Human Body Part Segmentation and Matching Between Poses

Shimi Smith

1. Introduction

Human pose matching is a problem made up of two parts. The first is commonly known as pose estimation and is the task of detecting the position of human body parts in an image. The second part is matching between two poses. This problem has useful applications such as motion tracking that could be then used for creating realistic animations and also has applications in health sciences for detecting gait abnormalities.

2. Relevant Work

Most research on this topic focuses on pose estimation. A significant challenge in pose estimation is estimating positions of joints that are occluded. For our project we are not considering this case and are only concerned with body parts that are not occluded. The typical approach to pose estimation as done in [1] produces 2D locations of anatomical keypoints such as joints. These keypoints are then combined for each person in the image to produce an estimation of the pose. This is in contrast to our method that uses a segmentation of the image. This works well for us since we are not trying to predict an underlying bone structure, rather to just match between limbs. Another recent research paper published is DensePose [2]. Rather than predicting keypoints like [1], they create a mapping of all pixels on the human body in the image to a 3D surface of the human body. We have no need to map to a 3D surface but we create a similar segmentation into body parts. Our method is computing a dense set of correspondences between body parts in two images but for visualization and robustness we refine the matches to a sparse set of the most reliable matches.

3. Method

Our method consists of two steps. First we use a segmentation network to predict a segmentation mask of our input images into human body parts. We then extract local features from our images and utilize the segmentation mask to create a matching between body parts in the two images.

3.1. Segmentation

3.1.1 Architecture

For segmentation we implemented the architecture from [3] with some modifications. The contractive part of the network is the VGG-16 architecture [4]. VGG-16 is the 16 layer variation of the VGG network trained on ImageNet [5] for object recognition. Rather than using the original VGG-16 architecture we used a modified version with batch normalization added after each convolution. As shown in [6] batch normalization speeds up training significantly by normalizing layer inputs throughout the network, using an approximation of mean and variance computed across a mini-batch. As a result of the proven succcess of batch normalization in training neural networks, we also added batch normalization after all other convolutional layers in the network. [3] used fixed bilinear interpolation kernels for upsampling in the expansive component of the network and in [7, 8] the authors described initializing the interpolation kernels to bilinear kernels and then learning the weights further. Additionally, they only used one interpolation kernel per output class. We decided to use learnable kernels for all interpolation to give the network full control over finding an optimal interpolation.

3.1.2 Training

As in [3] we trained our network on the Pascal Parts Dataset from [9] taking 80% for our training set and 20% for test. We combined the parts in the dataset into the following 6 body parts: left and right arms and legs, head and torso. Oliveira et al. [3] employ a layer-wise training approach. Since each of the refinement layers has N features where N is the number of classes we can consider the output of each of these layers as a segmentation mask. Thus we could first train the network with only one refinement layer, fix the weights of that layer and add another and continue training. This approach was proposed to speed up training as most training would not actually be done on the full network. An additional benefit is that the network could easily be extended further by adding additional refinement layers and training them individually. This could be useful to achieve better results for cases of fine grain segmentation where complex representations and higher expressiveness would

be required to achieve better results. A draw back of layer-wise training is that you have to train each layer for some fixed amount of time. [3] trains each layer for 2 days which is not feasible due to our time and resource constraints. Experimenting with lower training time would also be a time consuming iterative process. As a result we decided to train the entire expansive component of the network at once. We used the Adam optimizer [10] and began training with a learning rate of 0.001. Near the end of training we lowered our learning rate drastically to $1e-10$ to reduce oscillation. The contractive part of the network is initialized to the weights of VGG-16 and not trained further.

3.1.3 Segmentation Results

[3] presents IOU for the architecture trained on both 4 body parts and on 14. Due to the increased complexity of fine grained body part segmentation, as expected, they achieved lower IOU results with 14 body parts. They achived an average IOU over all body parts of 78.23% on 4 body parts and 71.7% on 14 body parts. Since we are predicting a segmentation at the granularity of 6 body parts we would except to see results somewhere between 78.23% and 71.7%. Our model only achieved an average IOU of 6.94%. Table 1 shows the IOU for each of our 6 body parts on the test set. Since [3] does not present individual IOU for the same body parts as us we cannot compare directly. Though we can see from the average IOU that our implementation preforms significantly worse than that of [3]. We warrant this significant difference in results to the difference in training time. [3] trained their network for 10 days with a state of the art GPU. We could not come close to that training time given the time constraints and hardware we are working with. Figure 1 shows our predictions versus the true segmentation mask for four different training examples. The first two examples are cases where the network does decently well and the second 2 are examples where it does worse or fails completely. In general, the network works best in cases where there is only one non-occluded person that is in the foreground. The second image is an ideal example. Here the only problem is that the two arms are getting reversed. We suspect that with longer training time the network would learn to distinguish left and right limbs better. The first image is another example of where the network preforms decently well, though it also illustrates a failure mode. The baby on the women's lap is not being detected fully which we suspect is due to having smaller body parts. In this example, the left and right arms and legs are consistently swapped. We can see that the network is able to distinguish different sides but has not learnt which is left and right yet. The third image, is a case where we have complex occlusion, where body parts are occluding other body parts. Examples like this are understandably challenging to learn. The last example in Figure 1 demonstrates how the network completely fails to detect small people who appear at a distance in the scene.

Head	Torso	Left Arm	Right Arm	Left Leg	Right Leg
5.59	5.27	4.95	6.35	10.45	9.06

Table 1. IOU of predictions for each of the 6 body parts

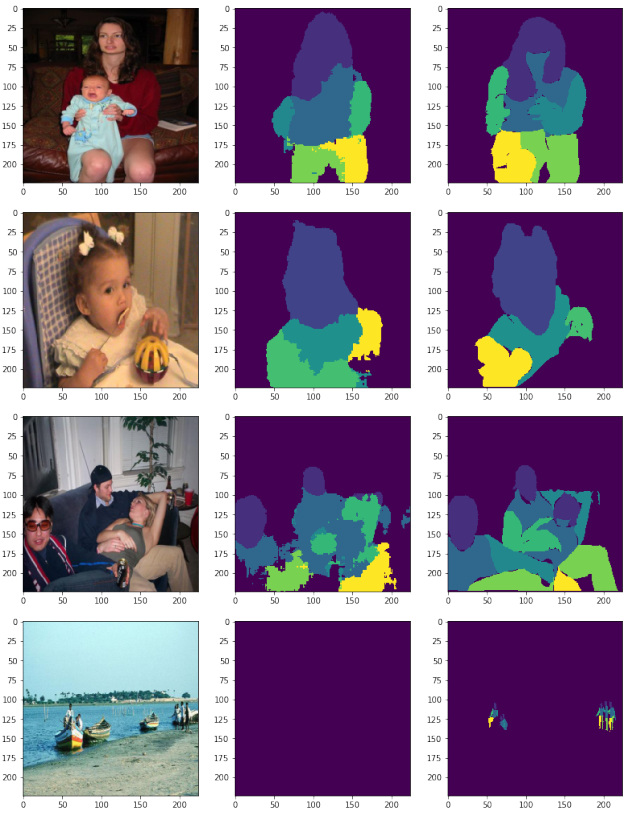

Figure 1. Test set examples. Input image, predicted mask and true mask.

3.2. Matching

Our approach to matching body parts between two images consists of first extracting meaningful features from the images and then utilizing our segmentation predictions to find a sparse set of matches between limbs.

3.2.1 Feature Extraction

Inspired by the success of VGG-16 for the contractive part of the segmentation network we decided to use it to extract local features from the image. We found that features from VGG16 resulted in us finding accurate sets of matches, though we suspect other deep convolutional architectures could be useful as well. The VGG16 architecture is illustrated in Figure 2. We can see that the VGG network pro-

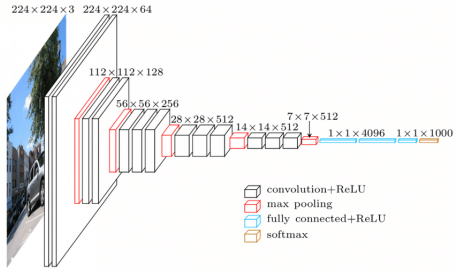

Figure 2. VGG16 architecture.

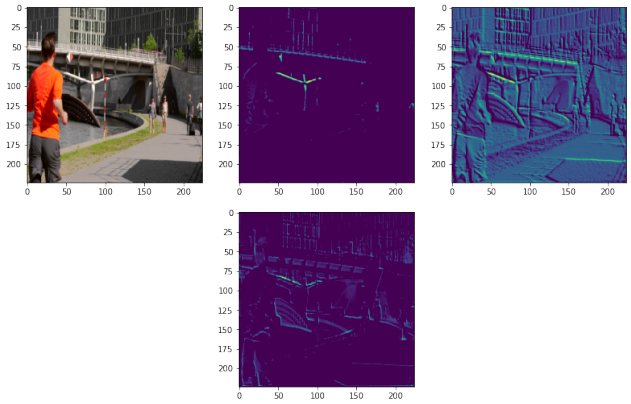

Figure 3. Image and the first 3 feature maps extracted from VGG16

duces a compressed represetntation of the input image into a volume of size $7 \times 7 \times 512$ through a series of convolutions and max pooling operations. This is a representation of the entire image which is useful for object recognition, whereas for feature matching we need local features across the entire image. We decided to take only the first two convolutional layers of the VGG16 network to compute local feature vectors with 64 dimensions each for every point in the input image. Figure 3 shows the first 3 features computed by VGG16. We can see that it responds to edges and significant structures in the image that would be useful for matching.

3.2.2 Key Point Selection

Our goal is to find a sparse set of reliable matches for each limb, thus we need to be selective when considering keypoints. We found success by defining a good key point to be a point in the image where some window around the point contains only pixels from the same body part as the point. This heuristic allowed us to find correct key points despite having poor segmentation masks by taking points whose predictions are most likely to be correct. This is based on the observation that points near the center of their class in the mask tend to be most reliable. The window size should be picked depending on the images being matched.

3.2.3 Matching

Let I_1 and I_2 to be the two images being matched. These could be any two images or frames from a video such as in our application. As mentioned in the previous sections we begin by selecting a sparse set of reliable key points in I_1 by comparing the segmentation class in some $k \times k$ window. For each key point in I_1 we scan I_2 and compare the features in patches around each key point. We only consider matches between points that are part of the same body part. We found that using the L1 norm of the difference was best because it found reliable matches and is not as computationally expensive as the L2 norm. The difference in computation time may seem insignificant but when our system is run on videos with a large amount of frames these optimizations make a larger difference.

3.2.4 Results

We present the results of our matching on frames of two videos. One of a man running in a park and another of a man rowing a boat. In both cases the window size used for refining keypoints is 11×11 and the patch size for matching is 7×7. For the running man video, a threshold of 300 was used for selecting matches and for the man rowing, a threshold of 500 was used. These parameters were selected by observing good visual results. Figure 4 shows the result of our algorithm. We can see that for the running man example, correct matches are found between matching body part classes. The segmentation does not fully capture the arms so no matches between arms are computed. In the rowing video our method correctly matches points on the head and torso and manages to match one of the hands. These examples illustrate how the matching is quite dependent on correct segmentation masks. By refining our keypoints as described in previous sections, we add robustness to noise in the segmentation mask since pixels in small areas of the mask will not be considered.

4. Challenges

The main challenge we faced was with training the network. Even though we simplified the training process we had to train the network for a long time to achieve the results we got. Even though we trained for a long time our segmentations do not come close to those of [3]. The key point selection we implemented was in part to add robustness to poor segmentations in addition to selecting a sparse set of matches.

5. Conclusion and Future Work

In summary, we have created a system that can match body parts between video frames. We did this by first predicting a segmentation mask of 6 different body parts. We

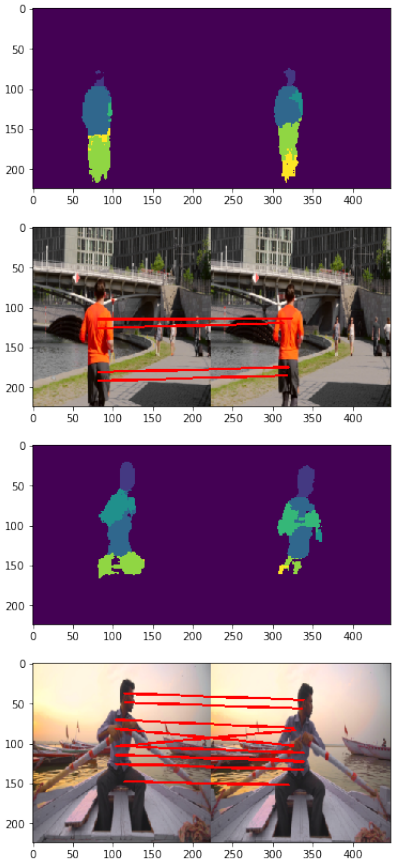

Figure 4. Matchings between video frames and the segmentation masks.

then used features extracted from the VGG16 network in combination with the predicted segmentation masks to find matches between video frames. We introduced an approach that refines the set of keypoints by rejecting keypoints that are not part of a reliable mask. This was done by enforcing a window around each keypoint to contain all pixels from the same class. Despite poor segmentation, we achieved meaningful matches between body parts. For future work we would primarily focus on improving the segmentation. We would experiment with training the network using the layer-wise approach and attempt to achieve better results. We would also like to investigate other features for the matching. The features from VGG16 preformed quite well for us but in the future we would like to investigate other options.

References

[1] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *CoRR*, vol. abs/1812.08008, 2018.

[2] R. A. Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7297–7306, 2018.

[3] G. Oliveira, A. Valada, C. Bollen, W. Burgard, and T. Brox, "Deep learning for human part discovery in images," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[8] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1605.06211, 2016.

[9] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. L. Yuille, "Detect what you can: Detecting and representing objects using holistic models and body parts," *CoRR*, vol. abs/1406.2031, 2014.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.