Pixel Labeling

Description of the Problem and the Chosen Technique

This report details how a mask R-CNN model is used to label pixels of images in a fashion dataset with a set of semantic classes [3]. The first task is to label all the pixels within the periphery of the person in the picture. The second is to label the pixels within each item of clothing that the person is wearing. The following explains this further using an example from the dataset being used in this project.

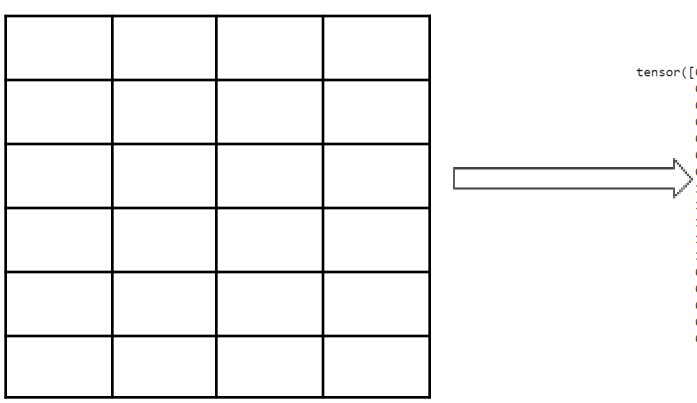
Image

Mask 1

Mask 2

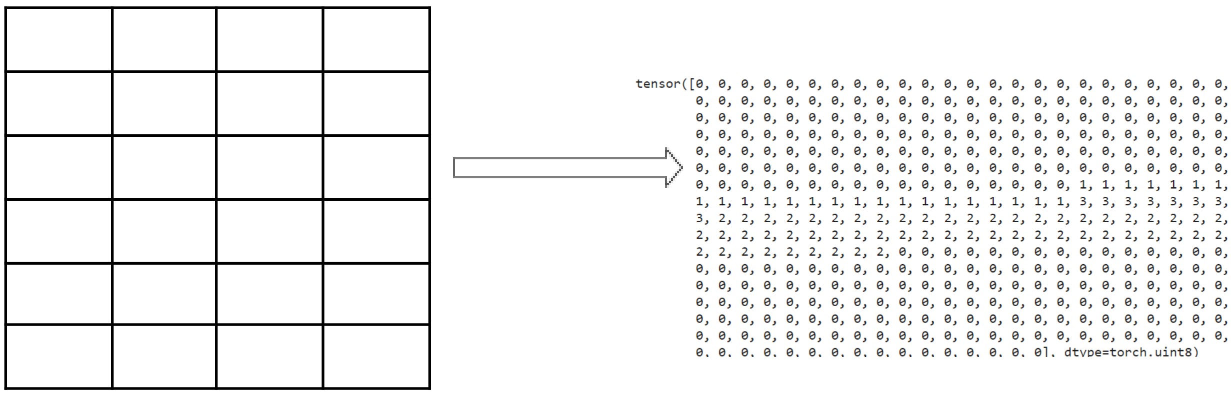
'Image' is an example from the fashion dataset that is used in this project. 'Mask 1' is a mask of the image where pixels are labeled as 'person' or 'background'. 'Mask 2' shows pixels labeled as 'background', 'skin', 'hair', 't-shirt', 'shoes', and 'pants'. Another category not included in this picture but is a part of the dataset's classes is 'dress'.

The dataset includes 600 unique images of people wearing different clothes. Each image has a height of 600 pixels, width of 400 pixels, and 3 channels for the RGB colors, thus giving them a dimension of 600 x 400 x 3. There are 600 masks for the people and 600 masks for the clothing items for the respective images, each of which have dimensions 600 x 400 x 1, where each pixel is labeled with a number representing its class.



```
tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=torch.uint8)
```

The figure above is an example of a mask that classifies pixels as the background or a person with the image on the right displaying the pixels in one of the rows. Each pixel is labeled as 0 or 1 with 0 denoting 'background' and 1 denoting 'person'.



Similarly, the figure above is an example of a mask that classifies pixels as the background or clothing items with the image on the right displaying the pixels in one of the rows. Each pixel is labeled as 0 for 'background' or a number from 1 to 6 denoting a different clothing item. The labels are 1 for 'skin', 2 for 'hair', 3 for 't-shirt', 4 for 'shoes', 5 for 'pants', and 6 for 'dress' [3].

The goal of this project is to build a program that is able to create these masks by labeling pixels in an unseen image with their appropriate class category. This can be done by the mask R-CNN model which is an instance segmentation model that does object detection followed by semantic segmentation. The following sections will describe the components of the pipeline, which includes descriptions of the data loading process, the model architecture, and the process in which the resulting masks are created from the model's output, the training process, evaluation of the results, the insights gained and key findings, challenges encountered, and potential future work.

<u>Components of the Pipeline</u>

Data Loading:

50% of the data is used for training, 10% for validation, and 40% for testing.

Once they are loaded, the images and masks undergo formatting to meet the input requirements of the mask R-CNN model for training. The pixels in each image are divided by 255.0 to make sure all pixel values are between the ranges of 0 and 1. Each image is converted into a tensor object. All the images are then stacked into one tensor object which has the dimension C x H x W where C is the number of images, H is the height of each image, and W is the width of each image [5].

For training, the mask R-CNN model requires a list of dictionaries called targets. A target has labels, masks, and boxes. A label is a tensor that has all the class labels that appear in a mask. It has a size of N where N is the number of classes that appear in an image. The mask tensor is of size N x 1 x H x W. It has

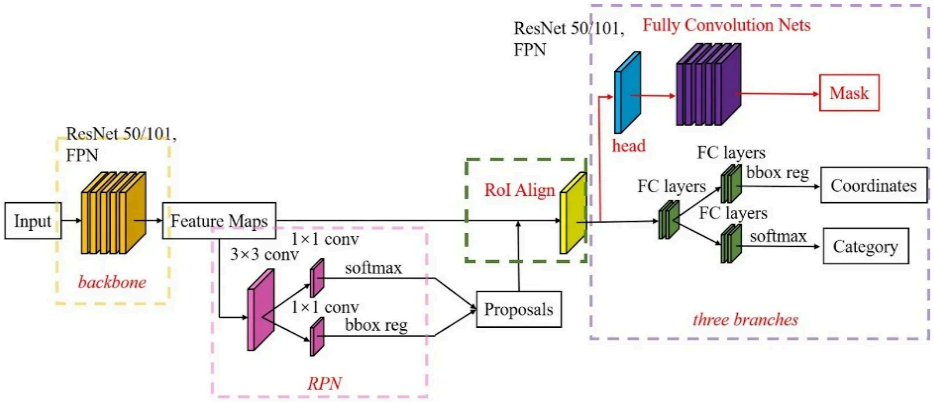
N segmentation binary masks for each of the objects where each pixel is labeled 1 if it belongs in the class and 0 if it does not. The box tensor is of size N x 4 which includes ground-truth boxes, (x1, y1, x2, y2), for each of the classes in an image [5]. The following is an example of a target dictionary from the targets list for the masks that labels pixels as 'background' or 'person':

```
{'masks': tensor([[[0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         ...,
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0]]], dtype=torch.uint8), 'boxes': tensor([[147.,  20., 284., 592.]]), 'labels': tensor([1])}
```

The following is an example of a target dictionary from the targets list for the masks that does labels pixels as 'background' or one of the labels of the clothing items:

```
{'masks': tensor([[[0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         ...,
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0]],

        [[0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         ...,
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0]],

        [[0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         ...,
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0]],

        [[0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         ...,
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0],
         [0, 0, 0,  ..., 0, 0, 0]]], dtype=torch.uint8), 'boxes': tensor([[101.,  32., 248., 515.],
        [179.,   5., 256.,  84.],
        [177., 496., 254., 594.],
        [127.,  82., 250., 330.]]), 'labels': tensor([1, 2, 4, 6])}
```

Mask R-CNN Model Architecture:



The Mask R-CNN model utilizes a ResNet-50-FPN backbone, consisting of a ResNet with 50 layers pretrained on ImageNet for feature extraction. The Feature Pyramid Network (FPN) processes the extracted features to construct a feature pyramid, providing the Mask R-CNN with multi-scale features for improved object detection and segmentation [1, 4].

The network's head begins with a Region Proposal Network (RPN), which generates bounding box proposals around objects in an image. Subsequently, the RoIAlign layer pools the proposed regions into fixed-sized feature maps. To maintain spatial accuracy and alignment with the extracted features, RoIAlign employs bilinear interpolation to compute the value of each sampling point. This process ensures precise localization of objects within the feature maps [1, 4].

Following region pooling, the model performs object classification and bounding box regression to predict class labels and refine the bounding boxes. Concurrently, a fully convolutional mask prediction branch predicts a binary mask for each object, labeling pixels corresponding to object presence as 1 from background pixels, labeled as 0. This allows the model to achieve instance-level segmentation, accurately labeling object boundaries within the image [1, 4].

Output Processing:

The images for validation and testing are passed through the trained model, which outputs a list of dictionaries, one for each image. Each dictionary contains an N-sized tensor of labels, an N x 1 x H x W tensor of masks, an N x 4 tensor of boxes, and an N-sized tensor of scores, where N is the number of objects predicted in the image. The tensor of labels lists all the predicted object classes in the image. The tensor of scores contains confidence scores, ranging from 0 to 1, for each predicted object, along with its mask and bounding box. The tensor of boxes includes the predicted bounding boxes for each object. The tensor of masks contains segmentation binary masks for each predicted object, where pixels predicted to be part of the object are labeled as 1 and others as 0 [5].

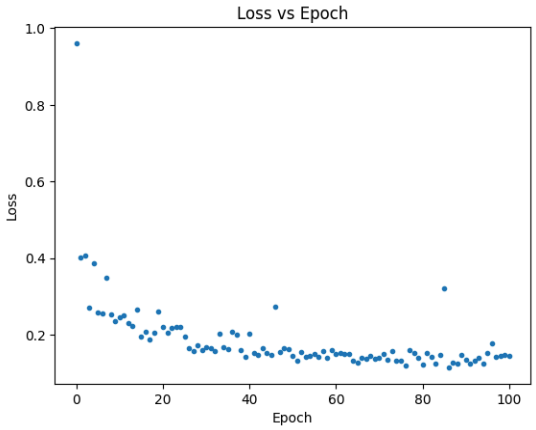
To create a singular mask for each image, masks with confidence scores greater than a specified threshold are selected. A combined mask tensor is then created, where all pixels are initialized to 0 for the 'background' class. Next, the selected masks are iterated over, and for each pixel predicted to be part of an object, the label of that object is assigned to the corresponding pixel in the combined mask.

Training Process

The weights of the model have been pre-trained on the large-scale object detection dataset called COCO through a process outlined by a team at Facebook AI Research [1, 2]. The hyperparameters that are trained in this project are the learning rate, the batch size, and the score threshold.

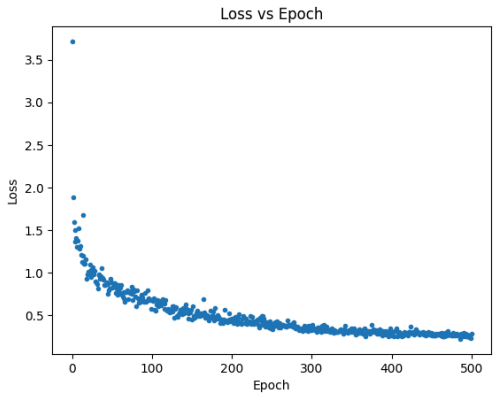
The total loss function of the Mask R-CNN model has three main components: classification loss, bounding box regression loss, and mask prediction loss, calculated separately for each sampled Region of Interest (RoI) during training. The mask loss is computed independently for each object class present in the image, applying the per-pixel sigmoid function and averaging the binary cross-entropy loss between predicted and ground-truth masks [1].

During training, the model parameters are iteratively adjusted through the stochastic gradient descent (SGD) process to minimize the total loss. Training occurs over multiple iterations, or epochs, with a small batch of samples randomly selected from the entire training dataset in each iteration. The batch size and learning rate are hyperparameters chosen to reduce the loss until convergence, determined through experimentation.
The following is the change in loss of the model predictions when trained to detect people:



The optimal learning rate is 0.0001 and optimal batch size is 10.

The following is the change in loss of the model predictions when trained to detect clothing items:



The optimal learning rate is 0.0001 and the optimal batch size is 10.

Once the optimal learning rates and batch sizes are found and used to train the model, the score thresholds have to be set to find the most accurately predicted masks. This is done using the predicted masks for the images in the validation dataset. The accuracy of the predictions are given by the following metric:
true positives / (true positives + false positives + false negatives)

The thresholds were set such that the highest accuracy could be achieved. The accuracies achieved for the predicted mask that label pixels as ‘people’ and ‘background’ are as follows:

```
>> evalseg
testing 60 images
Accuracy for each class (intersection/union measure)
      background: 67.005%
          person: 45.485%
-------------------------
person accuracy: 45.485%

ans =

   45.4852
```

The optimal threshold is 0.75.

The accuracies achieved for the predicted mask that label pixels as ‘background’ or as clothing items are as follows:

```
>> evalseg
testing 60 images
confusion: 53/60
Accuracy for each class (intersection/union measure)
      background: 75.783%
            skin: 12.549%
            hair: 12.560%
          tshirt: 15.340%
           shoes: 26.609%
           pants: 23.653%
           dress: 22.360%
-------------------------
skin accuracy: 18.845%
hair accuracy: 18.845%
tshirt accuracy: 18.845%
shoes accuracy: 18.845%
pants accuracy: 18.845%
dress accuracy: 18.845%

ans =

   12.5485
   12.5600
   15.3398
   26.6089
   23.6527
   22.3604
```

The optimal threshold is 0.75.

Evaluation

To evaluate the performance of the model, the model accuracy was finally evaluated on the predicted masks of the test dataset. The accuracy of the predictions are given by the following metric:
true positives / (true positives + false positives + false negatives)

The accuracies achieved for the predicted mask that label pixels as ‘people’ and ‘background’ are as follows:

```
>> evalseg
testing 240 images
confusion: 47/240
confusion: 98/240
confusion: 148/240
confusion: 198/240
Accuracy for each class (intersection/union measure)
      background: 68.147%
          person: 47.444%
-------------------------
person accuracy: 47.444%

ans =

   47.4438
```

The accuracies achieved for the predicted mask that label pixels as ‘background’ or as clothing items are as follows:

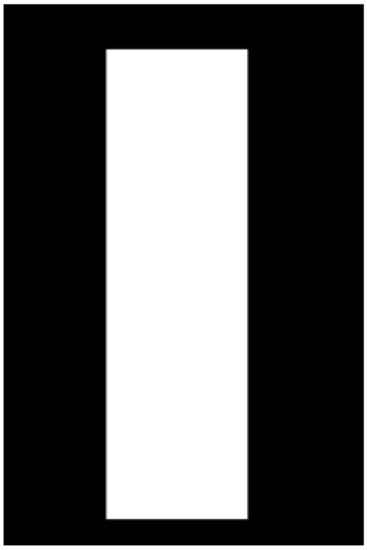
```
>> evalseg
testing 240 images
confusion: 47/240
confusion: 93/240
confusion: 139/240
confusion: 185/240
confusion: 232/240
Accuracy for each class (intersection/union measure)
      background: 76.044%
            skin: 11.692%
            hair: 12.673%
          tshirt: 20.470%
           shoes: 25.843%
           pants: 25.189%
           dress: 17.194%
-------------------------
skin accuracy: 18.844%
hair accuracy: 18.844%
tshirt accuracy: 18.844%
shoes accuracy: 18.844%
pants accuracy: 18.844%
dress accuracy: 18.844%

ans =

   11.6920
   12.6729
   20.4702
   25.8432
   25.1890
   17.1936
```

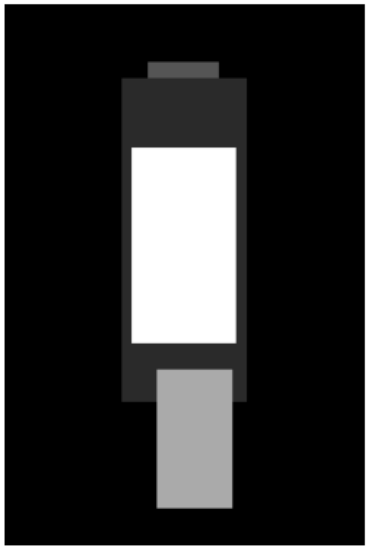
Insights Gained and Key Findings

The model performed really well on the training dataset as shown by the training curves displayed above. The losses resulting from the pixel-level labeling for both people and clothing items fell to around 20%. However, it can be seen that the model performed better in cases where it only had to classify pixels between ‘background’ and ‘person’ compared to classifying pixels between different clothing objects. The model had to be trained for five times longer to reach the same level of loss value in the case of the latter.

The model’s validation and test results’ accuracies were not very high when compared to the model’s performance during training. In both cases, the results for the classification of pixels of a ‘person’ had an accuracy between 45 and 47% and for the classifications of pixels for different clothing objects had an accuracy of around 18.8%. It can be noticed that, even in these cases, the model is able to conduct binary classifications at a higher accuracy rate compared to classifying pixels into a larger number of classes. Based on these results, it can be deduced that either the model is overfitting on the training dataset, or the method in which the optimal masks have been chosen from the predicted masks in the output of the model needs to be optimized further.

The images below showcases the predicted pixel-level labels to classify objects in the image on the left:





All the other results of the model are similar to this example, i.e. they have not performed better or worse. It can be seen that the objects of interest have been recognized and the pixels accordingly classified, but the pixels around the objects have been misclassified as well. This is giving rise to a large number of false positives which is leading to the low accuracy rate on the validation and test datasets. While the results might seem better for the model classifying pixels that are in a person compared to the performance of the model classifying pixels in a clothing object, this is because the mask of a person is not as detailed as that of a clothing item and it also takes up more space in an image, leading to less false positives.

Challenges Encountered

Reaching convergence of the training curve for the model labeling pixels for clothing items was challenging. Small increases in the learning rate led to the model parameters oscillating over each epoch and the resulting losses changed erratically from the ones prior. Alternatively, small decreases in the learning rate led to insignificant decreases in the losses over each epoch. Moreover, when masks were predicted from different classes on images from the validation set, the confidence scores for each of the predicted labels fell within a small range making them all equally likely. This made it difficult to find an optimal threshold to select the most accurate masks.

The challenges encountered when training the model to label pixels as 'person' were not as severe and it did not require a lot of experimentation to begin seeing results in both the training and validation stages. However, it proved to be difficult to improve accuracy regardless of the numerous values that were tried for the hyperparameters.

Potential Future Work

Increasing the performance of the models depends on training them for longer on a larger number of images. This can be done by collecting more images but also by augmenting the current ones.

The FAIR team has also stated that the mask R-CNN model they proposed has a simple convolutional neural network that is used to predict the masks and it is possible that a more complex model can lead to better results [1]. Further research can therefore be done in this domain.

References

Note: The mask R-CNN model provided in the PyTorch's torchvision package was used in the code to build the model and described in detail in the report. References that have been included below of coding tutorials were mostly used to develop an understanding of the package and the model before adapting the information for the specific use of this project. Special focus was given to not copy any sections of code directly.

[1] G. Gkioxari, K. He, P. Dollar, and R. Girshick, "Mask R-CNN," Facebook AI Research (FAIR), January 2018.

[2] G. Patterson, J. Hays, L. Bourdev, L. Zitnick, M. R. Ronchi, M. Maire, R. Girshick, S. Belongie, T. Lin, P. Perona, D. Ramanan, P. Dollar, and Y. Cui, "What is COCO?", COCO, 2015. [Online]. Available: https://cocodataset.org/#home. [Accessed: April, 2024].

[3] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, T. L. Berg. "Parsing Clothing in Fashion Photographs," CVPR, 2012.

[4] P. Potrimba, "What is Mask R-CNN? The Ultimate Guide." Roboflow, 2023. [Online]. Available: https://blog.roboflow.com/mask-rcnn/. [Accessed: April, 2024].

[5] PyTorch, "maskrcnn_resnet50_fpn," PyTorch, 2017. [Online]. Available: https://pytorch.org/vision/main/models/generated/torchvision.models.detection.maskrcnn_resnet50_fpn.html. [Accessed: April, 2024].

[6] PyTorch, "Visualization utilities," PyTorch, 2017. [Online]. Available: https://pytorch.org/vision/main/models/generated/torchvision.models.detection.maskrcnn_resnet50_fpn.html. [Accessed: April, 2024].

[7] PyTorch, "TorchVision Object Detection Finetuning Tutorial," PyTorch, 2017. [Online]. Available: https://pytorch.org/vision/main/models/generated/torchvision.models.detection.maskrcnn_resnet50_fpn.html. [Accessed: April, 2024].

[8] S. Eppel, "Train Mask R-CNN Net for Object Detection in 60 Lines of Code," Towards Data Science, 2022. [Online]. Available: https://towardsdatascience.com/train-mask-rcnn-net-for-object-detection-in-60-lines-of-code-9b6bbff292c3. [Accessed: April, 2024].