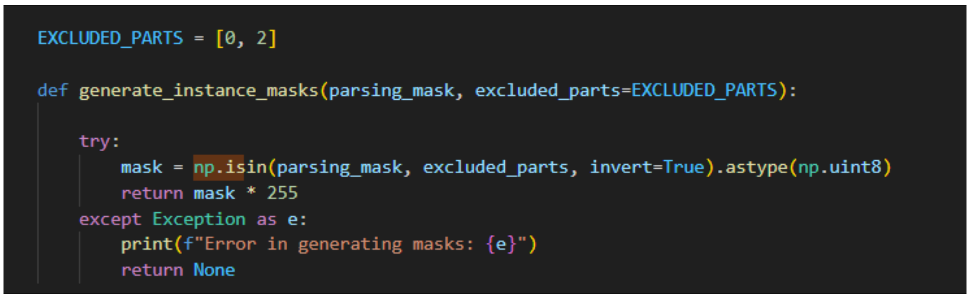
```
update_cvat.py
> detectron2.egg-info
v LV-MHP-v2
  > list
  > test
  v train
    > images
    > masks
    > parsing_annos
    > pose_annos
      data_list.json
  v val
    > images
    > masks
    > parsing_annos
    > pose_annos
      data_list.json
    LV-MHP-v2_colormap.mat
```

```
v train
  > images
  v masks
      1_1.png
      1_2.png
      4_1.png
      4_2.png
      5_1.png
      5_2.png
      6_1.png
      6_2.png
      9_1.png
      9_2.png
      10_1.png
      10_2.png
```

Figure 7: File Structure and naming convention

3. Dataset Class Modification

An important note from our work with the LV-MHP-v2 dataset is that the annotations provided in the README file are shuffled. After extensive testing and debugging, we determined that label 0 corresponds to hands and label 2 corresponds to the face. When generating segmentation masks, we exclude these parts by specifying EXCLUDED_PARTS = [0, 2] to focus on more relevant body parts. The key function used to generate instance masks is np.isin, which checks for the presence of specific values within an array see Figure 8.



```python
EXCLUDED_PARTS = [0, 2]

def generate_instance_masks(parsing_mask, excluded_parts=EXCLUDED_PARTS):

    try:
        mask = np.isin(parsing_mask, excluded_parts, invert=True).astype(np.uint8)
        return mask * 255
    except Exception as e:
        print(f"Error in generating masks: {e}")
        return None
```

Figure 8: Generating masks with numpy

The np.isin function creates a boolean array indicating which elements of parsing_mask are in excluded_parts. The invert=True parameter flips the boolean values, effectively excluding the parts specified in excluded_parts. The result is then cast to np.uint8 to convert True values to 1 and False values to 0. Finally, multiplying by 255 transforms the mask into a binary image where the excluded parts are black (0) and the retained parts are white (255). This is the most critical step in generating the segmentation masks, the generation takes some time about 25k masks each for train and val, the following figure shows the results of the masks and the applied mask for a certin image.

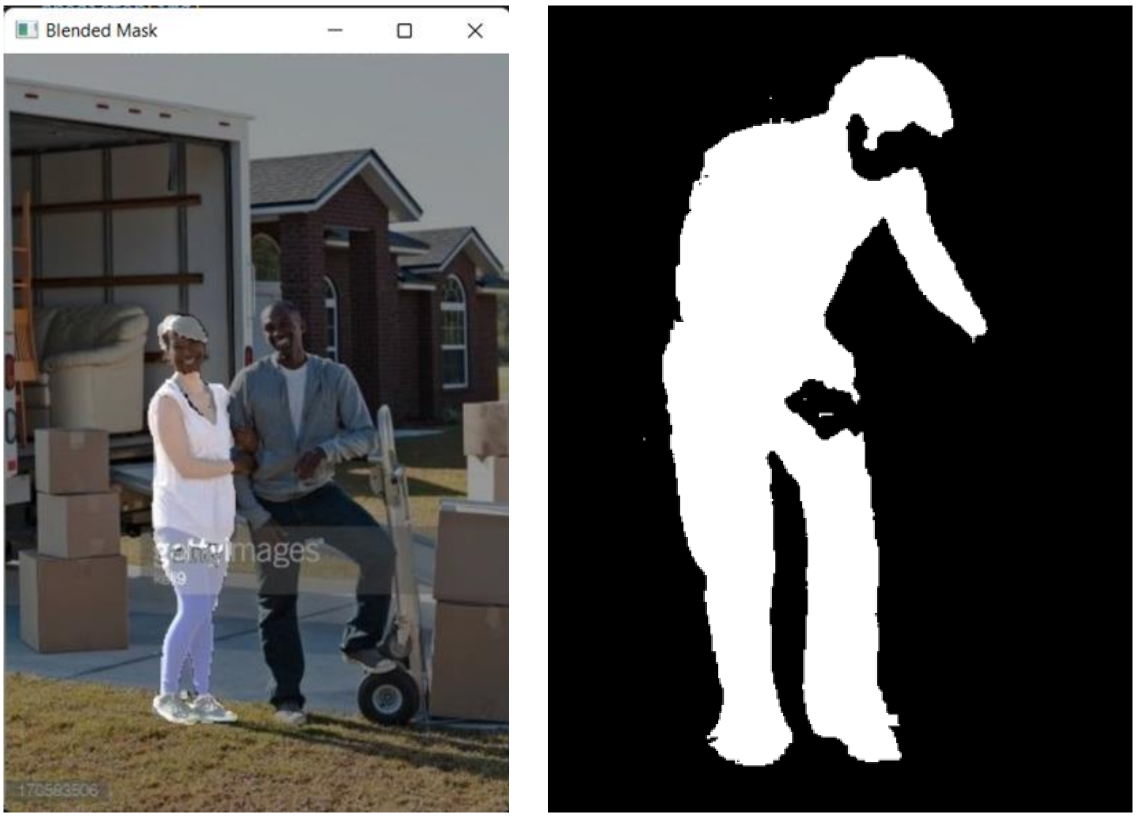


Figure 9 mask result and a mask overlaid on an image

Now applying certain cover for the mask either overlay black or gaussian blur, only the Females are applied the masks for, to cover them up, using both gaussian blur and overlay black see Figure 10 and Figure 12

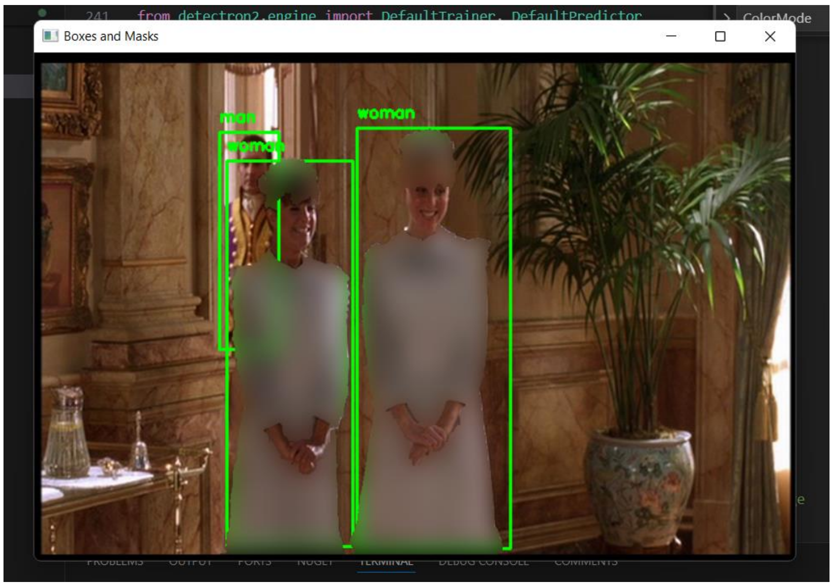


Figure 10 Gaussian Blur overlaid on Female only

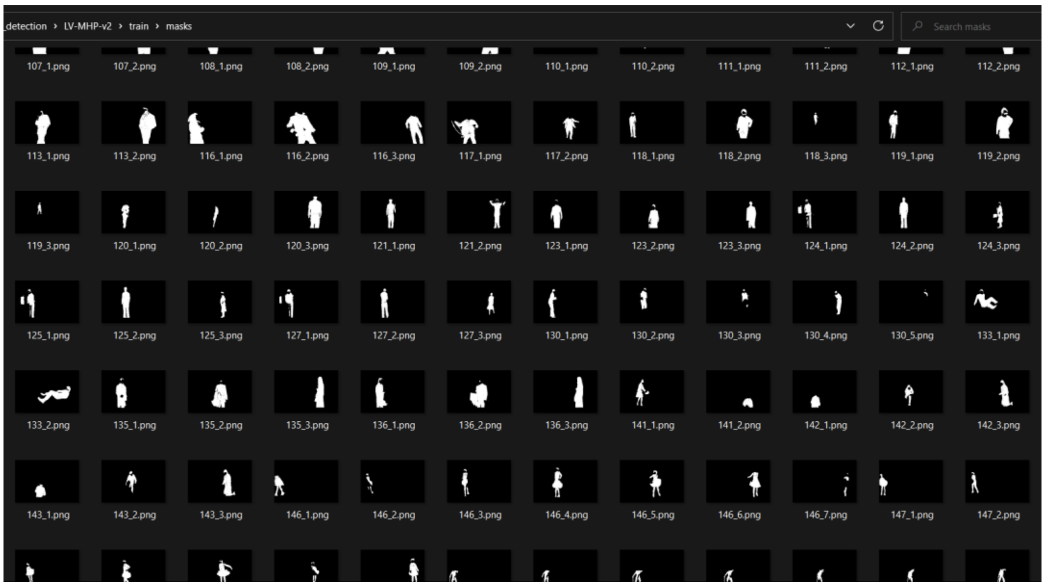

detection › LV-MHP-v2 › train › masks
Search masks
107_1.png 107_2.png 108_1.png 108_2.png 109_1.png 109_2.png 110_1.png 110_2.png 111_1.png 111_2.png 112_1.png 112_2.png
113_1.png 113_2.png 116_1.png 116_2.png 116_3.png 117_1.png 117_2.png 118_1.png 118_2.png 118_3.png 119_1.png 119_2.png
119_3.png 120_1.png 120_2.png 120_3.png 121_1.png 121_2.png 123_1.png 123_2.png 123_3.png 124_1.png 124_2.png 124_3.png
125_1.png 125_2.png 125_3.png 127_1.png 127_2.png 127_3.png 130_1.png 130_2.png 130_3.png 130_4.png 130_5.png 133_1.png
133_2.png 135_1.png 135_2.png 135_3.png 136_1.png 136_2.png 136_3.png 141_1.png 141_2.png 142_1.png 142_2.png 142_3.png
143_1.png 143_2.png 143_3.png 146_1.png 146_2.png 146_3.png 146_4.png 146_5.png 146_6.png 146_7.png 147_1.png 147_2.png

Figure 11 25k+ images for both train and validation

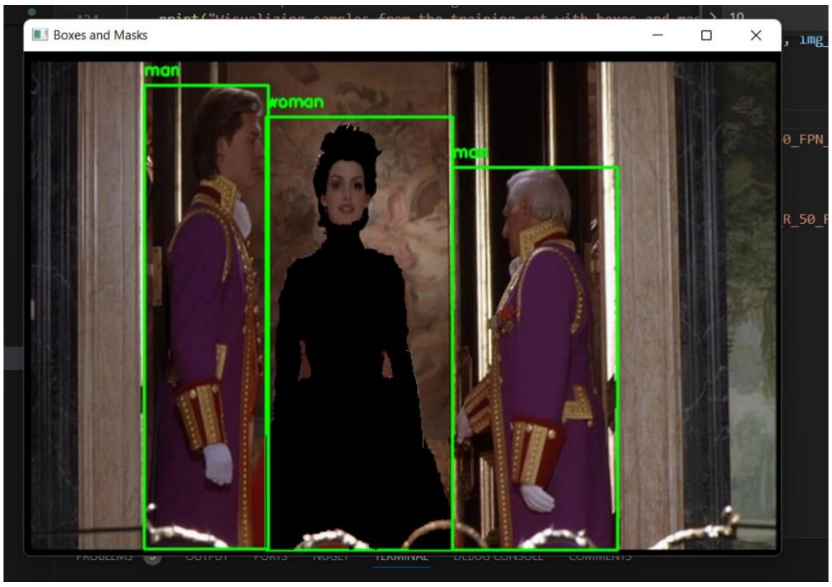


Figure 12 Overlay Black overlaid on Female only

How will you handle the gender labels during training?
gender labels during training as a separate classification task within the Detectron2 framework.

D) Phase 4 Instance Segmentation and Gender Classification Model Training

Before the JSON annotations are updated, the segmentation field from the CVAT-generated JSON file must be converted into a format compatible with COCO (either RLE or polygons). Polygons will be used for this purpose.

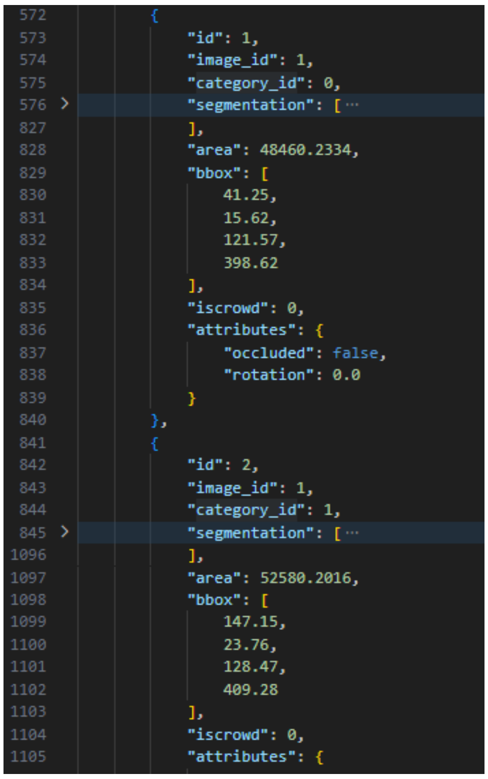
The model can be trained with masks for both male and female categories, or for a single category (e.g., female only). To train for a specific category, the provided code line must be uncommented:

```
# "segmentation": poly if ann["category_id"] == 1 else None,
```

Where category id 1 is female and 0 is male
Weights must be initialized from the Mask R-CNN model for the initial run.
Segmentation training must be enabled by setting:
cfg.MODEL.MASK_ON = True
To resume training from the last iteration, resume=True must be set to load previously saved weights from the output directory and prevent resetting to iteration 0:
trainer.resume_or_load(resume=True)

How will you incorporate gender classification into the mode ?

Gender classification will be integrated as an additional classification task within the dataset dictionary. Each instance will be assigned a category_id representing its gender class (0 for male, 1 for female). The
cfg.MODEL.ROI_HEADS.NUM_CLASSES
parameter will be set to 2 to account for both gender categories.



```
        {
            "id": 1,
            "image_id": 1,
            "category_id": 0,
            "segmentation": [⋯
            ],
            "area": 48460.2334,
            "bbox": [
                41.25,
                15.62,
                121.57,
                398.62
            ],
            "iscrowd": 0,
            "attributes": {
                "occluded": false,
                "rotation": 0.0
            }
        },
        {
            "id": 2,
            "image_id": 1,
            "category_id": 1,
            "segmentation": [⋯
            ],
            "area": 52580.2016,
            "bbox": [
                147.15,
                23.76,
                128.47,
                409.28
            ],
            "iscrowd": 0,
            "attributes": {
```

Figure 13 mapping masks with their gender

For optimization, we experimented with various hyperparameters, data augmentations (such as flipping), and other training strategies. While gender classification has shown significant improvement, segmentation evaluation metrics remain suboptimal and learning very slowly, However, visual inspection of the segmentation outputs (see Figure 13 & Figure 14) suggests that the model produces somewhat acceptable results, indicating a possible discrepancy between the evaluation metrics and qualitative results.

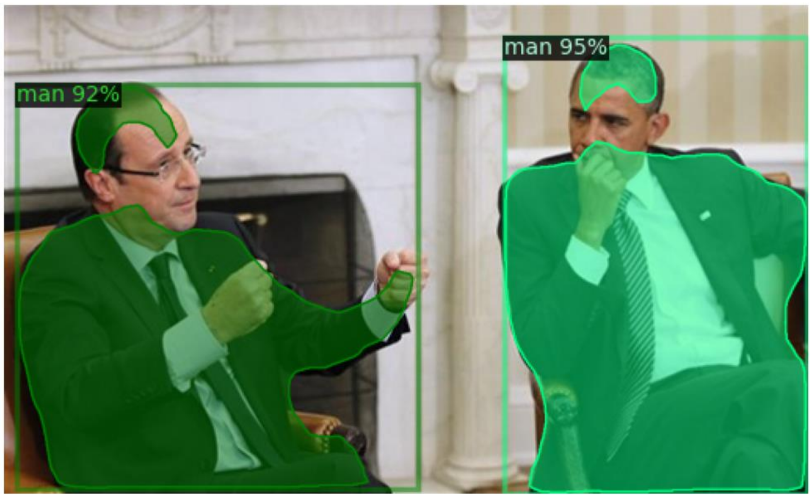


Figure 13 Visual Predictions of the Segmentation

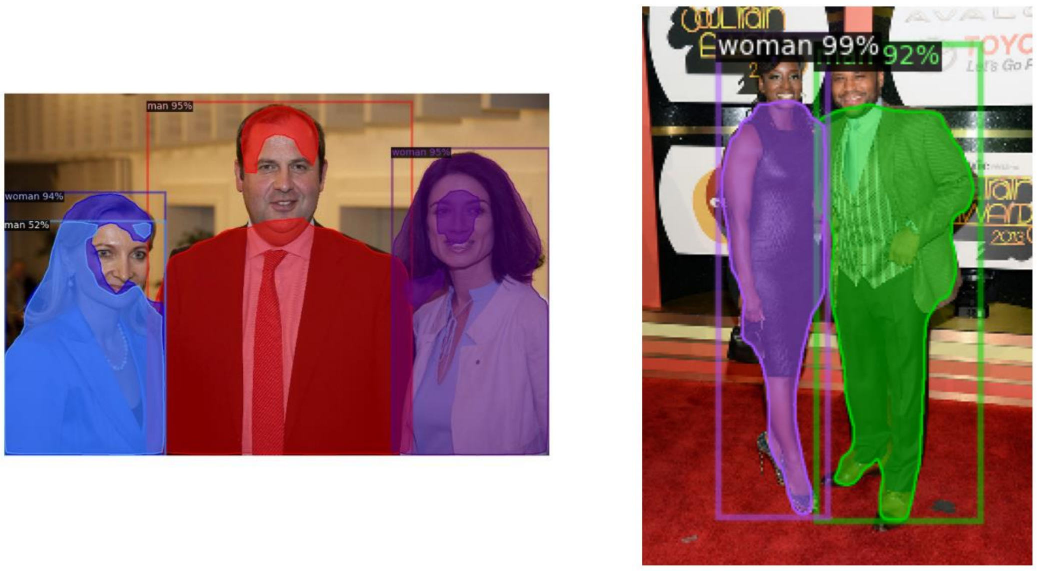
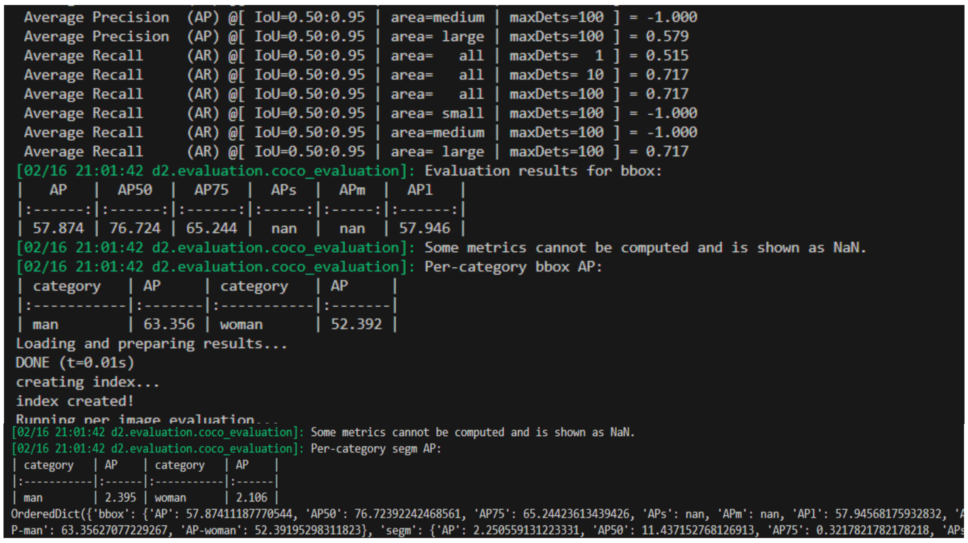


Figure 14 Visual Predictions of the Segmentation

The model is trained using segmentation masks for both male and female categories, providing the flexibility to apply masks selectively for males, females, or both during inference.



```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.579
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.515
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.717
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.717
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.717
[02/16 21:01:42 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs  |  APm  |  APl   |
|:------:|:------:|:------:|:-----:|:-----:|:------:|
| 57.874 | 76.724 | 65.244 |  nan  |  nan  | 57.946 |
[02/16 21:01:42 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[02/16 21:01:42 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP     | category   | AP     |
|:-----------|:-------|:-----------|:-------|
| man        | 63.356 | woman      | 52.392 |
Loading and preparing results...
DONE (t=0.01s)
creating index...
index created!
Running per image evaluation...
[02/16 21:01:42 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[02/16 21:01:42 d2.evaluation.coco_evaluation]: Per-category segm AP:
| category   | AP    | category   | AP    |
|:-----------|:------|:-----------|:------|
| man        | 2.395 | woman      | 2.106 |
OrderedDict({'bbox': {'AP': 57.87411187770544, 'AP50': 76.72392242468561, 'AP75': 65.24423613439426, 'APs': nan, 'APm': nan, 'APl': 57.94568175932832, 'A
P-man': 63.35627077229267, 'AP-woman': 52.39195298311823}, 'segm': {'AP': 2.250559131223331, 'AP50': 11.437152768126913, 'AP75': 0.3217821782178218, 'APs
```

Figure 15 Metric Predictions of the Segmentation

E) Phase 5 Evaluation and Visualization

In phase 5, we utilize the trained model weights to evaluate predictions on the test dataset, generating both segmentation masks and gender classification results (see Figure 15). You can see that the aggressive training caused some overfitting.

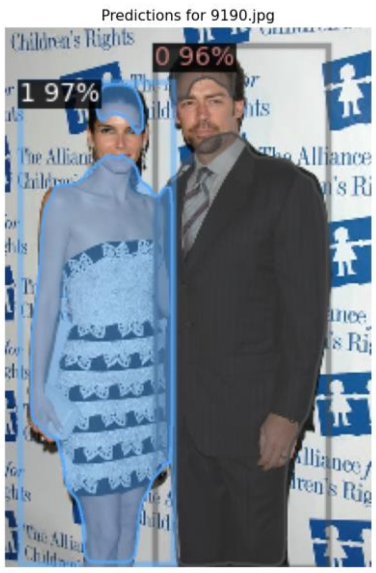


Figure 16 Visual Predictions of the Segmentation on Test Data

Accuracy: 0.4108, Precision: 1.0000, Recall: 0.4108, F1-score: 0.5824

We can see that the precision is perfect one because of aggressive training and overfitting, we can lower that by updating the weights and doing lesser iterations or generalization techniques like drop out.

Conclusions

In this project, we successfully implemented gender classification for male and female categories using an instance segmentation model trained on the **LV-MHP-V2** dataset. Our model was trained to detect individuals, classify their gender, and generate segmentation masks using parsing annotations for both training and validation images.The gender classification results were promising, with strong performance metrics indicating that the model effectively distinguished between male and female categories. However, while the segmentation metrics were relatively low, visual inspection of the predictions demonstrated more accurate and detailed segmentation results than the numerical metrics suggested. This highlights a potential gap between the evaluation criteria and the model's actual performance. In the final phase, we applied our trained model weights to the test dataset, achieving reasonable results in both segmentation and gender classification tasks. Despite facing technical challenges such as limited GPU memory, insufficient RAM, and corrupted datasets, we successfully navigated these obstacles and gained valuable experience in training and optimizing deep learning models for instance segmentation and classification. This project provided an in-depth understanding of Detectron2, human parsing, and instance segmentation, offering a strong foundation for future work in similar domains.

References

- https://github.com/Nuri-benbarka/EE569DeepLearningFall2024 \
- https://medium.com/@victorolufemi/building-your-first-object-detector-with-detectron-ii-9d9e19de6273