Deep Learning for Human Pose Estimation

Ashwin Gupta
University of Michigan
Ann Arbor, MI
ashmg@umich.edu

Spencer Gritton
University of Michigan
Ann Arbor, MI
sgritton@umich.edu

Abstract

Pose estimation is a computer vision task that includes detecting, connecting, and tracking human joint positions in input images, videos, and other vision-based signals. In this work we implement the first stage of DeepPose [5], a seminal work in applying deep convolutional networks to pose estimation in images. During the course of this implementation we validate the DeepPose network on a novel dataset, Microsoft Common Objects in Context (COCO) [3] and propose a variety of improvements to the original network.

1. Introduction

In our work we attempt to apply deep learning to the problem of human pose estimation. Pose estimation involves estimating the positions of keypoints on the human body. These keypoints typically include joint positions like the shoulders, ankles, knees, head, and other areas.

There are multiple practical applications for pose estimation including safety monitoring, interaction in VR/AR environments, sports analysis, and surveillance for law enforcement. We expect reliable monocular pose estimation on RGB images to become an increasingly important capability as VR/AR become more common in industrial and commercial applications.

In this work we choose to implement and evaluate the pose estimation network proposed in DeepPose [5]. As part of this implementation we validate the network on a novel dataset, COCO [3], a modern and ubiquitous dataset in the computer vision space. This network takes as input cropped photos containing a single subject, which in practice would be provided by a human object detector, and outputs 17 pairs of (x, y) coordinates of keypoints found on the subject within the image. These keypoints could point to a number of important features including: the knees, elbows, ears, and more.

2. Related Work

There have been a variety of methods proposed for the task of pose estimation. Here we cover only the more modern deep learning architectures as they have thus far provided superior results. The majority of these methods fall into one of two categories.

Top-down methods require first detecting people in images with standard object detectors such as YOLO. These detections are subsequently cropped and taken as input to a pose estimation network that runs on a per person basis. This type of approach requires reliable object detection capabilities. DeepPose [5] was an early pioneer of this method. Some top-down state-of-the-art methods exist such as AlphaPose [2].

Bottom-up methods, first introduced by DeepCut [4] and since improved upon in OpenPose [1], involve the detection of keypoints directly on the input image and subsequent grouping of those keypoints to separate subjects in the image. These approaches tend to run more efficiently for multi-person scenes as only one pass through the estimation network is required. Unfortunately though, bottom-up methods are more prone to errors attributing joints to the wrong subject. As such, a majority of the latest research is focused on improving these methods, as the optimal bottom-up method would provide both speed and accuracy.

In our work we focus on the highly influential top-down approach from DeepPose [5] which we deemed to be more feasible to implement than a novel bottom-up method given our time and resource constraints. However, we drew inspiration from AlphaPose [2] in our choice of the COCO dataset.

We would like to acknowledge that the original DeepPose paper refines keypoint predictions with subsequent stages of the same model trained with different weights. This refinement provides marginal accuracy improvements, but requires an exponential increase in training time — due to computational restrictions we opted to forgo this portion of the implementation.

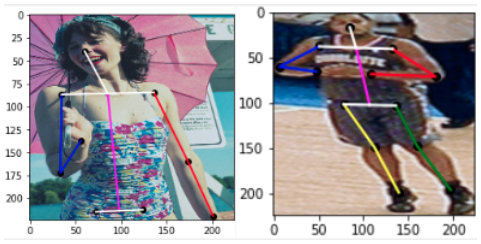

Figure 1. Left: Not all keypoints are visible. Right: All keypoints are visible. Our initial network implementation requires the latter.

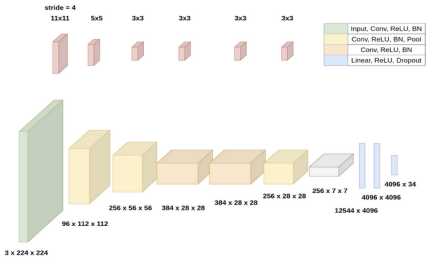

Figure 2. Architecture diagram of DeepPose network. Showcases convolutional feature extractor followed by fully-connected keypoint regression layers.

3. Methods

3.1. Dataset

The COCO dataset is a standard for bench-marking, training, and evaluating state-of-the-art computer vision models in: segmentation, object recognition, captioning, and pose estimation. Due this popularity and a relative ease of obtaining the data as well as over 200,000 keypoint annotated images, it was natural to select it for the implementation and testing of the DeepPose network.

3.2. Data Processing

Keypoint data inside COCO is made up of image-keypoint label pairs where the labels vector contains 17 (x, y, v) pairs. These pairs denote the (x, y) coordinates of each joint within the image and v contains a visibility flag of: 0 (not in image), 1 (occluded), or 2 (visible). To reduce training time and simplify the network learning process, we opted to only train and validate on a subset of COCO where all 17 keypoints were visible. This allowed us to avoid dealing with occluded or invisible keypoints, although we do address this later. Thus the pre-processing involved the following steps:

1. Find all keypoint annotated images in COCO
2. For each image determine if all 17 keypoints exist in the image with a visibility label of: 2 (visible)
3. Crop the image content to the bounding box of the person in the image
4. Scale the image to 224x224 and move the keypoints accordingly
5. Normalize keypoints between -0.5 and 0.5

Keypoints were transformed into a normalized coordinate system where the center of the cropped input image was considered the origin, and the image was considered to have half-unit length in each spatial dimension. The normalized (x, y) coordinates for all 17 keypoints were then concatenated into a single high dimensional vector for each image. A labeled keypoint vector z of the following form:

$$\vec{z} \in \mathbb{R}^{34}, \quad \vec{z} = \begin{bmatrix} x_1 \\ y_1 \\ \dots \\ x_{17} \\ y_{17} \end{bmatrix} \tag{1}$$

$$(x_i, y_i) \in \{(a, b) \in \mathbb{R}^2 | -0.5 < a, b < 0.5\} \tag{2}$$

This data processing step yielded a total of 4,037 training images and 159 validation images.

3.3. Architecture

The architecture follows a familiar CNN paradigm for other common perception tasks. A convolutional backbone is used to learn a semantic feature map. Subsequent fully connected layers learn to regress keypoint (x, y) positions based on this feature map. The input to the network is an image batch tensor $x \in \mathbb{R}^{B \times 3 \times 224 \times 224}$ where B is the batch dimension and images are in standard 3-channel RGB colorspace. The network outputs predictions $\hat{y} \in \mathbb{R}^{34}$ to match the processed ground truth label format.

We utilize stride 1 convolutions with the amount of padding necessary to maintain the input size at each layer. The exception to this is the first layer, which uses a stride 4 convolution. After each convolution there is a ReLU non-linearity and a batch normalization (BN) layer. While BN was not included in the original DeepPose architecture, we suspected it would improve training by reducing internal covariate shift and so we chose to add it. A MaxPool operation is included after some layers to reduce resolution and increase the receptive field. Finally, the fully connected regressor is made up of 2 hidden layers where both are followed by a ReLU non-linearity and dropout with a probability of 60%. A complete diagram of the network with hidden layers and filter sizes can be found in Figure 2.

4. Experiments

4.1. Training

The network was trained for 2,700 epochs using a combination of a cloud-based RTX 5000 and a local RTX 2060 Super. We applied a learning rate of 0.001 and batch size of 64 images. The loss function was a standard L2 loss between the predicted keypoint vector $\hat{y}$ and the label keypoint vector y.

$$L(\theta) = \sum_{i}^{B} ||\hat{y}_i - y_i||^2 \tag{3}$$

Inside the training loop we utilized stochastic gradient descent and continued training until validation loss was unable to make further improvements.

4.2. Validation

For validation we use the percentage of correct parts (PCP) metric. This metric measures detection accuracy per limb. PCP performs a binary classification on a particular keypoint detection as correct or incorrect based on the limb it belongs to. If the Euclidean distance between the two predicted keypoints making up the limb and the ground truth keypoints is at most half the length of the corresponding limb then that prediction is considered correct.

4.3. Quantitative Results

Table 1 summarizes the overall performance of our network across all limbs on the validation split of our processed dataset. Note that this comparison is not entirely one-to-one as we use COCO while DeepPose uses the Leeds Sports Pose Dataset [5], but it is indicative of the relative performance of our network. The results of stage-three of DeepPose are included for completeness, but we did not implement the multi-stage approach on our network.

Ours	DeepPose St.1	DeepPose St.3
31%	54%	61%

Table 1. Average PCP for all arms and legs.

While our network is clearly less accurate than DeepPose, it does relatively well considering the limited time and computing resources available to us. The original stage-one DeepPose network was trained distributively for 72 hours via 100 GPUs with 11,000 train images [5]. Beyond this, each successive stage of DeepPose was trained for 168 hours on roughly 440,000 samples [5]. In comparison, our network was trained for roughly 12 hours on one GPU with 4,037 images. With more data and training time, we would likely see a similar PCP. However, the DeepPose architecture is clearly somewhat applicable to pose estimation tasks where resources are constrained.

4.4. Qualitative Results

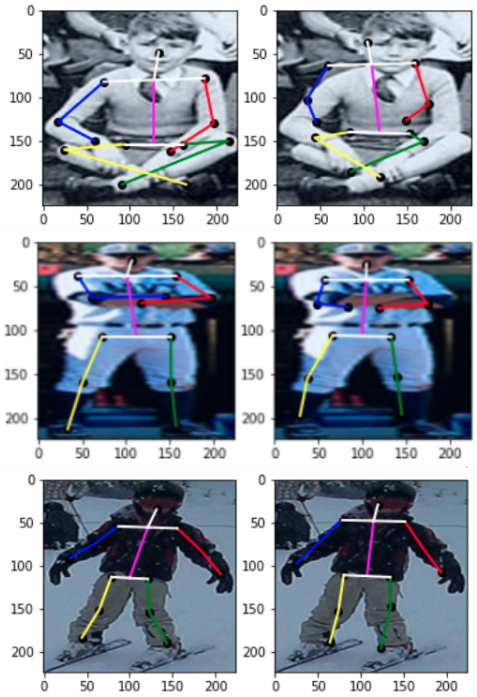

Figure 3. Successful detections. Left: Ground truth. Right: Predictions.

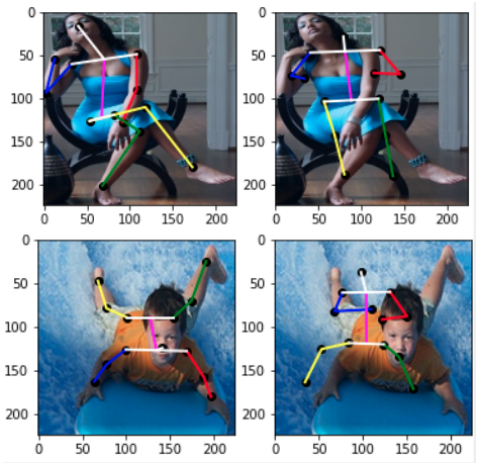

Figure 4. Unsuccessful detections. Left: Ground truth. Right: Predictions.

It is clear from Figure 3 that our network was able to develop a reasonably complex semantic understanding for the pose estimation task. This is especially evident from its ability to learn various pose archetypes beyond standing figures with arms to the side such as the baseball player with crossed arms or the boy sitting down. However, the failure cases shown in Figure 4 demonstrate how the network

struggled to detect poses in images that were severely out-of-distribution. A vast majority of the images in the training set with all 17 keypoints contained images of subjects standing. Thus, in validation images where people were sitting or prone, the network was less likely to demonstrate good semantic understanding. Additional data would likely remedy this problem.

Furthermore, data augmentations like flips, rotations, and random resizing would help improve detection capabilities at multiple scales and orientations. Adding these augmentations in addition to a larger amount of data and increased training time would likely push the results look much closer to ground truth.

4.5. Improvements

4.5.1 Augmentations

Once overfitting became clear on our relatively small dataset we found the next natural course of action to be applying augmentations. With this in mind we applied a variety of image-based augmentations including: grayscale shifting, random color jitters, solarization, posterization, auto-sharpness adjustments, and random equalizations. Despite the number of augmentations applied we found that these added very little in terms of quantitative performance. Due to this, we believe that flips, rotations, and additional crops should be explored as they force the model to learn from an input that has undergone much more significant changes.

4.5.2 Non-Visible Keypoints

Due to the nature of photographs, it is likely that many images will not contain all possible keypoints. To address this issue we attempted to train a network where the output for each keypoint consists of an (x, y, v) pair where (x, y) are normal coordinates and v indicates if the point is in the image *1* or not *0*. Thus, if a photo does not include a left knee, an optimal network would output a *0* as v for that joint. To accommodate this we made the following changes:

1. Allowed images into the dataset where keypoints were missing

2. Switched the output of the model from 34 to 51 regressed outputs (adding the 17 additional v points)

3. Implemented a new L2 loss function, given in Equations 4 and 5, that ignores (x, y) regressions if the ground truth point is not in the image.

$$L(\theta) = \sum_{i}^{B} w * ||\hat{y}_i - y_i||^2 \tag{4}$$

$$w = \begin{cases} [0,0,1] & y_i^v = 0 \\ [1,1,1] & \text{otherwise} \end{cases} \tag{5}$$

After implementation we found this method to be promising. Unfortunately, as a result of allowing more images into the dataset, we did not have adequate time to train this model and thus were unable to evaluate it properly.

4.5.3 ResNet-18 Convolutional Backbone

Another promising idea was to replace the DeepPose convolutional backbone with a pre-trained ResNet-18 model. We felt this would provide the regressor a more semantically rich feature map, without the expense of training. However, this proved to be a difficult substitution. As shown Figure 5, the network learned to always predict the mode distribution of keypoints over the training set. We feel the most likely reason for this would be ResNet's pre-trained convolutional features encoding unnecessary ImageNet class information instead of useful keypoint information. We propose that given adequate training time, it is likely a ResNet backbone trained from scratch would perform better than its DeepPose counterpart. This backbone would allow us to take full advantage of our addition of BatchNorm, which has been shown to enable training on much deeper networks.

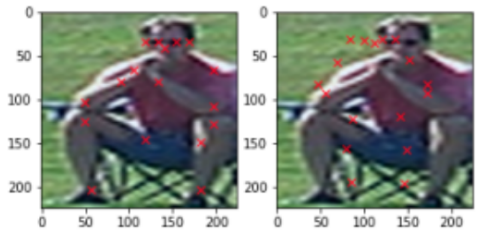

Figure 5. ResNet predictions tended toward the mode of the dataset. Left: Ground truth. Right: Predictions.

5. Conclusions

Based on our implementation, we find DeepPose to be a capable top-down pose detector on the COCO dataset. Despite our significant lack of training with respect to the initial DeepPose paper, we find that our implementation displays good semantic understanding for the most common types of poses in the training data. It is clear, however, that the our reduced training set was ultimately a limiting factor in producing results that rival those found in the original implementation. In addition to these findings, we proposed improvements to the DeepPose paper including: increased augmentations, utilizing a ResNet backbone, and handling non-visible keypoints. We believe that if given ample training time and resources, a network utilizing these changes could be both more accurate and useful than the original DeepPose implementation.

References

[1] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 1

[2] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. RMPE: Regional multi-person pose estimation. In *ICCV*, 2017. 1

[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. 1

[4] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V. Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1

[5] 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. 1, 3