1 Attention UNET for Fashion Segmentation Results

In this paper we will summarize the results of our investigation into using UNETs for fashion segmentation. A complete break down of the model architecture and post processing algorithms are provided in the companion paper along with more detailed explanations of mathematical concepts involved.

1.1 Dataset

Our model was trained on about 12,700 labeled images from the DeepFashion dataset. This dataset had 24 unique labels and featured models in a variety of poses. Since the pictures were taken using diffused studio lighting against a white backdrop, the model was not trained to segment out background information. Furthermore the distribution of labels for each image were greatly imbalanced. Inside of our test set containing 10% of the data, [tie] and [glove] did not appear once while [neckwear] appeared only a single time. In order to reduce the model complexity we did not vary the scale of the input images. Each portrait of 750 x 1101 pixels had a random 750 x 750 horizontal strip sampled and then downscaled to 512 x 512 before being processed by the network. As a result the network could not detect jewelry like chains and earings if they were too small. This choice was made intentionally to guarantee that the network had enough context information to identify clothing items like dresses and rompers which would by impossible to classify from looking at just the top or bottom half of the image (without the network memorizing specific outfits).

The network was trained using a gradient accumulation step of 64 samples (optimal batch size depending on the GPU RAM available) and ADAM optimizer with a β term of 0.9. As a result each epoch consisted of 179 gradient accumulation steps. We pretrained the network for 2 epochs using a simple bottleneck and a learning rate of 1e-3, then replaced the bottleneck with a robust attention bottleneck and trained just the bottleneck for half an epoch. Finally we trained the whole network for another 12 epochs at which point the training loss started outpacing the test loss. We acknowledge that having a combined test/validation set of just 1270 images means that it is not fair to compare the results of our model to others that use a 1/3, 1/3, 1/3 split and we will not do so here. We will however compare variants of our own architecture against each other.

1.2 Bags and Belts

For some reason instead of actual belts, the dataset labeled any clothing item tied around the waist as being a belt. This was not good for the upbringing of our model which has little idea of what a belt is supposed to be. Compounding on this, models often draped articles of clothing over their arms in place of a handbag and due to lighting it was not always clear which was which.

1.3 Results

We summarize the results for four different models, all values are median reported results. Pixel accuracy was computed by dividing correctly labeled pixels by total pixels. Subject accuracy was pixel accuracy restricted to non background pixels. Results were taken from the test set with rare labels excluded. Label scores were median intersection over union results restricted to samples where they appeared. This is similar to pixel accuracy except we penalize both false negatives and false positives (whereas pixel accuracy only penalizes false negatives). IoU scores are lower than pixel accuracy scores because error is double counted. Frequency is how often the label appeared amongst 1270 training samples. Proportion is the percentage of labeled pixels out of all labeled pixels in the training set.

If we consider the IoU score of a sample where the label does not appear, if the network does not predict that label to exist anywhere in the image we get an IoU score of 1, if we predict a single pixel to exist for that label anywhere in the image we get an IoU score close to 0 (depending on the smoothing parameter ϵ). Depending on how we tune the smoothing parameter, reporting these results either makes our model look much better than it actually is or much worse. For the sake of clarity we omitted these results entirely from the calculation.

	Proportion	Frequency	512_96	512_96_pp	512_64_xl	256_64_la
background	73.9616	1270	**0.996**	0.996	0.995	0.995
skin	5.9680	1270	0.935	**0.936**	0.914	0.910
top	5.1749	904	0.940	**0.943**	0.914	0.930
pants	4.9410	829	0.954	**0.958**	0.932	0.945
dress	3.3596	286	0.967	**0.968**	0.945	0.957
outer	2.4575	294	0.818	**0.836**	0.716	0.746
hair	1.1265	951	**0.833**	0.827	0.793	0.780
rompers	0.8339	66	0.962	**0.965**	0.773	0.918
skirt	0.6451	105	0.917	**0.934**	0.708	0.829
footwear	0.5047	619	0.768	**0.795**	0.718	0.733
face	0.3950	604	0.912	**0.913**	0.874	0.858
bag	0.1971	108	0.388	0.366	**0.731**	0.460
belt	0.1725	72	0.143	0.102	**0.274**	0.005
leggings	0.1015	35	0.763	**0.895**	0.672	0.807
wrist w.	0.0453	595	0.483	**0.494**	0.405	0.024
necklace	0.0396	348	**0.203**	0.186	0.173	0.017
head w.	0.0334	69	0.585	0.630	**0.639**	0.634
socks	0.0230	43	0.173	0.067	**0.403**	0.032
ring	0.0161	816	0.193	**0.205**	0.157	0.071
eyeglass	0.0033	17	0.503	0.541	**0.581**	0.015
pixel accuracy			0.988	**0.989**	0.981	0.983
subject accuracy			0.960	**0.963**	0.937	0.944

- 512_96: 512 x 512 pixel model, (96, 96, 192, 384, 768, 1536) feature scaling, 6 attention blocks in bottleneck, 4 convolutions in each upBlock. 560MB
- 512_96_pp: same as 512_96 except we perform post processing using predicted boundary data. 560MB
- 512_64_xl: 512 x 512 pixel model, (64, 128, 256, 512, 1024, 2048) feature caling, 12 attention blocks in bottleneck, 3 convolutions in each upBlock. 1800MB
- 256_64_la: 256 x 256 pixel model, (64, 128, 256, 512, 1024) feature scaling, 6 attention blocks in bottleneck, 3 convolutions + local attention in each upBlock. 300MB

1.4 Bottleneck Replacement for Accelerated Training

It was found imperically that at the beginning of training the model output had a fairly large dependence on the bottleneck. As the model trained, the importance of the bottleneck decreased while the importance of the residual connections greatly increased. It was found that a bottleneck consisting of just a couple convolutions could be trained with a high learning rate (1e-3) down to a mean cross entropy loss below 0.3 in just two epochs. During this time the network dependence on the bottleneck became fairly minimal. Afterwards the bottleneck could be swapped out with a much more robust bottleneck containing a stack of multihead attention and 1x1 convolution blocks. We then trained just the bottleneck for half an epoch (so that it could line up with the original model) before training the entire model as a whole. By using group norm and SiLU activation functions, it was possible to train the network at a rate of at least 1e-4 though it is very likely that the effective training rate can be pushed even higher. The model was then trained for 12 epochs before being tested. Since the majourity of the labeled data (90%) was used for training set we did not want to overfit.

We experimented with enabling block local attention in the UNET decoding phase (upblocks). Due to the massive training cost involved, this feature of the model was only enabled after finetuning the overall network and bottleneck. When training 256x256 versions of the model, it was found to reduce cross entropy loss by 20%. However it also increased the computational cost of training by a factor of 3-4. While training higher resolution versions of the model 512x512 we determined that we gained more performance by increasing the base feature size from 64 to 96 and increasing the number of convolution steps in each upblock by one (94% test model pixel accuracy for local attention version vs 96% for the thicker model).

Most UNETS double the number of features with each spatial halving. Since our initial investigations using 256x256 models (and 16 x 16 bottleneck) proved fairly robust, we decided to simply extend the model by adding a 512x512 spatial layer with same number of top level features (64 or 96). We experimented with a model that used a classic feature progression (64,128,256,512,1024,2056) and twice the attention blocks in the bottleneck but it ended up being 3 times larger and performed slightly worse than the (64,64,128,256,512,1024) local attention model. This experiment also suggests that the current bottleneck is not a limiting factor in our model.

1.5 Instance Normalization before Softmax

Normalizing each vector individually before passing to softmax increased initial training speed by about 5 times. We had accidentally implemented this in our earlier test by calling softmax on our output vectors before passing the result to the torch cross entropy function (which called softmax again). After fixing the bug we were surprised to find that training performance greatly degraded.

1.6 GroupNorm vs BatchNorm

Due to the size of the models involved, we cannot efficiently fit more than 2 or 4 images in each batch. As a result trying to use batchnorm in the model ends up reducing performance (and producing nonsense in eval mode) since the batch sizes are too small for the batch statistics to match the population statistics.

1.7 ReLU vs SiLU

We replaced BatchNorm with GroupNorm at the same time that we replaced ReLU with SiLU. We ended up using similar hyperparameter schedules as determined in the 256x256 case in order to train the larger 512x512 models. The fact that the we did not have to decrease the learning rates shows that our changes did relax the convergent hyperparameter domain. However we did not carry out any ablative study to determine how big of a difference replacing ReLU with SiLU in particular made.

1.8 Boundary Segmentation

The simple method to produce a segmentation given label probabilities for each pixel is to assign to each pixel the label of highest probability. While quick and easy, this method has a number of down sides. First of all the network might be able to segment out an article of clothing but have trouble reaching consensus as to what that article is. The network might end up thinking that pixels near the boundary of a pair of pants are slightly more likely to belong to a skirt. To fix this issue we implemented a post process step where a segmentation of the image into unlabeled regions is used to build a collection of masks. Those mask are then used to pool statistical

information from the segmentation and output one cohesive label for the mask. Any leftover pixels are then added to whichever mask has the most similar statistics or promoted to masks in their own right if their certainty is high enough (since jewelry is sometimes too small to generate a large enough mask in the first step to meet the threshold).

Ideally the network would output a perfect segmentation and we would have a single mask per connected label. In practise the network ends up predicting a lot of extrenuous boundaries. Even worse, the network will sometimes miss a boundary and as a result two different labels end up pooled together. Extra boundaries are not serious from a practical standpoint but they are a strong indicator that the network has trouble segmenting individual items (in other words the bottleneck needs to be more robust). Extra boundaries make the algorithm run a bit slower but don't make the output that much worse. If the network thinks that an item is actually two different items there isn't much more to do at that point. Missing boundaries on the other hand can have catastrophic concequences for our accuracy, especially when the [top] and [pants] are mixed together or a [top] and the [background]. We used a custom Binary cross entropy loss that penalized false negatives at 20 times the rate of false positives to greatly reduce the chance of missing boundaries at the expense of creating a lot of incorrect ones. Furthermore we used a simple expand contract algorithm that is capable of closing small pixel gaps (but fails when the boundary information becomes sufficiently noisy).

Using the ground truth gradient as the segmentation ground truth has a number of issues. The largest issue is that we are tasking the network with ignoring label self intersections. As an example the points where the edge of a sleeve overlaps with the interior of a jacket is clearly a physical boundary but is not a point on the segmentation gradient (since we are going from jacket pixels to jacket pixels). This means that we are trying to teach the network to completely ignore certain self evident boundaries which would not harm our decomposition at all if they were included. This makes the network much more likely to ignore boundaries that are critical to separating different clothing items.

Since jewelry in the mask ended up generating dense boundary information, implementing boundary detection in the model served to amplify detection of jewelry which was a very nice side effect. Since the boundary cross entropy loss and the label cross entropy loss are in some sense complimentary (information about one provides information about the other) we were not concerned with the coupling coefficient between them (ended up using 1). The coupling coefficient between loss terms is much more important when the network is more likely to make one term worse while making another better. An example where the coupling coefficient would be important is in a variational auto encoder where improving the reconstruction loss comes at the expense of making the latent variables less guassian like and vice versa). A more clear example is something like image sharpening where we want to make the image look more sharp without straying too far from what the original image looks like.

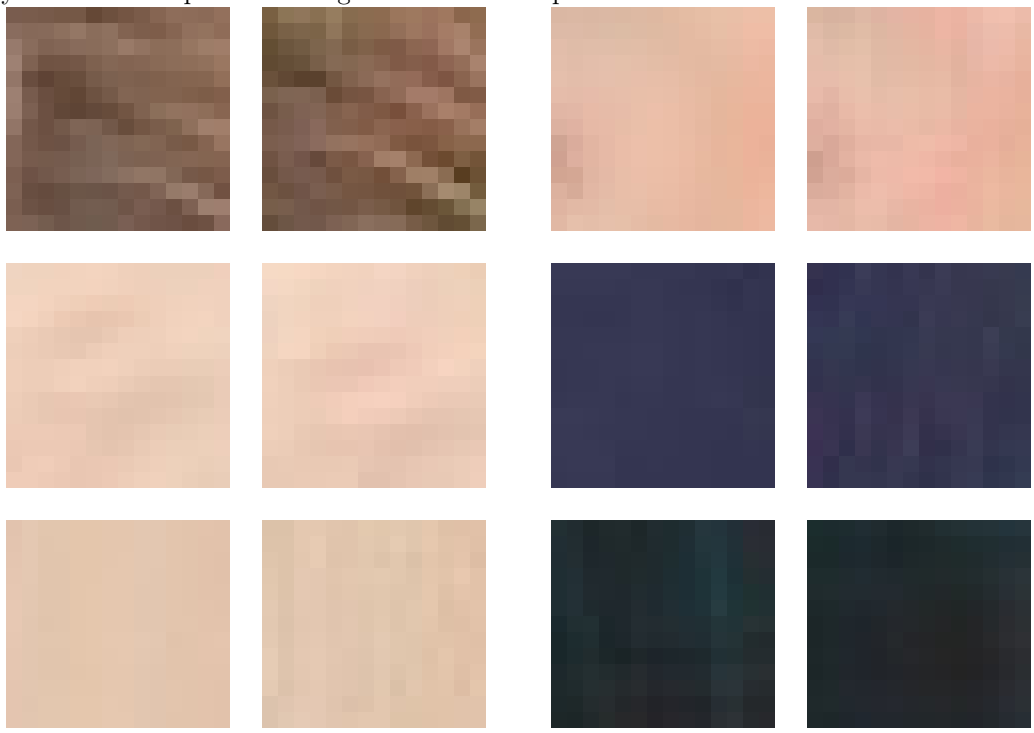
1.9 Local Attention

Our model using local attention ended up performing slightly better than xl model for large common labels despite having half the bandwidth, half the bottleneck depth, and half of both horizontal and vertical image resolution. On the other hand increasing resolution and feature size was much more effective when it came to identifying jewelry. The biggest drawback to local attention is that it has terrible performance scaling, enabling it increased training time by 3-4 times. We determined that increasing feature size and upBlock depth led to overall better results than trying to enable local attention for a 512 x 512 model given limited training resources. Enabling local attention in the 3 deepest layers might be viable but anything shallower would require an A100 GPU. Since local attention primarily improves consensus, it would be most valuable in accurately predicting the segmentation boundary.

If we were to scale this up to production, we would use a much bigger dataset (DeepFashion2), train a sister model to segment people out of backgrounds, increase the bottleneck depth to deal with additional data, and only then consider enabling local attention.

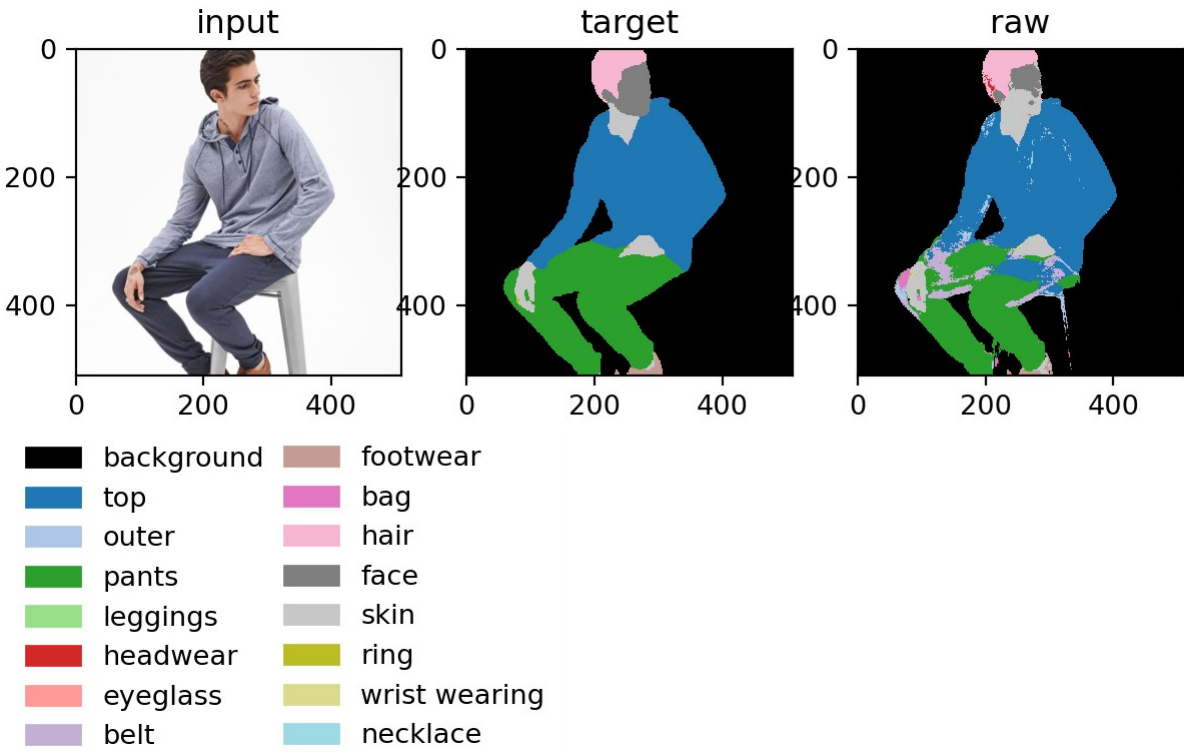
1.10 Running the Network Backwards

Instead of training the network so that the inputs match the desired outputs, we can fix the network and train the input to match the desired outputs. This exercise can give us some insights into how the network actually works and what it is specifically looking for when assigning labels. Performing 100 iterations with a learning rate of $1e-3$ is usually sufficient for the network to perfectly match the desired output. The change in the input image ends up very subtle and requires zooming in on individual pixels to tell the difference.

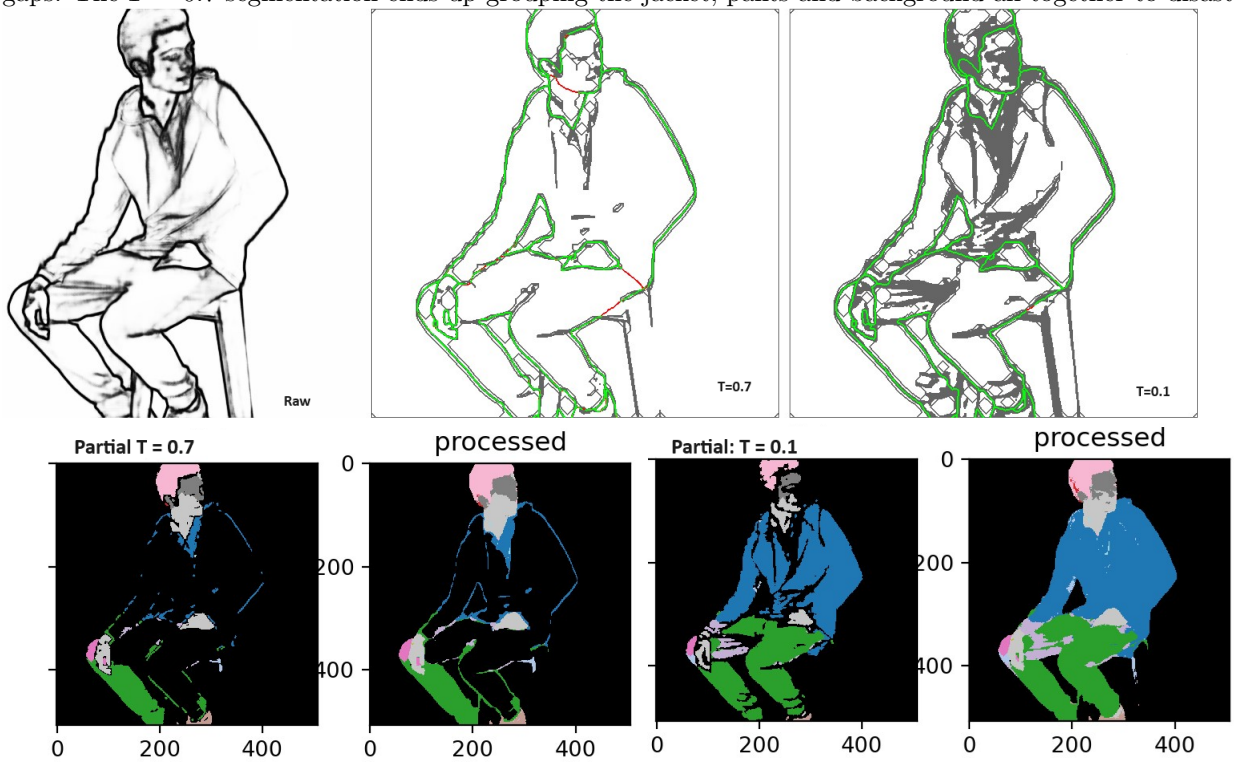


Samples from the original image appera on the left and the processed image appear on the right. They are from top to bottom, left to right: hair, skin, outerwear, face, top, pants. The effect is subtle but more uniform regions have a texture added that looks a little bit like jpeg artifacts. We believe that replacing visually uniform regions with textures gives the model a stronger signal for generating consensus since there will be a greater variety of actuated convolution masks. This suggests that using images of even higher resolution would improve output. There is also the danger that consistent studio lighting has taught the model to recognize certain labels by looking at subtle pixel patterns.

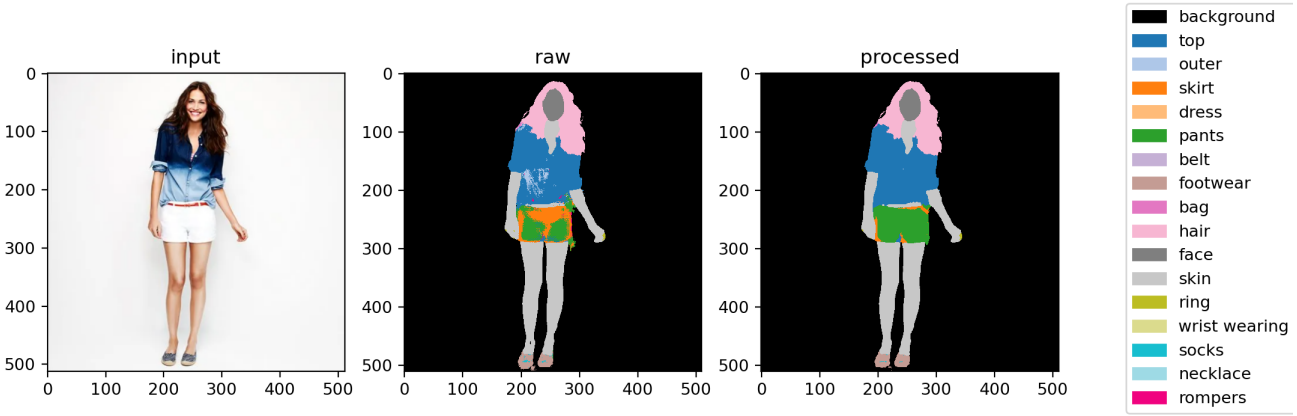
2 Boundary Segmentation Thresholding



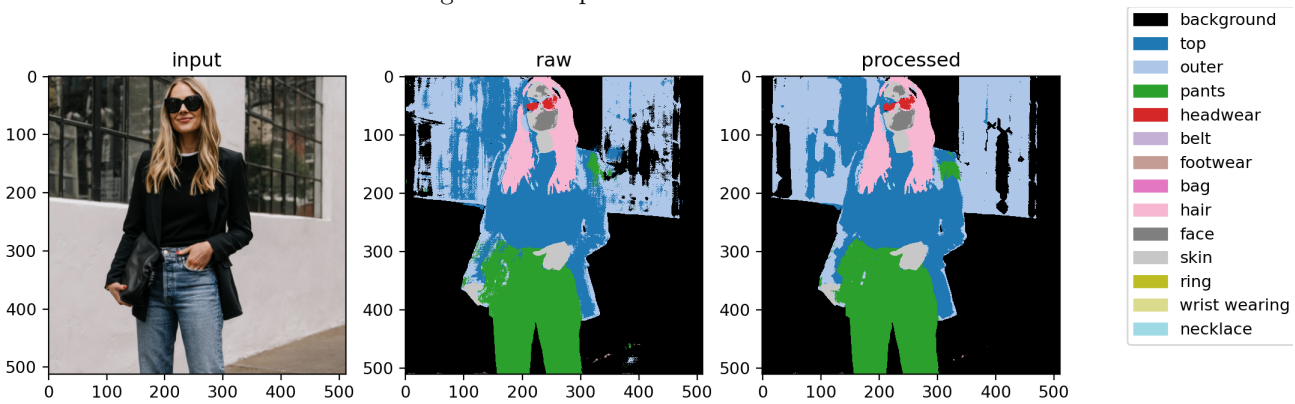
Pictured above is a test image that the model had particular trouble with. Below is the boundary segmentation as a grayscale image inverted for better clarity. Our basic sanitation steps is to first cut off all pixels below a certain treshold (0.7 in the middle and 0.1 on the right). Ground truth boundary segments are highlighted in green where they match and in red where they are missing. As you can see, aggressively excluding more pixels (T = 0.7) leads to far less false positives but it also leads to large gaps. The $T = 0.7$ segmentation ends up grouping the jacket, pants and background all together to disasterous results.



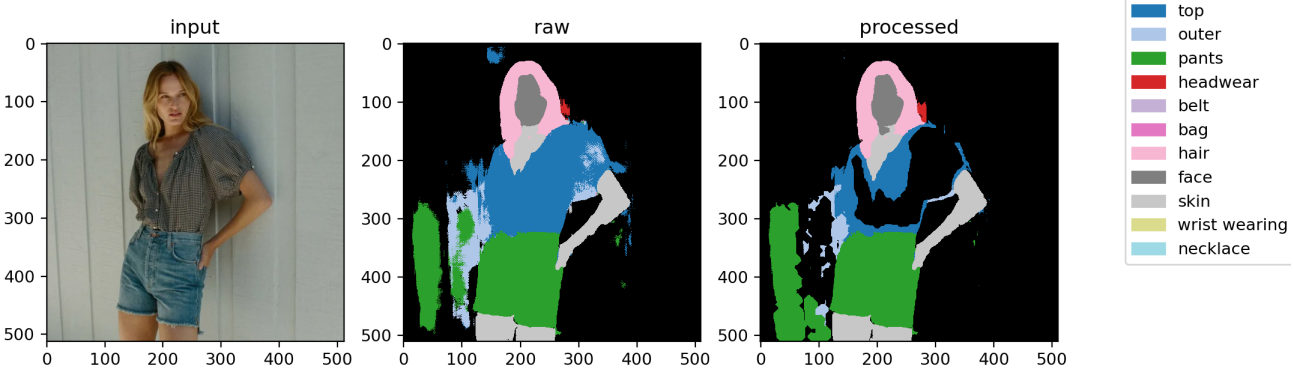
3 Results From Google Images



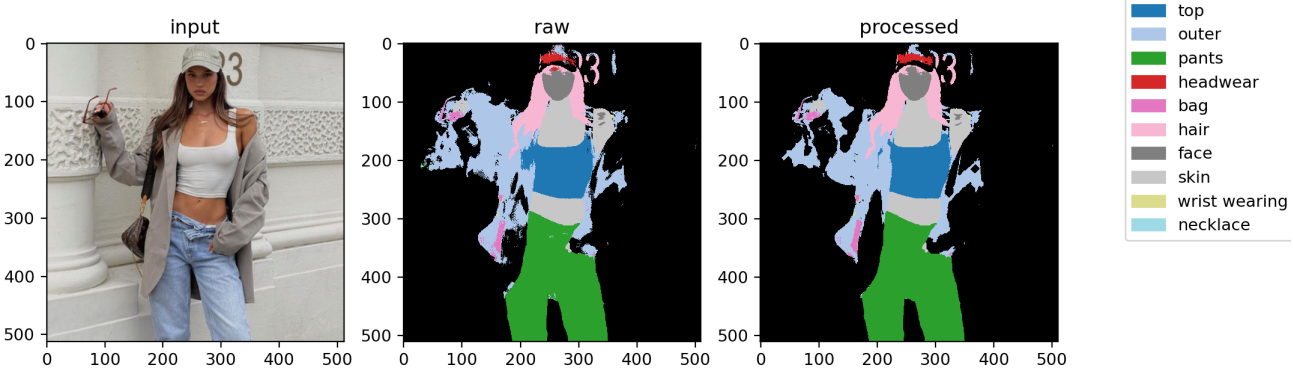
We chose this image since it had a diffused white background (the same as the model was trained on). However the scale of the image is significantly smaller than the training size (instead of about 70-80% of the model we have all of her). The model had some doubts about whether she is wearing a skirt or pants.



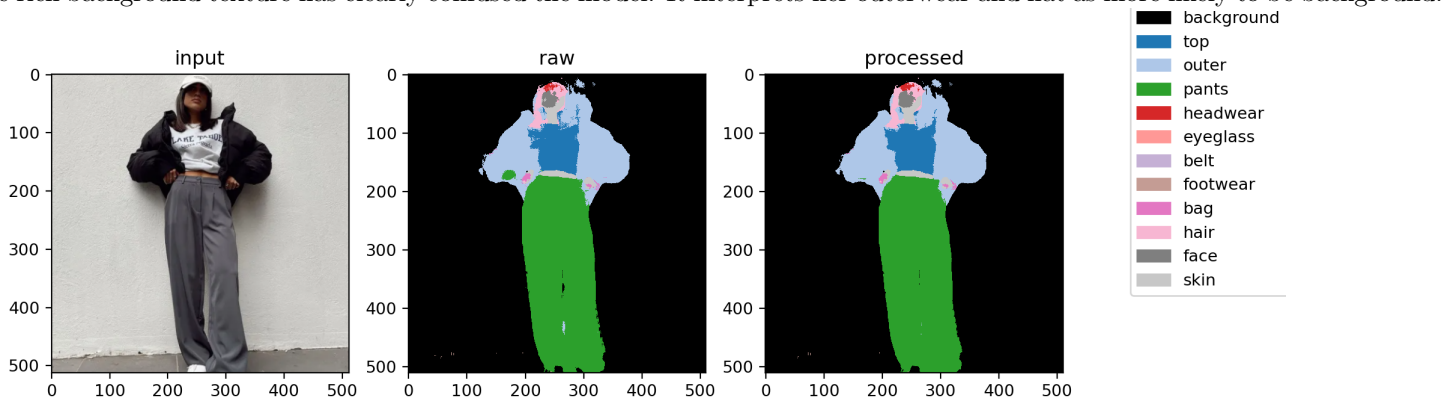
We chose this image because we wanted to see what the model would make of the noisy reflections and the bag. The windows obviously caused a great deal of confusion but they did not appear to corrupt the rest of the image too bady. The model had trouble with bags since the training data often had draped outer garments that looked very similar to bags.



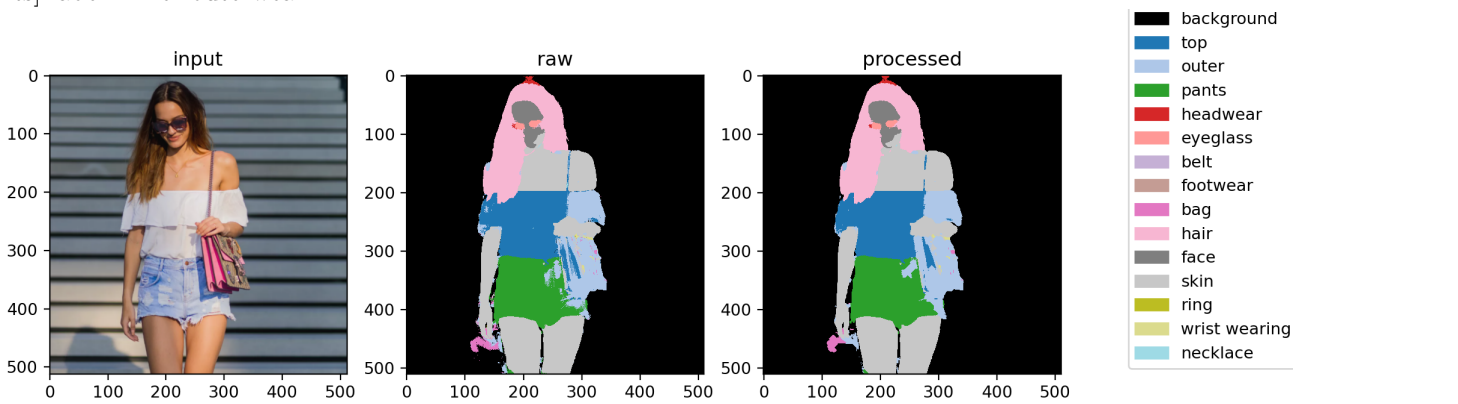
We chose this image to see if the model could handle a fairly neutral background. Ignoring the background, the model swas properly segmented, however the boundary was not properly inferred and the left side registered as pants.



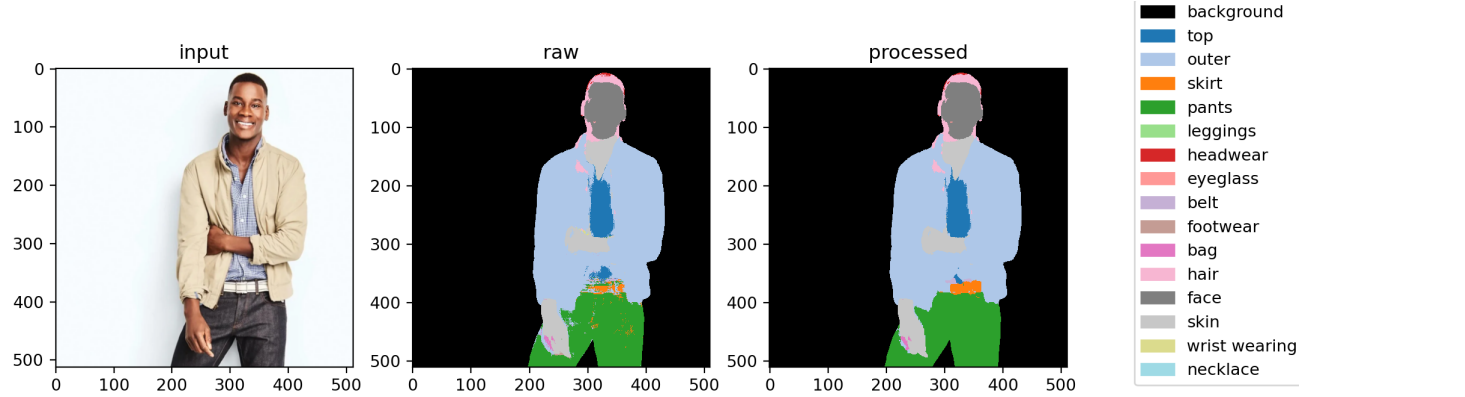
The rich background texture has clearly confused the model. It interprets her outerwear and hat as more likely to be background.



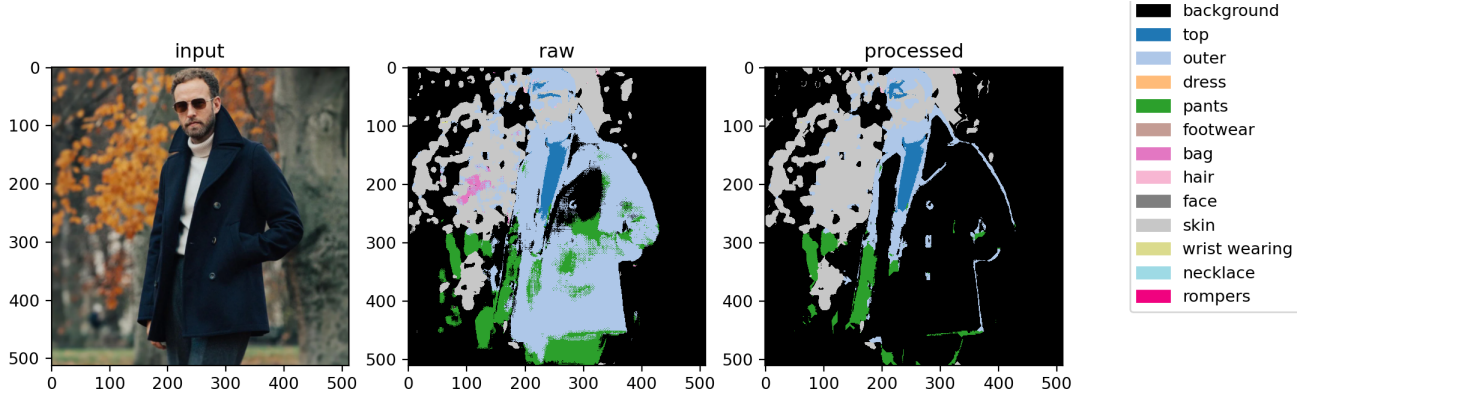
We chose this image to see how the model responds to an unfamiliar pose and insufficient lighting. The shadows around the hands end up registering as handbag while the face end up kind of a mess. The post processing ends up removing most of the [pants] label in her outerwear.



This was another easy one, the background being out of focus seems to help. The model detects glasses which are a rare label but has issue with the negative space created by her shadow and the handbag plus outer sleave is labeled as outerwear.



The good news is that the training data had a wide variety of skin tones so this particular image is handled fairly well. Unfortunately the training data did not have a good variety of belts so it ends up detected as a skirt.



Finally we chose an image with a particularly noisy background. The model interpretted the leaves as skin and that caused all kinds of issues.