Convolutional Neural Networks for Visual Analysis of Clothing

Anmol Gupta [1] **Viswanathan Subramanian** [1] **Pei-Chen Wu** [1]

Abstract

In recent years, visual analysis of clothing trends has received a lot of attention. Recognizing apparel products and their attributes enables enhanced shopping experience for customers. Understanding the combination of clothing and material choices is essential for personalized recommendations and also important to improve the work efficiency of fashion professionals. The iMaterialist Fashion 2020 challenge combines the clothing recognition task with instance segmentation and attribute classification tasks. We extended the Mask R-CNN (He et al., 2017) implementation in Detectron2 (det, b) framework for our work. Specifically, we added an attribute classification network branch in parallel to the category classification and box regression tasks. We experimented with different loss functions and found focal loss (Lin et al., 2017) to significantly improve our results. We experimented with different backbones and various hyperparameters to fine tune our results. Given the unprecedented challenges we faced this quarter, the best results we got based on Kaggle's evaluation criteria (kag, c) was a mean Average Precision (mAP) of 14%.

1. Introduction

We live in a world where personalization is everywhere - music, ad, tv shows, etc. Visual analysis of clothing trends enables enhanced personalized recommendations in fashion. For our project, we chose to work on the iMaterialist Fashion 2020 challenge (kag, a), which is a fine-grained segmentation and attribute classification task for fashion and apparel. The challenge is meant to aid both fashion designers and consumers.

In our project, we built on top of the latest research in the field of computer vision, which uses Convolution Neural Networks (CNNs), to solve the kaggle challenge. The vision community has developed several powerful baseline systems, such as Fast/Faster/Mask R-CNN (Girshick, 2015; Ren et al., 2015; He et al., 2017), Fully Convolutional Networks (Long et al., 2014) and Feature Pyramid Networks (Lin et al., 2016) for object detection and instance segmentation tasks. As part of the project, we explored these existing architectures and used the Mask R-CNN system for the segmentation task, and added a network branch for the attribute classification task.

[1]Stanford University. Correspondence to: Anmol Gupta <anmolg@stanford.edu>, Viswanathan Subramanian <vsub@stanford.edu>, Pei-Chen Wu <pcwu1023@stanford.edu>.

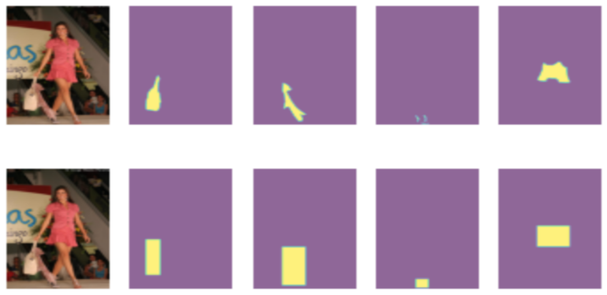
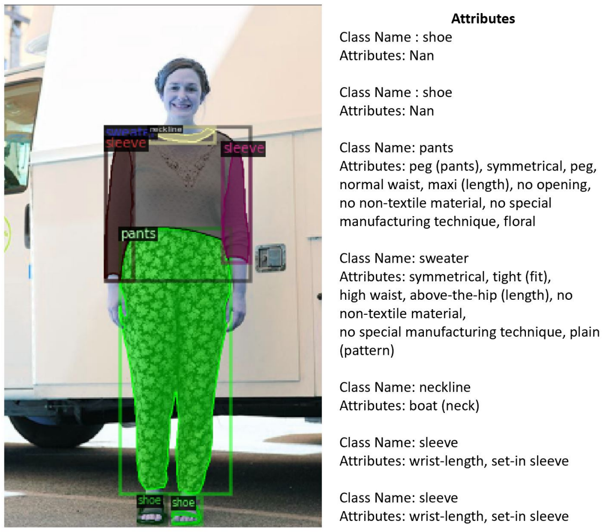


Figure 1. Dataset Example of Instance Segmentation and Object Detection. Categories (Left to Right): Bag, Scarf, Shoe, Skirt

1.1. Fashion Dataset

Kaggle provides a dataset (kag, b), which has a large number of images and fashion/apparel segmentations. For 48.8K images, the dataset columns contain an unique id per image and the information about masks in run-length encoded format. In addition, for each mask, a class id is specified, which represents the apparel category, and attribute ids are specified for each attribute present within the mask. Class id includes details about the supercategory, which specifies which part of the body the apparel belongs to. Attribute ids include details and in-depth information about the clothing item. Examples of attributes include textile, manufacturing, pattern, etc. The dataset contains images with 46 apparel classes and 294 related fine-grained attributes. The attribute class ids are not contiguous and there are missing numbers between 0 and 340. Figure 1 shows an example from the dataset with its instance segmentation and object detection. Figure 2 shows an example of attributes with each object in the image.



Attributes

Class Name : shoe
Attributes: Nan

Class Name : shoe
Attributes: Nan

Class Name: pants
Attributes: peg (pants), symmetrical, peg, normal waist, maxi (length), no opening, no non-textile material, no special manufacturing technique, floral

Class Name: sweater
Attributes: symmetrical, tight (fit), high waist, above-the-hip (length), no non-textile material,
no special manufacturing technique, plain (pattern)

Class Name: neckline
Attributes: boat (neck)

Class Name: sleeve
Attributes: wrist-length, set-in sleeve

Class Name: sleeve
Attributes: wrist-length, set-in sleeve

Figure 2. Dataset Example of Attributes for each object. Here Nan refers to no attributes for the objects

1.2. Network Architecture and Frameworks

We started with Detectron2 (det, b) framework, and were able to train a model starting with the pre-trained weights available for Mask R-CNN implementation with Resnet-50 (He et al., 2015) FPN backbone by changing the number of classes to 47. We did the necessary data pre-processing to use our data with the detectron framework. We used the Torchivsion (tor, b) framework to get a firm understanding of the network pipeline. Once we gained sufficient insight, we modified the detectron framework and added the attribute classification task. We tried different loss functions for the new task to improve the score. We found focal loss to significantly improve the performance of the task.

1.3. Evaluation

We used the evaluation metric provided by Kaggle (kag, c). Segmentation is scored using Intersection over Union (IoU) of a predicted set of object pixels and a set of true object pixels. Attribute Classification is evaluated using F1 score on each segmentation mask. IoU and F1 score pairs are selected on multiple specified thresholds. For each segmented object, an average precision is calculated over all the thresholds. The final score is the mean of this average precision of each segment in each image in the test dataset. We modified the Coco evaluator (pyc) to add the Kaggle evaluation metric.

1.4. Hyperparameter Tuning and Results

We experimented with different MaskRCNN backbones: Resnet50-FPN, Resnet101-FPN, ResNeXt-101-32x8d FPN, loss functions: BCE and Focal Loss, and several hyperparameters: Attribute classification layers, object and attribute class score thresholds. Our mean Average Precision with both IoU and F1 score increased from 5% to 14%.

2. Related Work

Jia et al. (Jia et al.) used a modified version of Faster R-CNN model to produce clothing proposals and category classifications, and add extra branches for attribute detection. It extended the Faster R-CNN object detection framework with ResNet 101 and ROI-align with two modifications, pruning mechanism and additional clothing attributes branches parallel to category branch. For evaluation of category prediction, they considered two metrics: (i) average precision (AP) per class and weighted mAP. (ii) CorLoc per class and weighted mean CorLoc. For each image, they pick the top 5 detections per image and check if the ground truth is correctly detected per class with a matching IoU threshold of 0.5 with ground truth boxes. CorLoc is computed as the ratio of number of detected ground truth instances over number of total ground truth instances per class, and weighted mean CorLoc is such ratio regardless of classes.

Ye et al. (Ye et al., 2019) targets imbalanced data distribution, particularly attributes with only a few positive samples. Ji et al. (Ji et al., 2018) proposed a semantic locality-aware segmentation model, which adaptively attaches an original clothing image with a semantically similar auxiliary exemplar by search.

DeepFashion2 (Ge et al., 2019) is a large-scale fashion image benchmark with comprehensive tasks and annotations, which contains 491K images and has a novel Match R-CNN to solve four tasks, such as clothes detection, pose estimation, segmentation, and retrieval.

Shiva (Shiva, 2018) focused on training the modified Mask-RCNN, fine-tune ResNet and all the layers, etc.

3. Method

3.1. Baseline

There are two tasks in the challenge, segmentation and attribute classification for each segment. For the first task of segmentation, we start with one of the popular segmentation networks Mask R-CNN (He et al., 2017) with Resnet-50 (He et al., 2015) FPN backbone by changing number of object segmentation classes to 47.

In order to understand the low level implementation better, we started exploring Pytorch Torchvision (tor, b). We modified the Mask R-CNN implementation in torchvision and added the attribute classification task. Torchvision doesn't have all the capabilities that are available from detectron2 out-of-box. So, we added code for data pre-processing to get the training to work. We used both torchvision and detectron based Mask R-CNN implementation for our baseline.

For attribute classification on each of the segmented image, we started with adding a single FC layer in the torchvision network implementation. R-CNN implementation has a module box predictor that outputs class scores and bounding box predictions. We added a third layer parallel to these two with 1024 inputs and 341 outputs. We started with publicly available Kaggle notebooks that uses detectron and torchvision, and made the above mentioned changes (det, d).

3.1.1. Baseline Loss Function

To compute the loss for our baseline, we used binary cross entropy loss with sigmoid activation for attribute classification. Mask R-CNN computes multiple losses: bounding box loss, classification loss, mask loss, RPN bounding box loss, RPN classification loss.

3.2. Attribute Classification

We faced some issues with the torchvision implementation both during training and evaluation. The training was really slow, and we kept running out of memory on GPU. We were not able to complete the training on a larger dataset size for even a single epoch. We then switched to the detectron implementation and added the attribute classification to the detectron framework.

3.2.1. Attribute Data Pre-processing

Number of attributes for each object is between 0 and 14 in the training dataset. For cases with no attributes, we added a background class id 294 to attributes. Also, to make sure we have same sized attribute tensors, we took a fixed-sized tensor of 14 and padded it with background class id for each segment.

3.2.2. Attribute Dataset

Standard dataset fields (det, a) available with detectron don't support attributes field for segmentation tasks. We added the custom attribute structure and dataset mapper. We referred to some publicly available examples and notebooks (det, d; Neverova et al., 2019) to add attributes field to detectron framework. We derived attributes structure from boxes data structure in detectron and added some helper functions to extract attribute tensors and other fields during loss and evaluation phases. We extended detectron's default dataset mapper (det, e) to include this new Attribute structure.

3.2.3. Attribute Classification Network

For Attribute classification, we added a new module and called it trend-rcnn. This module is derived from fast rcnn module. This module adds a new branch for attribute classification in the box predictor module along with class prediction and box regression with its own loss and evaluation functions. We added new roi head classes that instantiates trend-rcnn. We experimented with multiple networks for attribute classification. First network has 1 FC layer and the second one has 2 FC layers with ReLU activation. For each of these networks, we select the number of output neurons with and without object class. We used the detectron configuration system (det, c) to control these four options from Jupyter notebook itself.

3.2.4. Attribute Classification Loss and Prediction

In addition to detectron Mask R-CNN loss, we used BCE loss (tor, a) for multi-label attribute classification. In the code, we added options to exclude cases from loss computation where ground truth has no label for the segment and studied its performance impact. We also studied the performance impact of using Focal Loss (foc) (Lin et al., 2017). For attribute class prediction, we used sigmoid function on logits to get class scores. We added attribute score threshold configuration option as a hyper-parameter to improve the predictions. This option is also useful to control background class prediction.

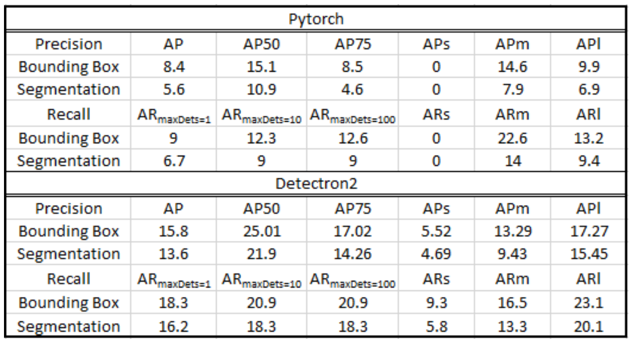
3.3. Evaluation

We made necessary changes to the Coco Evaluator (pyc) to implement the Kaggle evaluation metric. IoU computation for segmentation task already existed in the original implementation. We added F1 score computation for attributes found in ground truth object and prediction object that had an IOU score greater than the specified threshold. F1 score is calculated only for ground truth objects that have attributes otherwise its set to 1. For average precision computation, true positives were considered only when the the single predicted object matches the ground truth object with both IoU and F1 score above the respective thresholds. We report both with and without F1 score thresholding to understand how much the attribute classification task performance affects the final score.

4. Dataset

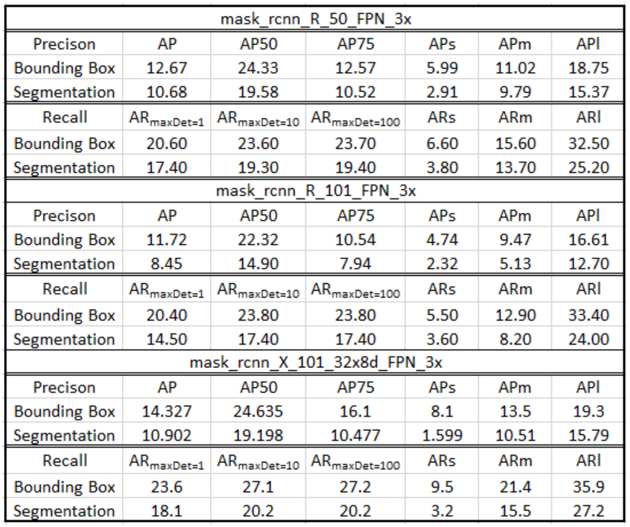
4.1. Torchvision Baseline Dataset

As part of data pre-processing for our torchvision baseline, we used the given run-length encoding, height and width to create binary image mask and get bounding box for each segment. To expedite training process, we resized images



Pytorch						
Precision	AP	AP50	AP75	APs	APm	APl
Bounding Box	8.4	15.1	8.5	0	14.6	9.9
Segmentation	5.6	10.9	4.6	0	7.9	6.9
Recall	$AR_{maxDets=1}$	$AR_{maxDets=10}$	$AR_{maxDets=100}$	ARs	ARm	ARl
Bounding Box	9	12.3	12.6	0	22.6	13.2
Segmentation	6.7	9	9	0	14	9.4
Detectron2						
Precision	AP	AP50	AP75	APs	APm	APl
Bounding Box	15.8	25.01	17.02	5.52	13.29	17.27
Segmentation	13.6	21.9	14.26	4.69	9.43	15.45
Recall	$AR_{maxDets=1}$	$AR_{maxDets=10}$	$AR_{maxDets=100}$	ARs	ARm	ARl
Bounding Box	18.3	20.9	20.9	9.3	16.5	23.1
Segmentation	16.2	18.3	18.3	5.8	13.3	20.1

Figure 3. Torchvision and Detectron2 Baseline Segmentation Scores without f1



mask_rcnn_R_50_FPN_3x						
Precison	AP	AP50	AP75	APs	APm	APl
Bounding Box	12.67	24.33	12.57	5.99	11.02	18.75
Segmentation	10.68	19.58	10.52	2.91	9.79	15.37
Recall	$AR_{maxDet=1}$	$AR_{maxDet=10}$	$AR_{maxDet=100}$	ARs	ARm	ARl
Bounding Box	20.60	23.60	23.70	6.60	15.60	32.50
Segmentation	17.40	19.30	19.40	3.80	13.70	25.20
mask_rcnn_R_101_FPN_3x						
Precison	AP	AP50	AP75	APs	APm	APl
Bounding Box	11.72	22.32	10.54	4.74	9.47	16.61
Segmentation	8.45	14.90	7.94	2.32	5.13	12.70
Recall	$AR_{maxDet=1}$	$AR_{maxDet=10}$	$AR_{maxDet=100}$	ARs	ARm	ARl
Bounding Box	20.40	23.80	23.80	5.50	12.90	33.40
Segmentation	14.50	17.40	17.40	3.60	8.20	24.00
mask_rcnn_X_101_32x8d_FPN_3x						
Precison	AP	AP50	AP75	APs	APm	APl
Bounding Box	14.327	24.635	16.1	8.1	13.5	19.3
Segmentation	10.902	19.198	10.477	1.599	10.51	15.79
Recall	$AR_{maxDet=1}$	$AR_{maxDet=10}$	$AR_{maxDet=100}$	ARs	ARm	ARl
Bounding Box	23.6	27.1	27.2	9.5	21.4	35.9
Segmentation	18.1	20.2	20.2	3.2	15.5	27.2

Figure 4. Detectron2 Segmentation Different Backbones

to min and max size of 400 and 600 pixels because the original dataset had images with a wide range of height and width, ranging from 269 to 8688 and 151 to 10717 respectively. We split the dataset into train, validation and test set. For training, we randomly flipped the training images and ground-truth horizontally for data augmentation.

4.2. Detectron2 Dataset

In later model development and training, we have created our own json file to grab 5K images from the original kaggle dataset in order to help us develop and test our attribute classification networks. The dataset has 294 unique attribute classes. In the training dataset, each object has a maximum of 14 attributes. The data loader creates a list of 14 indices, where missing attributes are replaced with classid 294 which we referred to as background class.

5. Experiments

5.1. Preliminary Segmentation Results

We tried both torchvision and detectron implementations of Mask-RCNN for segmentation in the beginning for evaluating which framework would work the best for our project. After running several experiments for both detectron and torchvision, we decided to use detectron for our final project.

5.1.1. Torchvision or Detectron2 Baseline

Figure 3 shows the precision and recall for both segmentation and bounding box. Torchvision was easier to understand, however, training was really slow compared to detectron2. The evaluation function for torchvision reference tries to create index on entire validation set which caused it to crash every time. We could only run it on a small dataset of 1000 images for a few epochs. Bigger dataset was running too slow.

5.1.2. Different Backbones for Segmentation

We ran several experiments on smaller (3K) kaggle dataset for different backbones (det, f), the evaluation results are presented in figure 4. We observed that mask-rcnn-X-101-32x8d-FPN-3x performs the best, it fits training data well. However, training costs too much time. Training mask-rcnn-R-101-FPN-3x also needs pretty long time. We decided to focus on mask-rcnn-X-101-32x8d-FPN-3x, since it has the best performance and mask-rcnn-R-101-FPN-3x has much shorter training time for attribute classification.

5.2. Attribute Classification

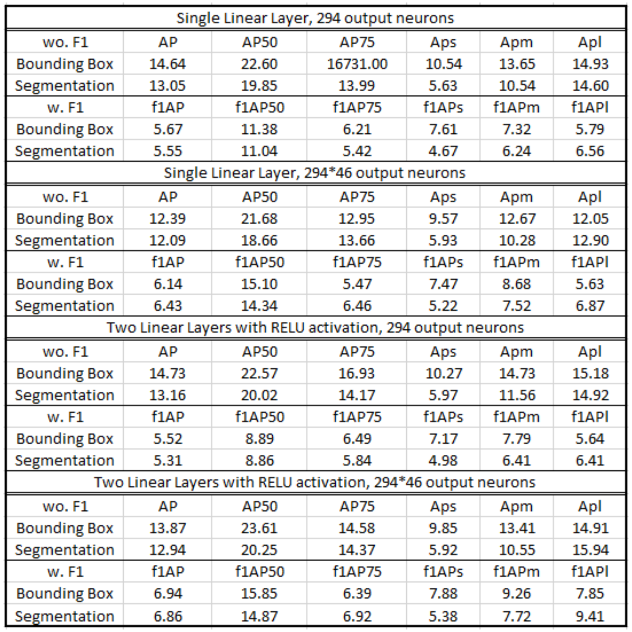
We used detectron to experiment with different attribute classification implementations on Mask-RCNN with Resnet-50 FPN backbone. We trained the model on dataset with 5000 training images and 500 validation images, ran for 8 epochs. The score and attribute threshold are set to 0.5 .

5.2.1. Different Classification Networks

We have implemented 2 different classification networks on the top of detectron MaskRCNN network. One is a single FC layer and the other is 2 FC layers with ReLU activation. Each of network was tried with different number of output neurons. To compute the loss, we used BCE loss. The results are presented in figure 5.

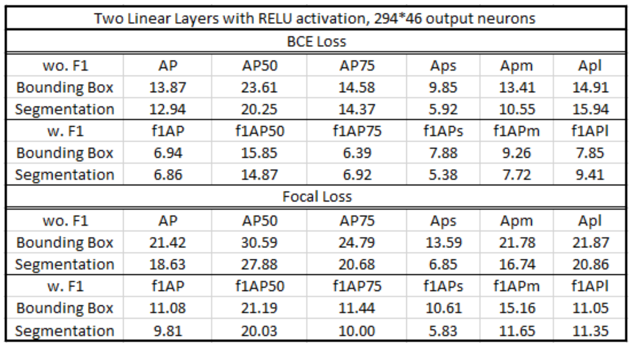
5.2.2. Different Loss Function

We also tried focal loss which showed improvements over BCE Loss. We excluded samples with 0 ground truth from loss computation. The results shown in figure 6.



Single Linear Layer, 294 output neurons						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	14.64	22.60	16731.00	10.54	13.65	14.93
Segmentation	13.05	19.85	13.99	5.63	10.54	14.60
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	5.67	11.38	6.21	7.61	7.32	5.79
Segmentation	5.55	11.04	5.42	4.67	6.24	6.56
Single Linear Layer, 294*46 output neurons						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	12.39	21.68	12.95	9.57	12.67	12.05
Segmentation	12.09	18.66	13.66	5.93	10.28	12.90
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	6.14	15.10	5.47	7.47	8.68	5.63
Segmentation	6.43	14.34	6.46	5.22	7.52	6.87
Two Linear Layers with RELU activation, 294 output neurons						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	14.73	22.57	16.93	10.27	14.73	15.18
Segmentation	13.16	20.02	14.17	5.97	11.56	14.92
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	5.52	8.89	6.49	7.17	7.79	5.64
Segmentation	5.31	8.86	5.84	4.98	6.41	6.41
Two Linear Layers with RELU activation, 294*46 output neurons						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	13.87	23.61	14.58	9.85	13.41	14.91
Segmentation	12.94	20.25	14.37	5.92	10.55	15.94
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	6.94	15.85	6.39	7.88	9.26	7.85
Segmentation	6.86	14.87	6.92	5.38	7.72	9.41

Figure 5. Different Attribute Classification with BCE Loss



Two Linear Layers with RELU activation, 294*46 output neurons						
BCE Loss						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	13.87	23.61	14.58	9.85	13.41	14.91
Segmentation	12.94	20.25	14.37	5.92	10.55	15.94
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	6.94	15.85	6.39	7.88	9.26	7.85
Segmentation	6.86	14.87	6.92	5.38	7.72	9.41
Focal Loss						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	21.42	30.59	24.79	13.59	21.78	21.87
Segmentation	18.63	27.88	20.68	6.85	16.74	20.86
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	11.08	21.19	11.44	10.61	15.16	11.05
Segmentation	9.81	20.03	10.00	5.83	11.65	11.35

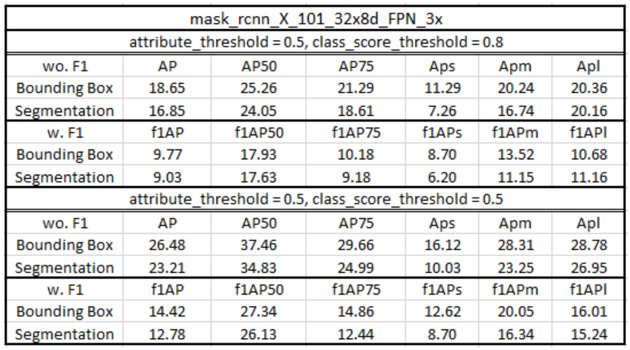
Figure 6. BCE Loss vs. Focal Loss

5.2.3. Different Thresholds

We tried with different attribute threshold and class score threshold on 500 images in validation set. We have presented the results in figure 7, 8.

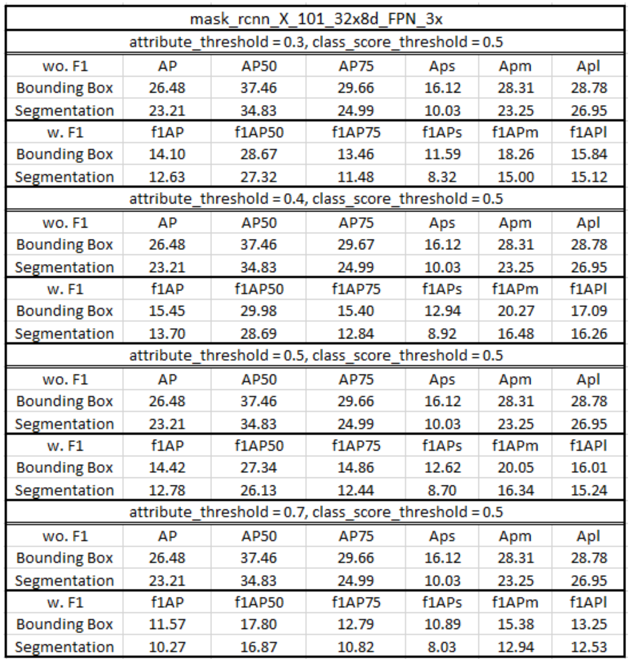
5.2.4. Final Network Result

Figure 10 was trained on 40000 images with input minimum size 800, maximum size 1333, learning rate 0.00025 for 2 epochs. We used SGD and Nesterov Momentum. For attribute classification, we used double linear layer with RELU activation function and 46*294 neurons for output. We show the results for two backbones, ResNeXt-101-32x8d FPN and ResNet50 FPN. Figure 9 shows the losses for Mask-RCNN with ResNeXt-101-32x8d FPN backbone. Figure 11 shows the predicted attributes for 1 sample image



mask_rcnn_X_101_32x8d_FPN_3x						
attribute_threshold = 0.5, class_score_threshold = 0.8						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	18.65	25.26	21.29	11.29	20.24	20.36
Segmentation	16.85	24.05	18.61	7.26	16.74	20.16
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	9.77	17.93	10.18	8.70	13.52	10.68
Segmentation	9.03	17.63	9.18	6.20	11.15	11.16
attribute_threshold = 0.5, class_score_threshold = 0.5						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	26.48	37.46	29.66	16.12	28.31	28.78
Segmentation	23.21	34.83	24.99	10.03	23.25	26.95
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	14.42	27.34	14.86	12.62	20.05	16.01
Segmentation	12.78	26.13	12.44	8.70	16.34	15.24

Figure 7. Attribute Threshold 0.5 - Different Class Score Threshold



mask_rcnn_X_101_32x8d_FPN_3x						
attribute_threshold = 0.3, class_score_threshold = 0.5						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	26.48	37.46	29.66	16.12	28.31	28.78
Segmentation	23.21	34.83	24.99	10.03	23.25	26.95
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	14.10	28.67	13.46	11.59	18.26	15.84
Segmentation	12.63	27.32	11.48	8.32	15.00	15.12
attribute_threshold = 0.4, class_score_threshold = 0.5						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	26.48	37.46	29.67	16.12	28.31	28.78
Segmentation	23.21	34.83	24.99	10.03	23.25	26.95
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	15.45	29.98	15.40	12.94	20.27	17.09
Segmentation	13.70	28.69	12.84	8.92	16.48	16.26
attribute_threshold = 0.5, class_score_threshold = 0.5						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	26.48	37.46	29.66	16.12	28.31	28.78
Segmentation	23.21	34.83	24.99	10.03	23.25	26.95
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	14.42	27.34	14.86	12.62	20.05	16.01
Segmentation	12.78	26.13	12.44	8.70	16.34	15.24
attribute_threshold = 0.7, class_score_threshold = 0.5						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	26.48	37.46	29.66	16.12	28.31	28.78
Segmentation	23.21	34.83	24.99	10.03	23.25	26.95
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	11.57	17.80	12.79	10.89	15.38	13.25
Segmentation	10.27	16.87	10.82	8.03	12.94	12.53

Figure 8. Different Attribute Threshold

and Figure 12 shows segmentation results from the trained network with ResNeXt-101-32x8d FPN backbone.

6. Conclusion and Future work

6.1. Conclusion

We present instance segmentation and attribute classification for fashion dataset in kaggle (kag, a). Our final implementation uses detectron. We added custom dataset field and experimented with multiple networks for attribute classification. For evaluation we extended Coco evaluator to compute F1 scores for attributes as well. Given the GPU resource constraints and GCP issues, we could only manage to run the experiments for a few epochs on the complete dataset. We got best results for Mask RCNN with ResNeXt-101-

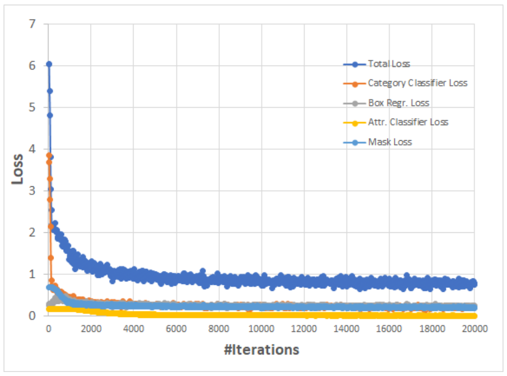
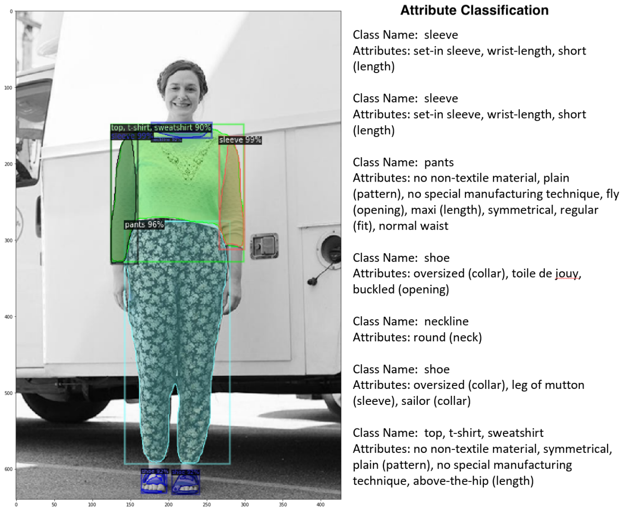


Figure 9. Training Loss plot

mask_rcnn_R_50_FPN_3x						
attribute_threshold = 0.4, class_score_threshold = 0.3, 2000 validation images						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	27.72	42.69	31.28	20.19	27.61	29.52
Segmentation	24.67	37.63	26.58	10.82	20.88	28.69
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	12.49	21.05	14.30	14.00	17.92	12.69
Segmentation	11.56	19.05	12.39	7.75	14.01	13.49
mask_rcnn_X_101_32x8d_FPN_3x						
attribute_threshold = 0.4, class_score_threshold = 0.3, 2000 validation images						
wo. F1	AP	AP50	AP75	Aps	Apm	Apl
Bounding Box	26.99	39.19	29.89	14.70	24.96	29.41
Segmentation	23.74	35.78	25.48	7.58	18.64	27.94
w. F1	f1AP	f1AP50	f1AP75	f1APs	f1APm	f1APl
Bounding Box	15.97	31.02	15.82	11.49	17.14	17.71
Segmentation	14.23	29.04	13.62	6.31	13.16	17.28

Figure 10. Trained on 40000 images and 2000 validation images for evaluation



Attribute Classification

Class Name: sleeve
Attributes: set-in sleeve, wrist-length, short (length)

Class Name: sleeve
Attributes: set-in sleeve, wrist-length, short (length)

Class Name: pants
Attributes: no non-textile material, plain (pattern), no special manufacturing technique, fly (opening), maxi (length), symmetrical, regular (fit), normal waist

Class Name: shoe
Attributes: oversized (collar), toile de jouy, buckled (opening)

Class Name: neckline
Attributes: round (neck)

Class Name: shoe
Attributes: oversized (collar), leg of mutton (sleeve), sailor (collar)

Class Name: top, t-shirt, sweatshirt
Attributes: no non-textile material, symmetrical, plain (pattern), no special manufacturing technique, above-the-hip (length)

Figure 11. Result from Model trained on 40000 images

32x8d FPN backbone and two fully connected layers for attribute classification. On 2000 images in the validation set, we could an average precision of 14 for segmentation

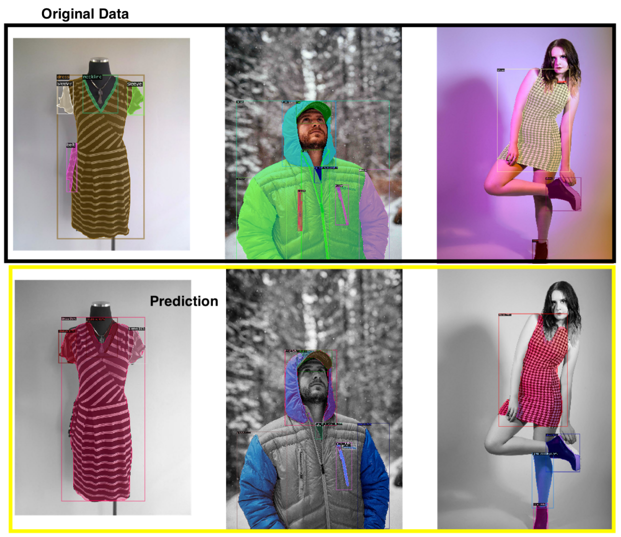


Figure 12. Result from Model trained on 40000 images

with attribute classification averaged over multiple range of thresholds.

6.2. Future Work

We want to run the same experiments for more epochs on the entire dataset. We would like to try different attribute classification techniques and have deeper pipeline with convolutional layers as in the mask predictor branch of Mask R-CNN. We want to improve our evaluation code to handle ground truth with no attributes in a better way. We want to improve the loss functions with hard-negative mining so similar attributes are classified better. We also want to try different attribute thresholds for different subcategories of attributes based on object class.

7. Contributions of team members

All the team members have contributed equally to this project through very strong collaboration. We had discussions and brainstorming sessions together. We divided the work amongst ourselves for the research and development to go in parallel as much as possible. All of us worked in some capacity on all the pieces.

References

Detectron2 Standard Dataset. https://detectron2.readthedocs.io/tutorials/datasets.html, a.

Detectron: Facebook AI Research's next generation software system that implements state-of-the-art object detection algorithms. https://github.com/facebookresearch/detectron2, b. Accessed: 2020-04-26.

Detectron2 Config. https://detectron2.readthedocs.io/tutorials/configs.html, c.

iMaterialist Detectron2. https://www.kaggle.com/julienbeaulieu/imaterialist-detectron2/notebook, d.

Detectron2 Dataset Mapper. https://detectron2.readthedocs.io/_modules/detectron2/data/dataset_mapper.html#DatasetMapper, e.

Detectron2 Model Zoo. https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md, f.

Focal Loss Implementation. https://www.kaggle.com/c/tgs-salt-identification-challenge/discussion/65938.

iMaterialist (Fashion) 2020 at FGVC7 - Fine-grained segmentation task for fashion and apparel. https://www.kaggle.com/c/imaterialist-fashion-2020-fgvc7, a. Accessed: 2020-04-25.

iMaterialist (Fashion) 2020 at FGVC7 - Dataset. https://www.kaggle.com/c/imaterialist-fashion-2020-fgvc7/data, b. Accessed: 2020-04-26.

iMaterialist (Fashion) 2020 at FGVC7 - Evaluation. https://www.kaggle.com/c/imaterialist-fashion-2020-fgvc7/overview/evaluation, c. Accessed: 2020-04-26.

COCO API - http://cocodataset.org/. https://github.com/cocodataset/cocoapi/tree/master/PythonAPI/pycocotools. Accessed: 2020-04-26.

Pytorch BCE Loss. https://pytorch.org/docs/master/generated/torch.nn.BCEWithLogitsLoss.html, a.

Pytorch Torchvision. https://pytorch.org/docs/stable/torchvision/index.html, b. Accessed: 2020-05-22.

Ge, Y., Zhang1, R., Wu, L., Wang, X., Tang, X., , and Luo, P. Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. *arXiv preprint arXiv:1901.07973v1*, 2019.

Girshick, R. B. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL http://arxiv.org/abs/1504.08083.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

Ji, W., Li, X., Zhuang, Y., Bourahla, O. E. F., Ji, Y., Li, S., and Cui, J. Semantic locality-aware deformable network for clothing segmentation. *IJCAI*, 2018.

Jia, M., Zhou, Y., Shi, M., and Hariharan, B. A deep-learning-based fashion attributes detection model. https://arxiv.org/ftp/arxiv/papers/1810/1810.10148.pdf.

Lin, T., Dollár, P., Girshick, R. B., He, K., Hariharan, B., and Belongie, S. J. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. URL http://arxiv.org/abs/1612.03144.

Lin, T., Goyal, P., Girshick, R., He, K., and and, P. D. Focal Loss for Dense Object Detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. URL http://arxiv.org/abs/1411.4038.

Neverova, N., Novotny, D., and Vedaldi, A. Correlated uncertainty for learning dense correspondences from noisy labels. 2019.

Ren, S., He, K., Girshick, R. B., and Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

Shiva, B. Evaluating mask r-cnn performance for indoor scene understanding. *Stanford CS231N Final Project*, 2018.

Ye, Y., Li, Y., Wu, B., Zhang, W., Duan, L., and Mei, T. Hard-aware fashion attribute classification. *arXiv preprint arXiv: 1907.10839*, 2019.

A. Appendix

Code base

https://github.com/visu227/cs231n-project.