Clothing Detection using Convolutional Neural Networks and Pose Estimation

Henry Lin
University of Toronto
1006164366

Abstract

In this paper, we explore a simple, computationally efficient method of pixel-wise image segmentation for labeling clothing and people in images using modern techniques and methods while keeping the construction simple and straightforward.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

1 Introduction

With the rise of social media, the fashion and design industry has been turned on its head. Influencers now more than ever have to concern over their outer appearance and style knowing they have hundreds of thousands of eyes watching to see who will find the next trend, the next thing, the next fashion statement.

Along with this, computer vision techniques have witnessed remarkable advancements, enabling applications across various industries such as self driving vehicles, medical imaging, and even in fashion. In this paper, we will take a look at image segmentation models with the goal of labeling portions of images based off the article of clothing contained in the iamge

2 Data Augmentation

The dataset used for training in this paper is from "Kota Yamaguchi, M Hadi Kiapour, Luis E Ortiz, Tamara L Berg, “Parsing Clothing in Fashion Photographs”, CVPR 2012.". Due to time and compute constraints, for reducing model complexity and required compute per iteration. The data was augmented to be resized to 128 by 128 which reduced overall accuracy of the model, however, this reduced training time by up to 4x due to running out of gpu compute power on cloud computing solutions and having to use a laptop CPU.

Additional methods to save compute power was calculating canny edges and pose estimation overlays and appending them onto the dataset’s images. Pose estimation was calculated with the use of Tensorflow’s pretrained model movenet_thunder_f16.tflite, the more precise version of movenet compared to movenet_lightning however both models are claimed (1) to be capable of 30+ FPS on modern hardware including laptops and phones.

Below is an example of the same image throughout the possible datasets. From left to right, we have the human segmented dataset, the canny edge dataset, the original dataset, and the dataset with both canny edges and pose estimation overlaid.

Figure 1: Sample image from each dataset

Specifically, if we take a look at the canny edges example, one can observe the issue with downsizing the original image.

From the figures below, one can observe that the small sizing of the images reduces the clarify and precision of canny edge detection being portrayed on the image significantly. Using the canny edge dataset, random artifacts would appear causing a decrease in accuracy due to the spastic shapes in the image caused by the downsizing.

3 Model Architecture and Design Decisions

The CNN architecture is rather large, below is a text representation of the layer network

3.1 Model Architecture

```
    Input (Shape: input_shape)
    |
    |--> Conv2D (32 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> Conv2D (32 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> MaxPooling2D (2x2 pool size)
    |
    |--> Conv2D (64 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> Conv2D (64 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> MaxPooling2D (2x2 pool size)
    |
    |--> Conv2D (128 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> Conv2D (128 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> MaxPooling2D (2x2 pool size)
    |
    |--> Conv2D (256 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> Conv2D (256 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> MaxPooling2D (2x2 pool size)
    |
    |--> Conv2D (512 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> Conv2D (512 filters, 3x3 kernel, ReLU activation, padding='same')
    |
    |--> UpSampling2D (2x2 size)
    |--> Conv2D (256 filters, 2x2 kernel, ReLU activation, padding='same')
    |--> Concatenate (along axis 3)
    |--> Conv2D (256 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> Conv2D (256 filters, 3x3 kernel, ReLU activation, padding='same')
    |
    |--> UpSampling2D (2x2 size)
    |--> Conv2D (128 filters, 2x2 kernel, ReLU activation, padding='same')
    |--> Concatenate (along axis 3)
    |--> Conv2D (128 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> Conv2D (128 filters, 3x3 kernel, ReLU activation, padding='same')
    |
    |--> UpSampling2D (2x2 size)
    |--> Conv2D (64 filters, 2x2 kernel, ReLU activation, padding='same')
    |--> Concatenate (along axis 3)
    |--> Conv2D (64 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> Conv2D (64 filters, 3x3 kernel, ReLU activation, padding='same')
    |
    |--> UpSampling2D (2x2 size)
    |--> Conv2D (32 filters, 2x2 kernel, ReLU activation, padding='same')
    |--> Concatenate (along axis 3)
    |--> Conv2D (32 filters, 3x3 kernel, ReLU activation, padding='same')
    |--> Conv2D (32 filters, 3x3 kernel, ReLU activation, padding='same')
    |
    |--> Conv2D (7 filters, 1x1 kernel, Sigmoid activation)

Output (Shape: (None, height, width, 7))
```

3.2 Design Decisions

For human segmentation, the last 2D convolution is changed to only have 1 filter and the final 128 by 128 by 1 output image is threshold-ed at 0.5 for its predictions.

The model was trained on many variations of the original dataset resulting in a variation of the model for each. It was trained on the original dataset, a canny edge dataset where the canny edges where overlaid onto the original, a pose estimation dataset where the pose estimation was overlaid onto the original, a combination of the two where both edges and poses where overlaid, and a final revision where the dataset would be first ran through the human segmentation model, and the output would then have canny edges or pose estimation overlaid.

Not all combinations of the last dataset and model was possible due to computation constraints and the negative effect of canny edges when scaled down.

4 Results

4.1 Human Segmentation

The human segmentation model is the Convolutional Neural Network defined above with the small change where the layer has 1 filter instead of 7. The model was trained on a 256 by 256 image set with canny edge detection and pose estimation overlaid on top. Because the model's output is relatively linear, only having 2 classifications, and the model can use the edges and pose estimation to predict what is human, the model was trained off a modified dataset where canny edges and pose estimation was overlaid on top of the original image, which was then resized to 256 by 256.

We can see from Figure [1] that the model tends to predict rounded shapes. Although I suspect this to be due to the significant reshaping of the image, resulting in significantly less information for the model to learn off of. However, even with less information to work off of, the model was able to obtain a 95.6% training accuracy and a 95.95% validation accuracy. Accuracy in this case being as computed as simply the number of correctly classified pixels divided by the total number of pixels in the dataset.

Comparing the results to the original paper, we see that our model has approximately the same accuracy for human segmentation compared to the complete model of the paper comparing 95.95% to 95.6% although they have a 0.4% margin. This is rather impressive given that our model runs on a 128 by 128 input image compared to a 600 by 400 input image. The trained model in this paper runs of less than 1/14th of the input size compared to the original model.

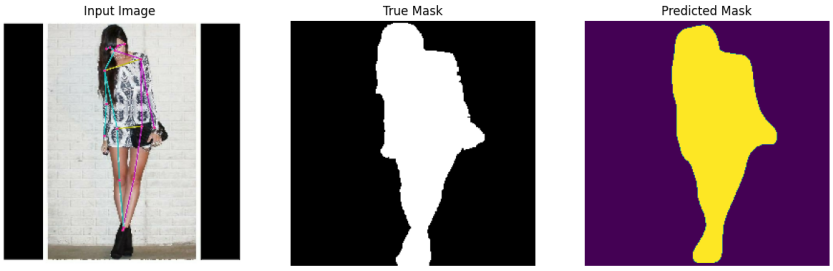
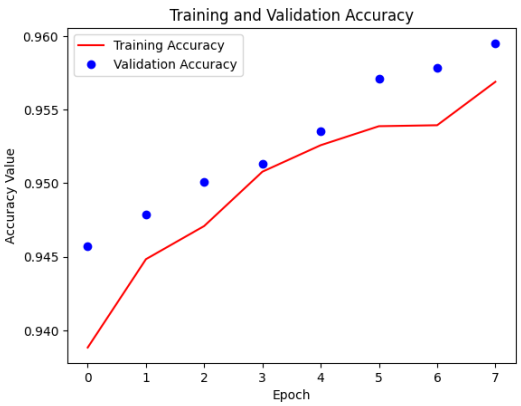
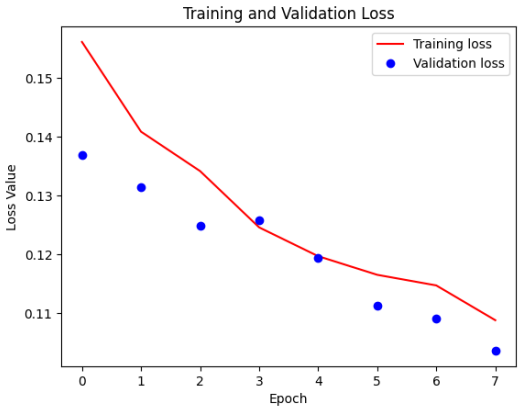


Figure 2: Example output of Human Segmentation CNN model



(a) Figure 3.1 Training and Validation Accuracy



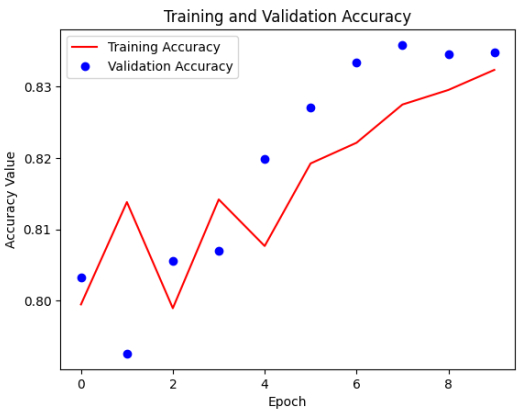
(b) Figure 3.2 Training and Validation Loss

4.2 Clothing Segmentation

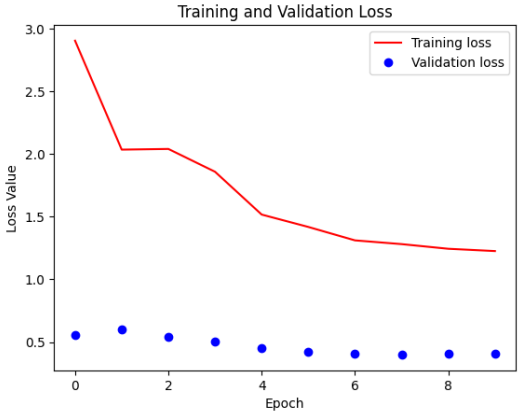
Due to the nature of the dataset, computing un-adjusted prediction accuracy for a model of this nature would be ill advised. This is because the majority of the images pixels in the datasets are classified as being the background while only a small proportion of pixels are labeled smaller pieces of clothing such as shoes compared to t-shirts and dresses. Because of this, the model's weights must be manually adjusted to account for this imbalance by counting the number of pixels in each class, taking the sum and weighing each class as its count divided by the sum and finally manually adjusting that value to produce optimal outputs.

From the above, we can see that the predicted mask tends to bled into the background. This is likely due to the fact that the weight of the background class was set to be anywhere from 3 to 5 times less than other classes to prevent the model from labelling the entire image as background (which results in a 77% accuracy according to the original paper, this number may change due to reshaping). Because the model weighs non background classes higher, it chooses to overestimate the clothing pixels to reduce its overall loss.

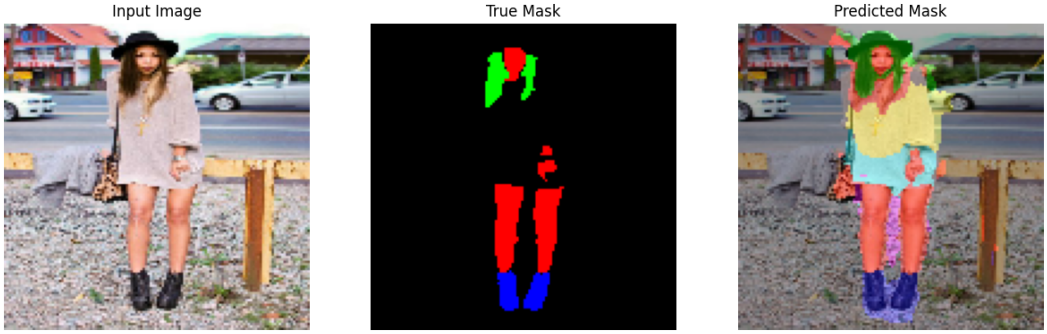
Although the predicted mask seems to predict rather inelegantly, the validation accuracy of the model is 83.5%, which is nearly 8% higher than the baseline solution and only 5% lower than the model presented in the original paper.



(a) Figure 4.1 Training Accuracy for Clothing



(b) Figure 4.2 Training Loss for Clothing



(c) Figure 4.3 Example Segmentation Output

5 Discussion

5.1 Limitations

There were many limitations during the research of this project. The first major limitation was compute power. Initially, all model training and dataset parsing was done within Google Colab, however, I quickly ran out of GPU compute tokens, thus forcing me to use the CPU which is incredibly slow, taking on average 20 minutes to train the model for 4 epochs. As this method of compute was incredibly inefficient, I was forced to replicate the environment on my laptop. Although it has a GPU, it also runs on Windows 11 which does not support native GPU runtime with TensorFlow. And due to other factors, using WSL was not a realistic method. Thus, I was forced to use my laptop CPU for inference. Although it is faster than Google Colab, it is by no means faster than a GPU.

The lack of compute also effected the usefulness of pose estimation and edge detection. Because the compute was lacking, I was forced to scale down the input images to 128 by 128 images from 600 by 400 otherwise the training time would be 4 to 7 times as long, which when ran a couple dozen times, would add up to a few hours. Attempting to train the model at full resolution for 10 epochs was estimated to take a little over 3 hours.

The last limitation was due to external time pressure. This project was released on March 28th. Since then, I have had on average 3 assignments due every week or tests and final exams. This left me with a little over 50 hours to complete this assignment from start to finish. Although this is not an excuse, it is worth mentioning due to the extreme nature of the time pressure.

Even with all these limitations and forced restrictions of computational complexity on the model, the model still performed admirably, only lagging behind the original model, achieving 95% of the original model for clothing segmentation, and and 99% of the original model for background, human segmentation

5.2 Key Findings and Future Work

There were two key findings in this research. The first is the in viability of the use of pose estimation or edge detection in combination with pixel wise segmentation for small images. Because of the image is relatively small, the canny edges, when overlaid on top of the original image and being reshaped down to a fourth of its original size, results in a pixelated image with slightly lighter pixels around edges. Although this does still achieve its purpose in providing the CNN information about the edges of the image,

A major area of improvement for the model would be increasing the resolution of the input. Because of aforementioned compute constraints, the dataset had to be downsized to reasonably be computed and tested within the limited time frame. Increasing the resolution likely significantly improves the performance of the model.

The second key finding in this research was a method used to reduce computational complexity of the model. Due to the overwhelming lack of compute power relative to time, the model had to be split up into 3 sub models, a human classifier, a pose estimator, and a clothing estimator. Each sub model was trained individually and separately to obtain good results independent from the other models. The sub models would then be linked together, or in this case, having the outputs be saved and ran through the subsequent model, to produce a small but good overall model.

This technique of divide and conquer is commonplace in computer science. However, I personally hadn't seen this technique, or a technique similar to this used to train a model from such small parts. Transfer learning would be the most similiar technique which I can recall where one uses a previously trained models existing inferences to train a new model.

6 Conclusion

In conclusion, through the use of clever tricks and techniques, this project produced a small, but efficient and easily trainable method and model for image segmentation of clothing. Although the performance of the model can still be significantly improved on, the model also can be seen as inspirational, only being trained on a CPU only slightly faster than the ones provided by Google Colab for free, showing that complex models in years past can be trained efficiently for a low cost in modern day and the creation of models isn't gated behind the latest GPU.

As for the content of the project itself, I learned many things from data augmentation due to the imbalance of the dataset, to the harsh reality of the compute required to train large scale models. In the future, I hope to refine this model by increasing its complexity and training using a GPU.

References

[1] Kota Yamaguchi, M Hadi Kiapour, Luis E Ortiz, Tamara L Berg, "Parsing Clothing in Fashion Photographs", CVPR 2012.

`http://vision.is.tohoku.ac.jp/~kyamagu/research/clothing_parsing/`

[2] TensorFlow, "MoveNet: Ultra Fast and Accurate Pose Estimation on Mobile Devices," TensorFlow Hub Tutorials

`https://www.tensorflow.org/hub/tutorials/movenet`

[3] TensorFlow, "Image segmentation," TensorFlow Tutorials

`https://www.tensorflow.org/tutorials/images/segmentation.`

[4] OpenAI, "OpenAI Chat," OpenAI Website, https://chat.openai.com/.

`https://chat.openai.com/`