CS230

Multi-person 2D Pose Estimation via CNN (Computer Vision)

Qingjun(Helen) Wang
qw77@stanford.edu

Michael Sylvester
mts@stanford.edu

Abstract

We re-implemented a box-free bottom up approach to efficiently detect 2D human pose in multi-person images. The proposed model should be able to maintain high accuracy while achieving real-time performance, irrespective of the number of people in the image. We propose to use convolutional neural network and Part Affinity Fields (PAFs), inspired by [1, 2]. Trained on COCO 2017 dataset, we expect our model to show high accuracy in performance and efficiency in both training and testing stages.

1 Introduction

Human pose estimation is an important topic and has enjoyed attention in computer vision communities for the past few decades. In various areas such as robotics, security and game animation, it is a crucial step to understand human pose in images and videos. For a multi-person pose detection scenario, the interaction of human activities and the relatively high runtime complexity of existing methods make this a challenging task. In this case, we propose to use bottom-up approach, which first predicts all keypoints for every person in the image in a fully convolutional way and then represents the association of keypoints via Part Affinity Fields (PAFs) [1], a set of 2D vector fields that encodes the location and orientation of limbs over the image domain. This algorithm can fully adopt the great prediction power of CNN and the speed of parallel computing of modern heterogeneous architecture.

2 Related work

The top-down approach [3, 4, 5] is to employ a person detector and perform single-person pose estimation for each detection. These approaches directly leverage existing techniques for single-person pose estimation [6]. Furthermore, the runtime of these top-down approaches is dependent to the number of people, because every person in the image requires the single-person detector to run. In such case, runtime performance will be highly impacted by the number of people in the image. In contrast, bottom-up approaches first identified all key points in the image. George Papandreou *et al* [8] propose a part-induced geometric embedding descriptor which allows us to associate semantic person pixels with their corresponding person instance, delivering instance-level person segmentations. Zhe Cao *et al* [1] present an approach using a nonparametric representation, which referred to as Part Affinity Fields (PAFs), a set of 2D vector fields that encode the location and orientation of limbs over the image domain. They demonstrate that simultaneously inferring these bottom-up representations of detection and association, encode global context sufficiently well to allow a greedy parse resulting in high-quality results, at a fraction of the computational cost. In particular, our work will be based on the algorithm Zhe *et al* presented and will further investigate the failure cases in the paper.

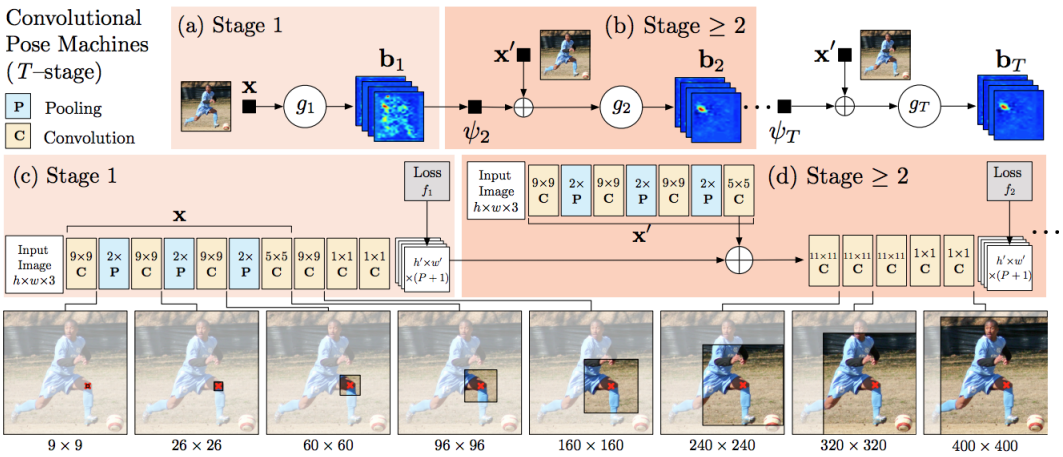

Figure 1: [1]Architecture of the two-branch multi-stage CNN. Each stage in the first branch predicts confidence maps St, and each stage in the second branch predicts PAFs Lt. After each stage, the predictions from the two branches, along with the image features, are concatenated for next stage.

3 Dataset and Features

We trained and evaluated our method on COCO 2017 [3] which has 118K training images and 5K validation images along with human pose key point annotations. These datasets landmark the key points for human pose (nose, neck, left/right eye, left/right ear, left/right shoulder, left/right elbow, left/right wrist, left/right hip, left/right knee and left/right ankle) and collects images in diverse scenarios that contain many real-world challenges as well as the failure cases Cao *et al* presented in the paper [1].
We use the full COCO 118K/5K train/dev dataset. For data preprocessing, to achieve certain precision, we normalize input cropped images to size 368 x 368. We also used a pre-trained VGG-19 network and dataloader module from [1] as a baseline of our work.

4 Methods

Our model follows the guideline of the method proposed in [1], with hyperparameter and learning stages tuned. We also used a pre-trained VGG19 network on ImageNet.

4.1 Pose Machines

A pose machine (see Figure 1a and 1b) consists of a sequence of multi-class predictors, gt(·), that are trained to predict the location of each part in each level of the hierarchy. In each stage t 1 . . . T, the classifier gt predict beliefs for assigning a location to each part Yp = z, based on features extracted from the image at the location z and contextual information from the preceding classifier in the neighborhood around each Yp in stage t.
In subsequent stages, the classifier predicts a belief for assigning a location to each part Yp = z, based on (1) features of the image data x again, and (2) contextual information from the preceeding classifier in the neighborhood around each Yp:

$$g_t\left(\mathbf{x}'_z, \psi_t(z, \mathbf{b}_{t-1})\right) \rightarrow \{b_t^p(Y_p = z)\}_{p \in \{0 \ldots P+1\}}$$

In each stage, the computed beliefs provide an increasingly refined estimate for the location of each part.

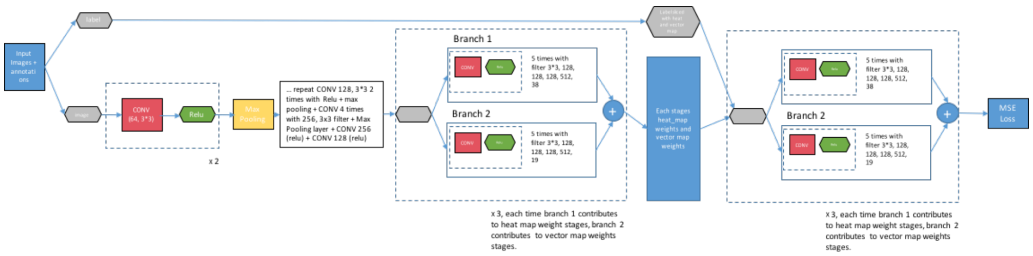

Figure 2: Architecture of our implemented two-branch multi-stage CNN, inspired by [1].

4.2 Convolutional Pose Machines

The first stage of a convolutional pose machine predicts part beliefs from only local image evidence. Figure 1c shows the network structure used for part detection from local image evidence using a deep convolutional network. The evidence is local because the receptive field of the first stage of the network is constrained to a small patch around the output pixel location.
A predictor in subsequent stages (gt>1) can use the spatial context (t>1(·)) of the noisy belief maps in a region around the image location z and improve its predictions by leveraging the fact that parts occur in consistent geometric configurations. In the second stage of a pose machine, the classifier g2 accepts as input the image features x^2 and features computed on the beliefs via the feature function for each of the parts in the previous stage. The feature function serves to encode the landscape of the belief maps from the previous stage in a spatial region around the location z of the different parts. For a convolutional pose machine, [1] do not have an explicit function that computes context features. Instead, [1] defined as being the receptive field of the predictor on the beliefs from the previous stage.

4.3 Learning in CPM Loss function

The sequential prediction framework of the pose machine provides a natural approach to training this deep architecture that can address gradient vanishing issue [1, 6, 7]. Each stage of the pose machine is trained to repeatedly produce the belief maps for the locations of each of the parts, encourage the network to repeatedly arrive at such a representation by defining a loss function at the output of each stage t that minimizes the l2 distance between the predicted and ideal belief maps for each part. The ideal belief map for a part p is written as bp (Yp = z), which are created by putting Gaussian peaks at ground truth locations of each body part p. The cost function we aim to minimize at the output of each stage at each level is therefore given by:

$$f_t = \sum_{p=1}^{P+1} \sum_{z \in \mathcal{Z}} \|b_t^p(z) - b_*^p(z)\|_2^2.$$

The overall objective for the full architecture is obtained by adding the losses at each stage and is given by:

$$\mathcal{F} = \sum_{t=1}^{T} f_t.$$

We use standard stochastic gradient descend to jointly train all the T stages in the network. To share the image feature x across all subsequent stages, we share the weights of corresponding convolutional layers (see Figure 2) across stages t >= 2.

5 Experiments

5.1 Hyperparameter Tuning

For each stage of the model, MSE loss is used for CPM loss. Regarding hyperparameters, we trained 50 epochs, with 2.8K iterations each. We use batch size 9, which is the maximum number to fit our

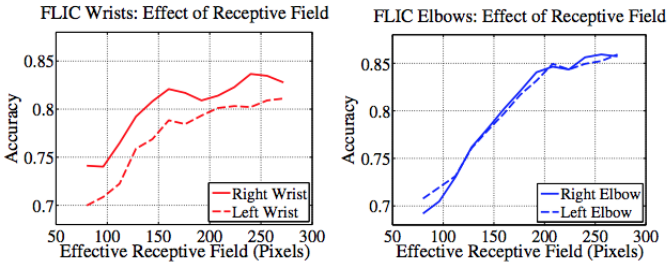

Figure 3: [1] Large receptive fields for spatial context. In our case we use 368 x 368 to get best accuracy

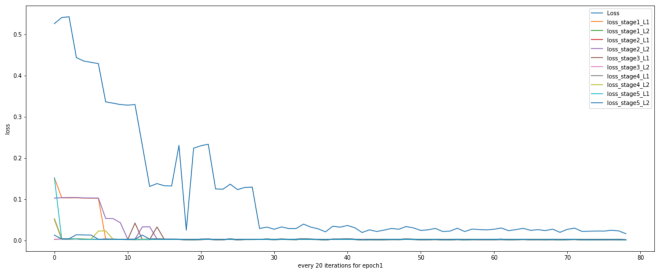

Figure 4: Loss values for different stages over iterations through training.

GPU memory - on both local machine and remote EC2 server, we have tried several batch size from 72, 36, 18 to 9. The learning rate is 0.1 with Adam optimizer.
And since we used the sequential prediction with learned spatial context features, and network is locally supervised after each stage using an intermediate loss layer that prevents vanishing gradients during training , the weight decay we used in this training is 0.
As proposed by [1] Figure 3 that accuracy improves with the size of the receptive field and will saturate around normalized size, we normalize cropped images into a size of 368 x 368 pixels for better precision.

5.2 Model Evaluation

Calculated mAP on COCO val2017 dataset is 0.63.
Figure 4 shows the loss value on different stages through the training integration also validating the model is training with loss decent through time. Figure 5 shows the picture evaluations on different epochs. As we can see, the pose estimator improved the performance with model training ran to more iterations.
We also show a live demo in the submitted video for a real-time pose detection from web camera, which can also show the model is able to maintain high accuracy while achieving real-time performance, irrespective of the number of people in the image.

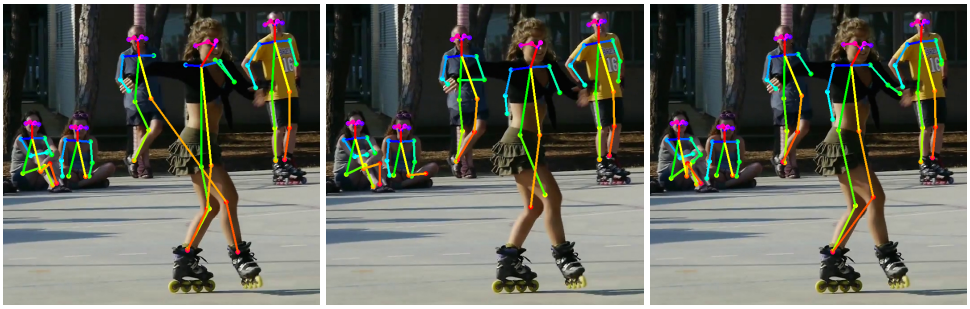

Figure 5: Pose estimation on same image using model trained by 16 epochs, 34 epochs and 50 epochs.

6 Conclusion

We successfully re-implement the real-time multi-person pose estimator through pytorch, which the model architecture proposed by [1]. During in the implementation, we encountered and tackled multiple technical difficulties including hitting computational resource limits and setting up webcam environment using openCV. By tuning the hyperparameters and changing effective receptive field, we see performance and accuracy improved.
Code is uploaded to github: https://github.com/QJWang23/PoseEstimation.git

7 Contributions

Qingjun(Helen) Wang contributes to the implementation, tuning of the model and write up for the report. Michael Sylvester contributes to the training cycle of the model, testing and evaluating the results.
Also we would like to thank our TA Advay Pal for the feedback during the office hours and many thanks to all the teaching stuff for CS230 delivering this great course.

References

[1] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[2] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[4] Jay A Alexander and Michael C Mozer. Template-based algorithms for connectionist rule extraction. In *Advances in neural information processing systems*, pages 609–616, 1995.

[5] Michael E Hasselmo, Eric Schnell, and Edi Barkai. Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region ca3. *Journal of Neuroscience*, 15(7):5249–5262, 1995.

[6] Xianjie Chen and Alan L Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *Advances in neural information processing systems*, pages 1736–1744, 2014.

[7] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer, 2016.