Extracting Kinematic Information Using Pose Estimation

Robbie Jones
Stanford University
rmjones@stanford.edu

Abstract

We present a preliminary step in building a lightweight system to extract kinematic information from a video file. Much of our work is based on the latest advancements in the domain of pose estimation. We begin with an examination of the current state of the art network for estimating human pose and then discuss methods for shrinking the network while maintaining accuracy. While they are simple, our experiments show great potential for achieving a substantially smaller network with comparable accuracy and open the door for more complicated methods to be used.

Figure 1. Example of output from Cao *et al.*'s convolutional network. Their work focused on improving multi-person pose estimation, but still performs very well on images of individual persons.

1. Introduction

Kinematic information, such as joint angles and joint velocities, are of great importance to fields interested in human motion. Athletes may want to have their jumping or throwing motions examined in order to assess injury risk or increase performance. A physical therapist may utilize such data while examining the gaits of rehabilitating patients. While relevant kinematic information can be obtained via motion sensor technology, such machinery is expensive and can be difficult to use. Thus, we seek a cheaper and more user-friendly alternative.

One possibility of recovering this information is through the use of inverse kinematics. The inverse kinematics problem takes as input the position of an object (possibly over a certain period of time given a sequence of images) and seeks to compute the necessary joint angles and movements to assume that position. It is a technique heavily used in the fields of robotics and computer animation, but we can apply the same approach to human movements. Therefore, what we desire is a system to take an image of a person and return coordinates of the person's joints. Once the person's position is predicted, a number of methods to solve the relevant kinematic equations can be used [3].

Moreover, possible applications of our system are worth noting. We aim to build a system that could fit on a mobile phone, which a user could utilize to film themselves walking, running, jumping, or performing some other motion, and quickly receive the relevant kinematic information they desire as the system performs all of its computations locally. Naturally, runtime, memory usage, and battery usage all become of supreme importance when considering possible solutions for this problem. While neural networks, especially convolutional nets, are notorious for their number of parameters and need for processing power, their recent involvement in computer vision breakthroughs cannot be ignored. More specifically, the use of convolutional networks in estimating human pose is of interest to us, as we can directly make use of recent work in this area to help solve our own problem of predicting joint locations from an image.

2. Related Work

The problem of "pose estimation", or predicting an object's position and orientation given an image, is nothing new to the field of computer vision [2, 6, 13, 20, 19, 12, 5].

However, only in recent years have convolutional neural networks emerged as powerful tools in solving this problem. Toshev and Szegedy [17] were some of the first to effectively utilize convolutional nets to estimate pose, basing their architecture on "AlexNet", which came from Krizhevsky *et al*. [10]. Their approach is relatively straightforward: their network takes an image, directly regresses it to joint coordinates, and then inputs these predictions into further convolutional networks to iteratively refine their network's predictions. This "cascading" technique of using further neural networks to refine predictions is paramount to the current state of the art networks in pose estimation. One of these networks comes from Wei *et al*. [18], who built upon [17] by introducing the idea of a "convolutional pose machine" which, instead of directly predicting Cartesian coordinates from an image, predicts a belief map of joint locations that is then refined by later convolutional networks. Predicting belief maps rather than explicit coordinates for each joint allows the network to better learn spatial dependencies between body parts, and it is an increasingly popular technique seen in recent pose estimation work [14, 15]. Most recently, Cao *et al*. [4] unveiled an architecture for estimating pose that has two branches, one for predicting confidence maps and one for predicting *part affinity fields* (detailed in the next section). Cao *et al*.'s work has achieved the highest accuracy on both the COCO 2016 keypoints challenge dataset [11] as well as the MPII human multi-person dataset [1], so it is their model which interests us for this problem.

3. Method

A more detailed overview of Cao *et al*.'s network is warranted because it forms the basis for our experiments. Figure 2 shows the entire pipeline of their model.

3.1. Part Affinity Fields

What separates this network from others in the field is the definition and use of *part affinity fields* (PAFs) in the estimation process. These PAFs were introduced in order to resolve the ambiguity in estimating the poses of multiple people in an image. Although we are not directly interested in computing the entire pose of a person, these part affinity fields are utilized in computing the confidence maps for joint locations, so it is worthwhile to understand how they work and what they mean.

Cao *et al*. define PAFs for each limb as a "2D vector [encoding] the direction that points from one part of the limb to the other." More rigorously, the part affinity field at a point $\mathbf{p}$ for a given limb is represented by the equation

$$\mathbf{L(p)} = \begin{cases} \mathbf{v} & \text{if } \mathbf{p} \text{ on limb} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

where $\mathbf{v}$ is the unit vector pointing in the direction of the limb (from one joint to the other).

3.2. Network Overview

The first ten layers of the network, initialized to be the first ten layers of VGG-19 [16], takes an input image and outputs a feature map for the latter, two-branch portion of the network, which is partitioned into a number of "stages". One branch then predicts confidence maps for each joint while the other simulataneously predicts PAFs for each joint. After the first stage, the initial predictions for the confidence maps and PAFs, as well as the original feature maps, are all concatenated together and input into the rest of the network. For the second stage and beyond, both branches utilize the cascading process of refining its predictions through latter stages, or networks (the model used for the results in their paper has six such stages). Each branch computes a loss function for each stage of the form

$$\sum_i \sum_{\mathbf{p}} ||\mathbf{S}_i(\mathbf{p}) - \mathbf{S}_i^*(\mathbf{p})||_2^2 \tag{1}$$

$$\sum_j \sum_{\mathbf{p}} ||\mathbf{L}_j(\mathbf{p}) - \mathbf{L}_j^*(\mathbf{p})||_2^2 \tag{2}$$

where $\mathbf{p}$ is a point on the image, $\mathbf{S}$ and $\mathbf{L}$ are the confidence maps and part affinity fields, respectively, and a * denotes the ground truth values. Restated, an L2 loss is computed for every confidence map/part affinity field for every joint/limb for every point on the image, and those losses are summed together. The final loss function just sums the total loss for each stage, i.e.,

$$\sum_{t=1}^{T} (f_{\mathbf{S}}^t + f_{\mathbf{L}}^t)$$

where at stage t, $f_{\mathbf{S}}^t$ and $f_{\mathbf{L}}^t$ are losses of the form (1) and (2), respectively.

3.3. Shrinking the Network

The obvious concern with Cao *et al*.'s network for our application is its runtime and memory usage. While Cao *et al*.'s network runs more or less instantaneously with 2 GPUs, our goal is to decrease the runtime and memory usage of the network to run more efficiently on a mobile device. To do so, we propose a number of different ways to shrink the network while still achieving similar accuracy:

1. *Decrease the number of stages in the latter part of the network.*: The two branch section of the network (where the network simultaneously predicts confidence map and Part Affinity Fields) refines its predictions over 6 identical stages. We can remove a number of these stages and measure the runtime improvement.

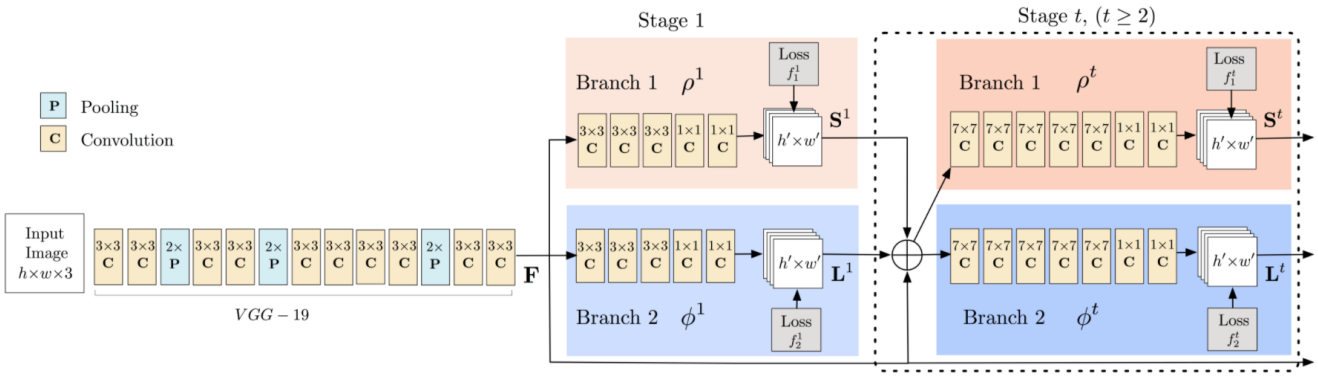

Figure 2. The entirety of Cao *et al.*'s convolutional network. The first portion, initialized by 10 layers of VGG-19, outputs a feature map to the 2-branch segment of the network. One branch predicts confidence maps of keypoints while the other predicts part affinity fields for two connected joints. These predictions are refined over successive stages before outputting the final prediction.

2. The network is initialized with the VGG-19 [16], so we can try initialization with smaller networks such as AlexNet [10] in order to reduce time and memory usage.

4. Dataset

The models are trained and tested on Microsoft's COCO dataset [11], which is a publicly available dataset for image recognition, segmentation, and captioning. It is widely used in the field of pose estimation because it has keypoints for 100,000 people, which are used as ground truth labels for detecting body parts.

Figure 3. Example of images from the COCO dataset. Many of the images involving people have associated keypoints for joints that can be used in pose estimation tasks.

5. Experiments and Results

All of our experiments utilize the open source code [1] from Cao *et al.* that they posted in conjunction with the release of their paper. The repository includes code for downloading the Microsoft COCO dataset as well as programmatically writing the .prototxt files required to train and test the models using the Caffe framework [9].

5.1. Note

An important note is that we took advantage of Caffe's naming protocol in order to directly load trained weights from Cao *et al.*'s original model into our smaller model. On their repository, Cao *et al.* comment that training their final model took around 5 days using 2 GPUs. With limited time and computational resources, we opted for loading weights in order to obtain more useful results. Therefore, these initial experiments with stage reduction did not involve any of our own training. This clearly leaves room for even better results from retraining these reduced models, and we will certainly address this concern in future work.

5.2. Analyzing Runtime

We analyze the runtime of our abridged models by creating a test set of 100 randomly sampled images from the COCO test set, computing the time it takes each model to run a single forward pass on each image, and averaging over the number of samples. The results of this experiment are shown in Figure 4. It is clear that the decrease in computational time becomes more apparent on increased scale factors for the image, although in general the runtimes for each model increase at roughly the same rate. The total time to compute predictions for a given image is just the sum across these scale factors, as the confidence maps and part affinity

[1] https://github.com/ZheC/Realtime_Multi-Person_Pose_Estimation

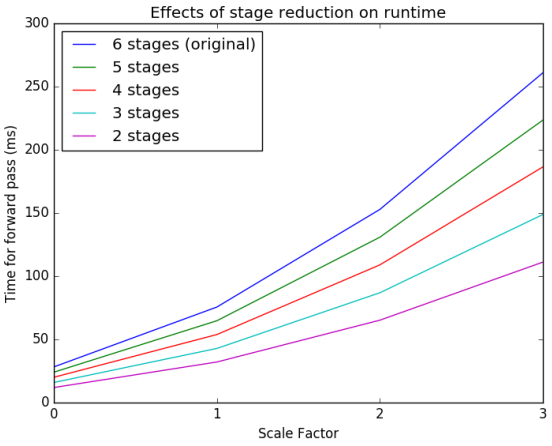

Figure 4. Results from decreasing the number of stages in the two-branch part of the network (the original network had 6 stages). Each net ran forward passes on 100 random images for increasing scales and the times were averaged out.

fields are computed for each scale and then averaged to give the final predictions.

These initial results are promising: with half the number of stages, the 3 stage network runs nearly twice as fast as the original 6 stage model. While increasing the speed on the order of milliseconds for a single image is hardly groundbreaking, the benefit becomes more plausible when considering a video file with many frames per second. For example, the iphone 6 and 6 plus can record videos at 60 frames per second. For a 5 second video, this would leave 300 frames to be processed. For an entire image, the 6 stage model takes an average of 500 milliseconds to compute its predictions, while the 3 stage model takes an average of 300 milliseconds. This means the 3 stage model could compute a pass over the video file a whole minute faster than the original 6 stage model.

5.3. Analyzing Accuracy

The reduction in runtime is encouraging, but it obviously means very little if the network's overall accuracy is highly compromised. To measure our model's accuracy, we use the original, 6 stage model's predictions as ground truth, making the assumption that we can only lose accuracy by reducing the number of stages. For each scale factor for each image, the computes confidence map and part affinity field predictions. These predictions are then averaged out over all the scale factors and compared with the ground truth average. We report the L2 losses incurred by the models in Table 1.

Raw numbers are not the most helpful in analyzing our models' performances, but one useful interpretation is comparing the loss to the size of the input images. For example, the 4 stage model incurs a confidence map loss of 6.68. Compared to the input size of 3 x 368 x 368, the fraction of the image that the model incorrectly computes is $\frac{6.68}{3x368x368} \approx 1e^{-5}$, a seemingly negligible margin. A point of emphasis for future work will be the extent to which we are willing to sacrifice accuracy in order to decrease the runtime and number of parameters for our model.

Stages	HeatMap Loss	PAF Loss
6	0.00	0.00
5	4.84	8.11
4	6.68	9.85
3	8.85	12.30
2	12.38	16.61

Table 1. Accuracy results from decreasing the number of stages in the two-branch portion of the network. Using the original model's predictions as ground truth, the loss for the predictions were computed for each scale factor and then averaged to give the final loss.

5.4. AlexNet

Our main experiment with training a network involved replacing the VGG-19 layers in the first portion of the network with layers from a pre-trained AlexNet model given by Caffe's Model Zoo. To do so, we directly modified the original network's .prototxt file by replacing all of the layers associated with VGG-19 and replaced them with all of AlexNet's convolutional layers, making sure to adjust stride and padding to maintain the correct output shape.

HeatMap Loss	PAF Loss
28.74	41.45

Table 2. Accuracy results from replacing the VGG-19 layers of the original network with layers from AlexNet. The loss was computed identically to how we did for stage reduction.

This AlexNet model was drastically outperformed by the stage-reduced models, but it should be noted that there is much more hyperparameter tuning to be done. Future work should experiment with other models of this size (e.g., SqueezeNet, ZFNet).

6. Conclusion

Our results, while straightforward, give reason to believe that a convolutional neural network designed for estimating pose can be implanted into a system for computing kinematic information. We have shown that simple structural changes to the network can noticeably improve run time and the number of parameters without sacrificing too much accuracy. Our experiments also leave the door open for future work in shrinking the size of convolutional pose estimators

for use in estimating kinematics.. Obvious possibilities include retraining these smaller networks instead of reusing learned weights from a larger model, as well as building on the experiment with AlexNet via more dedicated hyperparameter tuning, preferably on a machine with more computational power. Other avenues for future research involve more sophisticated algorithms for distilling and pruning neural networks [8, 7].

References

[1] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[2] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1014–1021. IEEE, 2009.

[3] R. Berka. Inverse kinematics-basic methods.

[4] Z. Cao, T. Simon, S. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1611.08050, 2016.

[5] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[6] W. Gong, Y. Huang, J. Gonzalez, et al. Enhanced mixtures of part model for human pose estimation. *arXiv preprint arXiv:1501.05382*, 2015.

[7] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015.

[8] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[11] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[12] C.-P. Lu, G. D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):610–622, 2000.

[13] F. Lu et al. Robot pose estimation in unknown environments by matching 2d range scans. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 935–938. IEEE, 1994.

[14] T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1913–1921, 2015.

[15] V. Ramakrishna, D. Munoz, M. Hebert, J. A. Bagnell, and Y. Sheikh. Pose machines: Articulated pose estimation via inference machines. In *European Conference on Computer Vision*, pages 33–47. Springer, 2014.

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

[17] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. *CoRR*, abs/1312.4659, 2013.

[18] S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. *CoRR*, abs/1602.00134, 2016.

[19] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1385–1392. IEEE, 2011.

[20] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012.