Residual Stacked Hourglass Networks for Human Pose Estimation

Sun Jay Yoo, Kevin Peng, Manan Aggarwal

601.482 - Machine Learning: Deep Learning - Spring 2019
Instructors: Mathias Unberath, Jie Ying Wu, Cong Gao

1 Introduction

Human pose estimation refers to the problem of detecting human figures and estimating their key body joints in images and video. This is different from segmentation in the sense that it is not concerned with recognizing who or what is in the image. In recent years there has been a surge in research in this area owing to its wide range of uses ranging from augmented reality to animation to fitness. In this paper we study, evaluate, and compare three existing architectures for 2D single human pose estimation: DeepPose, Chained Prediction, and Stacked Hourglass. We then propose a new architecture called Residual Stacked Hourglass that combines the best features from each of these architectures in an attempt to outperform existing results. We combine the residual network model from DeepPose with the innovative hourglass modules from stacked hourglass to obtain competitive results that are just shy of the state of the art results without intense optimization and hyperparameter tuning.

2 Dataset

One of the most important elements when evaluating deep learning and machine learning models is the dataset. For the problem of human pose estimation, we decided to train and evaluate our models on Microsoft's COCO 2017 dataset.[1] COCO is a collection of images taken from everyday scenes. It contains 91 object categories (though only 80 are available with annotations) and a total of 1.5 million labeled instances in 328K images. The dataset is formatted in JSON that consists of the following sections - "info", "licenses", "images", "annotations", and "categories". The annotations are organized based on tasks into 5 types: object detection, keypoint detection, stuff segmentation, panoptic segmentation, and image captioning. For our purposes, we are only interested in the annotations for keypoint detection of people, which indicates the location of their joints. Using these annotations will allow us to find people within a scene and extract their pose based on the position of their joints.

2.1 COCO 2017 for Keypoint Detection

- "images": contains information about each image in the dataset. Each image is identified by a unique image id that may be different from its corresponding file name.
- "categories": contains only one category: "person". Each "person" has a "keypoints" and a "skeleton" section in addition to other sections. "keypoints" indicates different body parts and "skeleton" indicates connections between points.
- "annotations": contains a list of every individual object annotation from every image in the dataset. It also contains a list of keypoints within each segmentation instance.

2.2 Pre-Processing

Data augmentation was performed using a combination of translation, scaling and rotation. Translations were limited to 2 percent of the image width. Scaling was fixed between 70 and 130 percent of the original image size. Rotations were limited to a range of 40 degrees in either direction. Target images were created using the annotations and Gaussian blur was applied to generate a heatmap. All images were cropped to 256x256 pixels.

3 Related Works Evaluated

3.1 DeepPose

DeepPose was the first major paper to describe a meaningful use of deep learning for human pose estimation tasks.[2] Moving beyond deformable part models, DeepPose uses CNN to replace hand-crafted features and graphical models, yielding drastic improvements on standard benchmarks for state of the art performance.[3] This approach considers the problem in a holistic fashion, i.e., even if certain joints and features are visibly hidden, they can still be estimated by approximation. The original model uses a 7-layer AlexNet backend and an extra final layer that outputs 2D joint coordinates $(x_i, y_i) * 2$ for $i \in \{1, 2, ..., k)$ where k denotes the number of joints. This first part is trained using L_2 loss, instead of classification loss. In the implementation of DeepPose for comparative analysis, we use ResNet-34 instead of AlexNet for more comparable performance with modern deep architectures.[4] The original paper argues that the use of these generic architectures is motivated by their outstanding results in classification and localization with evidence to show state of the art performance. It has the holistic ability to estimate final joint locations from complex nonlinear transformations of the original image.

[1] https://arxiv.org/pdf/1405.0312

[2] https://arxiv.org/pdf/1312.4659.pdf
[3] https://www.cs.cmu.edu/~deva/papers/pose_pami.pdf
[4] https://arxiv.org/pdf/1512.03385.pdf

The key with this model, as seen in Figure 1, is that it then takes these predictions for further refinement using cascaded regressors. Images are cropped near the predicted joint and are directly fed to the next stage so that subsequent regressors see higher resolution inputs and learn features at a more exact scale. This results in higher precision in joint detection using direct regression. Generating heat maps as outputs has also been tested in later work, but was not used in the original architecture.[5]

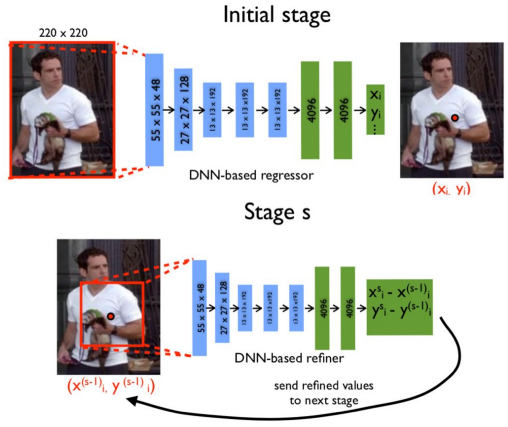

Figure 1: Overview of DeepPose architecture involving two stages. Top: schematic view of the DNN-based pose regression. Bottom: a cascaded regression applied on a sub-image to refine a prediction from the previous stage.

3.2 Chained Prediction

Chained predictions using CNNs present an adaptation of the sequence-to-sequence model for structured vision tasks, similar to RNNs.[6] Here, the outputs are predicted sequentially using a neural network and depends not only on its input but also on the previously predicted outputs. The model uses CNNs for extracting features from input images and a multi-scale "deception" (combination of deconvolution and inception) for making spatial predictions. Ultimately, we can avoid the hand-crafting of classical graphical approaches while utilizing the constraints of joint localizations across sequences. This chained model also achieves state of the art performance on human pose estimation for images and videos.

The chained model captures the dependence structure described above by maximizing the likelihood $P(Y \mid X)$ by consecutively applying the chain rule to model each output Y_t given the input X and all previous outputs $Y_{<t}$. More specifically, we let $Y = \{Y_{0,\ldots,} Y_{T-1}\}$ be the T objects to be detected such that Y_t is the location of the t-th joint or body part. $P(Y = y \mid X)$ is decomposed as:

$$P(Y = y \mid X) = P(Y_0 = y_0 \mid X) \prod_{t=1}^{T-1} P(Y_t = y_t \mid X, y_{0,\ldots,} y_{t-1})$$

In the paper, the likelihood $P(Y_t = y_t \mid X, y_{0,\ldots,} y_{t-1})$ is modeled with a CNN. For comparative analysis between models, we again use ResNet-34 as the CNN.

For the "deception" stage in our case of single images, the probability at each decomposition step of the previous likelihood equation is categorized by the hidden state h_t. This hidden state is computed with linear transformations of previous hidden states and output variables. A nonlinearity of ReLU is applied for each CNN stack. While this formulation is similar to RNNs in that the state of a previous step is transformed to the next, it differs in that the parameters involved in computing the hidden states and likelihoods are not necessarily tied across timesteps. According to the original paper, this design choice was appropriate for human pose estimation tasks where the number of outputs T, the number of joints or body parts, is fixed and where each step is different from the rest (i.e. single images vs. videos). As seen with Figure 2, this combination of CNN_x, feature extraction of the input image, and CNN_y, decoding the hidden state to heatmap over possible locations of a single body part, results in a "deception" stage that uses deconvolutional towers that are multi-scale in one layer, similar to the inception model with the difference that it is applied to deconvolutions.[7]

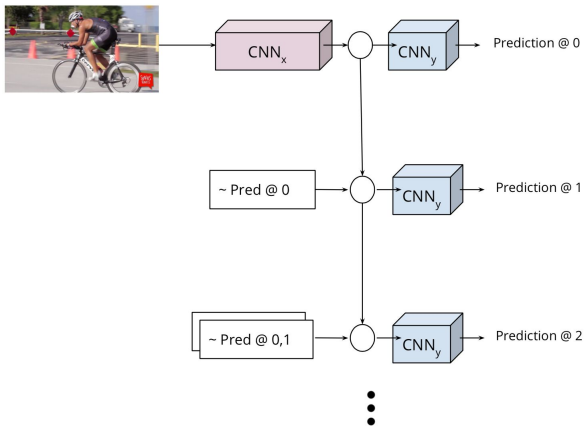

Figure 2: Visualization of the Chained Prediction model with the case of single images. Images are encoded with CNN_x and decoder CNN_y makes predictions using previous outputs and hidden states with a sequential model.

3.3 Stacked Hourglass

The stacked hourglass network for human pose estimation introduces a novel and intuitive network architecture that beats many previous state of the art human pose estimation models.[8] The "stacked hourglass" name is derived from the architecture's innovative technique of using pooling and upsampling layers which resemble an hourglass when stacked together. The design of such is inspired from the need to capture information at multiple scales of the image. Local information is essential for

[5] https://www.robots.ox.ac.uk/~vgg/rg/papers/tompson2014.pdf
[6] https://arxiv.org/pdf/1605.02346.pdf
[7] https://arxiv.org/pdf/1409.4842.pdf
[8] https://arxiv.org/pdf/1603.06937.pdf

detecting features like faces and hands, but global information is essential for final full-body pose estimates. The humans' orientation, arrangement of limbs, and the relationship of adjacent joints are among the many features that are best recognized at different scales. The network, in short, performs repeated bottom-up, top-down processing with intermediate supervision. Bottom-up processing transitions from high resolutions to low resolutions while top-down processing transitions from low resolutions to high resolutions. Skip connections are used to preserve spatial information at each of these stages and passes it through upsampling further down the hourglass. Intermediate supervision for each hourglass in the stack is also used to improve the final predictions with full spatial scale awareness.

The hourglass is set up by first using convolutional and max pooling layers to process features down to a very low resolution, At each max pooling step, the network branches off to apply more convolutions to the original pre-pooled resolution. After reaching the lowest resolution, the network starts the "top-down" sequence of upsampling and combination of features across these scales. Following the work that was originally used to generate heatmaps from DeepPose, the network performs a nearest neighbor upsampling of the lower resolution with a element-wise addition from the re-routed branch to bring together information across two adjacent resolutions.[9] The topology of each hourglass is symmetric, i.e., each bottom-up layer has a corresponding top-down layer. The output of the network is a set of heatmaps where for a given heatmap, the network predicts the probability of a human joints' presence at each pixel.

Some implementations utilize "inception-based" designs, but further experimentation did little to boost performance or training time.[10] These hourglasses are also stacked sequentially with intermediate supervision through a loss applied at the residual module when the network splits to produce a set of heatmaps. The theory here is that subsequent hourglass modules allow the network to process high level features to be re-processed with further evaluation and re-assessed with higher order spatial relationships. If supervision is only provided at the end of these stacks, the network cannot re-evaluate these features in a even larger global context. This combination of high-level context between hourglasses and low-level context within each hourglass produces the basis for this architecture's success. An overview of this architecture is provided in Figure 3.

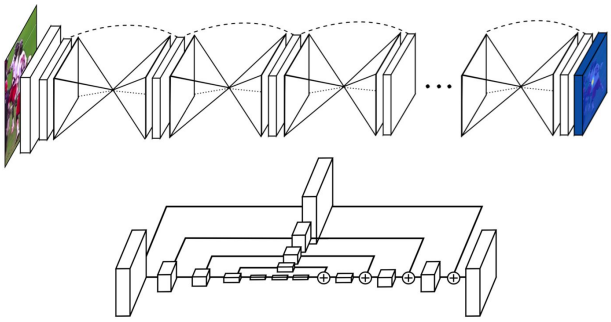

Figure 3: Top: overview of the Stacked Hourglass model with multiple hourglass modules. Bottom: single hourglass module with topological symmetry between routed highways.

4 Methods

We present implementations of the three aforementioned architectures, DeepPose, Chained Prediction, and Stacked Hourglass as well as our own architecture, the Residual Stacked Hourglass. The implementation we present uses the open-source PyTorch framework by Jain and Shah that provides excellent training and test codes with the PCK performance metric.[11] We also use the COCO dataloader provided by Microsoft's implementation of Simple Baselines for Human Pose Estimation and Tracking and we also use Cython integration for Pytorch from Faster R-CNN.[12] [13] For our compute, we use a Chrome Remote Desktop with a virtual instance hosted on the Google Cloud Platform for training and optimization with a NVIDIA Tesla P100.

4.1 Residual Stacked Hourglass

After exploring these three architectures we take the best features from each to create our own architecture: the residual stacked hourglass. Here we combine the residual network model from DeepPose with the innovative hourglass modules from Stacked Hourglass. For DeepPose, we see that the simple and elegant use of ResNet-34 produces excellent results due to the combination of deep filters and functional residual blocks.

First, we hypothesized that increasing the number of hourglass modules from two to five would help improve performance much like VGG's improvement over AlexNet with the ability to capture more nonlinearities and holistic approximations of human joints. Since each hourglass itself is supervised to capture both local and global information scales and stacking allows for even broader representations to be reprocessed for spatial context, adding more stacks would potentially allow us to improve performance by reprocessing these scales on an even larger scale. With DeepPose's high performance using simple deep architectures (ResNet-34), we think increasing depth would improve performance overall.

[9] https://www.robots.ox.ac.uk/~vgg/rg/papers/tompson2014.pdf
[10] https://arxiv.org/pdf/1512.03385.pdf

[11] https://github.com/Naman-ntc/Pytorch-Human-Pose-Estimation
[12] https://github.com/Microsoft/human-pose-estimation.pytorch
[13] https://github.com/rbgirshick/py-faster-rcnn

Furthermore, much like with DeepPose's use of ResNet-34, we believe our deeper stacked architecture would be easier to optimize with the use of "gradient highways" (i.e. residual blocks in ResNet) between hourglass scales. We also found the use of topologically symmetric highways within each hourglass from the Stacked Hourglass architecture to be innovative in their combination of lower-processed features with higher-process features at each level. While we do not change the intermediary output sizes between each hourglass stack, we do implement a similar structure on a grander scale by adding the output of the first stack to the output of the fourth stack and the output of the second stack to the output of the third stack with these gradient highways. With these gradients flowing through multiple levels of the architecture at multiple stages, we believe this would make optimization and training easier.

Given a sample 3-channel 256x256 input image, our network first reduces this to a workable size of 64x64 with a series of convolutional layers, batch norm, and ReLU. Then, we apply five hourglass stacks, each of which apply convolutional and max pooling layers to reduce the size to 4x4 with corresponding upsampling to restore the size to 64x64. As described with the Stacked Hourglass section, symmetric connections are made between each corresponding layer within each hourglass. Gradient highways are established between hourglass stacks, again with symmetry, and the final output remains as heatmaps. The overall structure of the adapted Stacked Hourglass model as the Residual Stacked Hourglass architecture can be seen in Figure 4.

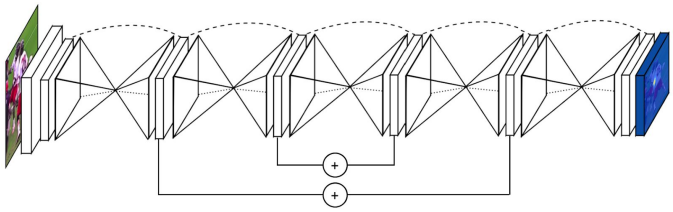

Figure 4: Schematic (adapted from the original Stack Hourglass paper) of the deep five-stack Residual Stacked Hourglass architecture with symmetric "gradient highways". Each hourglass remains the same as presented in Figure 3.

5 Results

5.1 Performance Metrics

For all models, optimization was performed on the training set using L_2 loss and RMSProp, same as those used by Thompson et al. and the original Stacked Hourglass implementation.[14] Likewise, the same learning rate of 2.5e-4 was used for all models. The DeepPose model resulted in direct regression, while all other models resulted in heatmaps as seen with Figure 5. The performance measure used to evaluate these models in comparison is the standard Percentage of Correct Keypoints (PCK) used in human pose estimation tasks which reports the percentage of detections that fall within a normalized distance of the ground truth.[15] We note that there may be extreme cases where a joint may be severely occluded and may not have any annotation at all. This metric used when evaluating these models do not exactly reflect how these situations are recognized by these models, but it does provide an excellent approximation of human pose estimation accuracy.

Figure 5: Example heatmap outputs produced by the network. The left image is the final pose estimate using the maximum activations across the sample heatmaps for each joint.

5.2 Model Comparison

Model	**Parameters**	**PCK**
DeepPose	40 M	70.4
Chained Prediction	26.5 M	82.0
Stacked Hourglass	12.6 M	84.7
Residual Stacked Hourglass	***31.1 M***	***81.1***

Table 1: PCK performance on the COCO dataset and the total number of trainable parameters (given 3-channel 256x256) for the four architectures compared.

As seen with Table 1, the Stacked Hourglass had the best PCK performance on the COCO dataset as it best uses its architecture for multi-scale feature extraction that creates best results for pose estimation. DeepPose simply uses a classification architecture as its backbone and does not use spatial scale to the fullest extent. Chained Prediction seemed to be more beneficial for videos leveraging hidden states across timeframes and would have diminishing results across single images. Our analysis revealed that although DeepPose refined localization of joints using a series of cascading regressors, the precision gained from repeating regression on a smaller segment of the image failed to exceed the precision of other methods. We believe that in Chained Prediction and Stacked Hourglass, the coupling of cascading regressions (and upsampling), each with their own learned parameters resulted in a more versatile model. Thus, our implementations explored using a similar framework that would combine the multi-scale connections within hourglass modules with the multi-stage connections of higher order processes as seen with DensePose and Chained Prediction to produce comparable results without intensive hyperparameter tuning and optimization.

[14] https://www.robots.ox.ac.uk/~vgg/rg/papers/tompson2014.pdf

[15] https://www.cs.cmu.edu/~deva/papers/pose_pami.pdf