Recurrent Human Pose Estimation

CS231A Project

Matthew Chen
Department of Computer Science
Stanford University
mcc17@stanford.edu

Melvin Low
Department of Computer Science
Stanford University
mwlow@stanford.edu

1. Introduction

Human pose estimation is the task of estimating the joint locations of one or multiple people within an image. It is a core challenge in computer vision because it forms the foundation of more complex tasks such as activity recognition and motion planning. For example, joint locations have been used to supplement other visual features to determine the trajectory of a person through a sequence of video frames.

In the past, the leading techniques for pose estimation were the deformable parts model and its variants. These approaches used convolutions of fast part detectors over the input image to generate a coherent joint configuration [3]. More recently, deep learning techniques such as convolutional neural networks as well as optimization methods such as mixed integer programs have achieved state of the art results, for detecting both single and multiple people.

From a probabilistic perspective, pose estimation is the task of finding a joint configuration of n joints, $y_1, y_2, ..., y_n$, given a source image x, i.e. $\arg\max_{y_1,...,y_n} p(y_1, ...y_n|x)$. Several state of the art methods have attempted to calculate this directly with a convolutional neural net. While this has worked very well, the natural dependencies between human joints suggest that explicitly modeling these dependencies may give a boost in performance. This corresponds to calculating

$$\arg\max_{y_i} p(y_i|y_{i-1}, y_{i-2}, ..., y_1, x), i = 1, 2, ..., n$$

—generating the joints one at a time in sequence. This paper investigates this approach. We use a recurent network architecture, which is particularly well suited for capturing sequence information. Preliminary results show that the recurrent approach unfortunately performs much worse than the straightforward convolutional one. It seems to enforce joint relationships too strongly and thus tends toward the most common pose in the dataset.

2. Previous Work

Prior to the widespread use of neural networks in computer vision, classic approaches were applied to the problem of pose estimation. These approaches generally took the form of expressing the body as a graphical model and encoding priors on the likely configurations of the parts [4]. A joint objective would then be optimized which accounted for both the match score from a part detector and the likelihood of the resulting configuration. These models, however, suffer in terms of computational tractability as they involve inference over a graphical model. Many of the techniques deployed to improve the tractability come with costs to the expressiveness of such models[16].

DeepPose was one of the first papers to deploy deep networks for this problem. The authors formulated the problem as a regression task. In their model the joint coordinates were predicted directly from a regression layer of a convolutional neural network (CNN). Due to the need to downsize the original image to fit the fixed CNN input size of 224 by 224 they used a cascade of CNN classifiers [13], each cropping the picture centred around the previous prediction at a higher resolution, to further refine their pose estimates. One issue with this approach is that the errors from previous mistakes can propagate into future decisions as each choice involves a crop of the image and thus limits the possible domain of solutions.

The authors in [2] similarly use a CNN to predict joint coordinates but add a iterative feedback mechanism where an initial pose is refined at every timestep given the image and the previous estimate. They accomplish this by defining an error function over joint offsets which measures the residual estimate error and learn a model to minimize this objective. An advantage of such an approach is that each timestep can get a better estimate of the joint distribution of the body pose given the previous estimate, incorporating some of the dependencies which exist between them. This idea of iterative refinement has been broadly adapted in many of the subsequent works. The methodology seems to build upon recent work on efficiently training deep networks by using residuals [6].

A feedback mechanism is used in [5] to model pose error in 3D images using a recurrent neural network (RNN). They find that using an RNN is able to improve performance by taking into account error dependencies across refinement steps. However, we have not found much in literature regarding a recurrent approach to directly model the joints.

Another notable approach, DeepCut, formulated the problem as an integer linear program and solves a global optimization problem [10]. The problem addresses the multi-person pose estimation which generalizes the single person task which we focus on in this paper. Their approach involves inference over a probabilistic model in which they use part detectors as unary factors and joint part dependencies as binary factors. In this light, it builds upon some of the previous ideas used in deformable parts models. However they use more expressive parts detectors, an adaptation of recent detection work [11], and a global objective function.

One limitation of using a standard CNN based model is that information, which can be used for fine grained localization, can be lost in the pooling layers of the network. This problem was addressed in Deep Pose with the cascade of networks, however this approach propagated errors from previous decisions.Two recent papers are able to achieve new state of the art results by addressing these limitations in their models. In Pose Machines, the authors take a multi-stage approach where they extract features from a local region in the image in one stage and then incorporate more global features in the latter stages [14]. Each stage is trained with using intermediate supervision and is able to iteratively refine the initial joint estimates with more information.

Similarly in Stacked Hourglass Networks, the authors propose a architecture which involves extracting features from images at different scales. These features are then concatenated with a deconvolution stage, creating an hourglass shape architecture [9]. By running the network across scales the architecture is theoretically able to incorporate global and local information into its estimates. Furthermore the stacking of multiple hourglasses can be seen as another form of iterative refinement.

3. Dataset

We used the MPII human pose estimation dataset, which is composed of 25,000 RGB images annotated with over 40,000 human poses [1]. The images were collected by sampling 3,913 videos from Youtube in various frames, which are at least 5 seconds apart, and filtering images which contained people. The annotations were obtained via crowd sourcing on Amazon Mechanical Turk and take the form of pixels coordinates corresponding to 16 joints for a given person in an image. A given image may contain multiple people, so we centered and cropped each image around a single person with some padding around their joint locations.

4. Methodology

We hypothesize that estimating joint locations sequentially will be able to better capture the dependency between joint locations since we explicitly condition on previous joint decisions. We choose to use a recurrent neural network model because it is able to express dependencies across sequences. Such models are common in natural language tasks which are naturally modelled as sequences. To test this hypothesis we choose our comparison baseline model to be a generic CNN which regresses all joint locations in one pass. This is similar to the method proposed in DeepPose, however we do not implement their cascade of network classifiers so we can isolate the effects of the sequence modelling without the added effect of refinement through higher resolution. For our experimental model we use a RNN whose inputs are feature which were extracted from the last convolutional layers of the CNN used for the base model. Comparing the two

Section	Type	Parameters
Layer 1	conv3-64	1,728
	conv3-64	36,864
	maxpool	0
Layer 2	conv3-128	73,728
	conv3-128	147,456
	maxpool	0
Layer 3	conv3-256	294,912
	conv3-256	589,824
	conv3-256	589,824
	maxpool	0
Layer 4	conv3-512	1,179,648
	conv3-512	2,359,296
	conv3-512	2,359,296
	maxpool	0
Layer 5	conv3-512	2,359,296
	conv3-512	2,359,296
	conv3-512	2,359,296
	maxpool	0
Layer 6	fc-4096	102,760,448
Layer 7	fc-4096	16,777,216
	fc-19	77,824
	softmax	0
Total		134,325,952

Table 1: VGGNet architecture and number of parameters

would thus show the difference in performance based on modelling the pose as a sequence. Full details on the models used are provided below.

4.1. CNN Base Architecture

We implemented a CNN based off of VGG16 as our base model [12]. VGG16 consists of a stack of convolutional layers followed by two fully connected layers. A chart of the full architecture along with the number of parameters associated with each layer is shown in Table 1. We modified the network by adding batch normalization layers after each non-linearity in order to improve training. The output of the fully connected layers was a 32 dimensional vector, corresponding to the (x, y) coordinates of 16 joints.

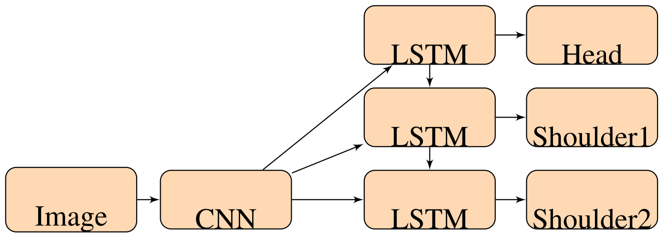

Figure 1: CNN-RNN architecture. A CNN is used to extract features from each image, which are then fed into an LSTM. The LSTM produces joint coordinates in pixel space.

4.2. CNN-RNN Architecture

We used the same base architecture to produce the initial hidden state and input to the RNN. A long short-term memory (LSTM) architecture was used for the RNN. The LSTM augments the traditional RNN formulation with a cell state, which allows gradients to flow additively through the network instead of multiplicatively. Concretely, at each timestep, the LSTM takes x_t, h_{t-1}, c_{t-1} and produces h_t, c_t via the following calculations:

$$\begin{aligned}
i_t &= \sigma(W^i x_t + U^i h_{t-1} + b^i) \\
f_t &= \sigma(W^f x_t + U^f h_{t-1} + b^f) \\
o_t &= \sigma(W^o x_t + U^o h_{t-1} + b^o) \\
g_t &= \text{tanh}(W^g x_t + U^g h_{t-1} + b^g) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \text{tanh}(c_t)
\end{aligned}$$

where i_t, f_t, o_t are the input, forget, and output gates. This structure improves gradient flow and has resulted in LSTMs being used to great success in recent literature for a wide variety of tasks.

We used two stacked LSTMs with a hidden state size of 128 and 16 timesteps. At each timestep, the hidden state of the LSTMs was projected into a two dimensional vector, which corresponded to the location of a single joint in the image (hence 16 timesteps for 16 joints).

4.3. LSTM with Attention

One issue with the current proposed architectures is that we need to downsize the input image to the input size of the CNN, which is 224 by 224. There is likely information lost at this stage which is acceptable for many classification tasks but not for precise localization. To deal with this problem we experiment with a soft attention for our CNN-RNN model. Soft attention was introduced by Xu et al. [15] and described a method to "guide" the focus of the recurrent neural network through the input image, over time. The mechanism attempted to relieve the feature encoder (a convolutional neural network in both our work and Xu et al.) of the need to compress the entire image into a single output vector. In order to implement attention, we made the following changes:

- We removed the two fully connected layers from our CNN, as suggested by Xu et al. Instead, we take the output of the last convolutional layer as input to the recurrent network. We call this output the *annotation vectors* or *annots*, with dimensions $(w*h, c)$ for a single image. c is the number of channels of the last convolutional filter, and w and h are the width and height of the image after passing through multiple max-pooling layers. In our case, $w = h = 14$ and $c = 512$. Thus we have 196 row vectors of size 512 each.

- At every timestep t, we calculate

  $$\alpha = MLP(h_{t-1}, annots)$$

  , where MLP is a multiplayer perceptron and h_{t-1} is the hidden state of the LSTM from the previous timestep. α is a single vector of dimension wh, or 196 in our case. The initial state of the LSTM is the mean of the annotation vectors.

- We calculate the next input to the LSTM as the weighted sum of the annotation vectors, $\alpha * annots$.

4.4. Training

We used l_2 loss aggregated over all timesteps. The stochastic optimizer was Adam, which is a recent optimizer which was shown empirically to be faster than stochastic gradient descent. The learning rate was set initially to 0.001, and was reduced by a factor of 2 every 1000 batch iterations. The model was implemented using TensorFlow and took one day to train on a Nvidia GTX 980 Ti.

4.5. Evaluation

To compare our results to previous works we use the standard percentage of correct key points (PCKh) measure. The h denotes that the measure is normalized by the length of the head in each image. This measure is calculated by counting the percentage of correct localizations for a given part across all images. A estimate is classified as correct if it falls within a certain distance of the ground truth annotation. That distance is defined to be half the length of the persons head in a given image.

Additionally we calculate the root mean squared error (RMSE) of the joint coordinates across all joints and training examples. This measure calculates the the average pixel distance of all visible joints in the image.

5. Results

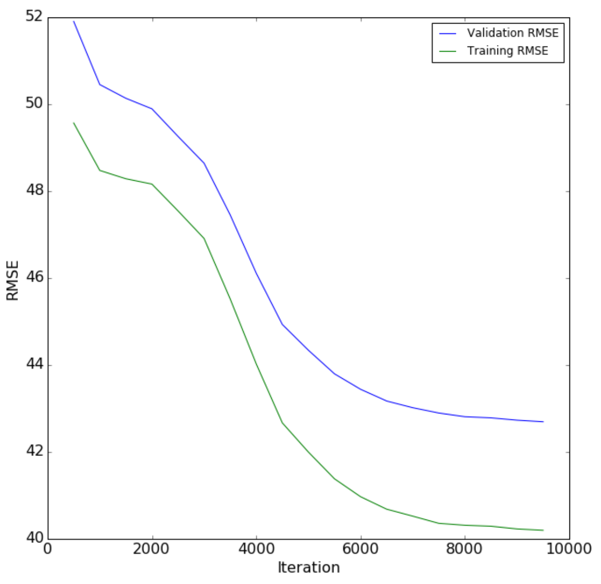

Figure 2: Training and validation RMSE over time for the CNN base model.

Method	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	Total
Iterative Error[2]	95.7	91.7	81.7	72.4	82.8	73.2	66.4	81.3
DeepCut[10]	94.1	90.2	83.4	77.3	82.6	75.7	68.6	82.4
Pose Machines[14]	97.7	94.5	88.3	83.4	87.9	81.9	78.3	87.9
Stacked Hourglass[9]	97.6	95.4	90.0	85.2	88.7	85.0	80.6	89.4
Our Base Model	52.0	40.6	22.5	16.8	31.7	21.2	16.0	32.8
Our LSTM Model	39.2	25.2	12.2	9.7	18.7	16.6	7.5	21.5

Table 2: Model Performance Comparison (PCKh)

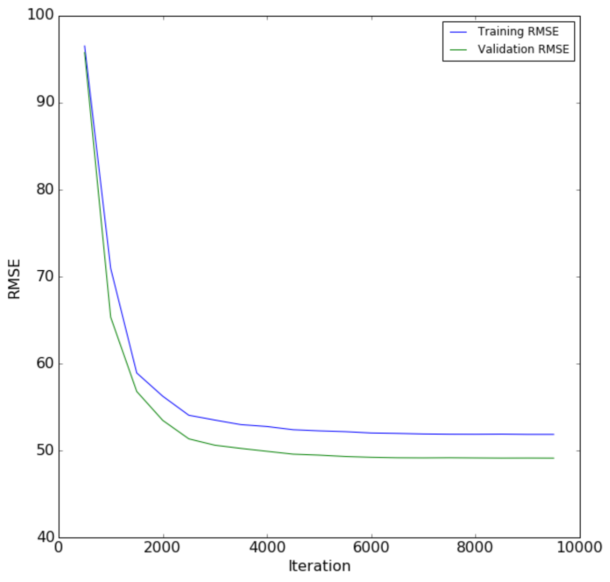

Figure 3: Training and validation RMSE over time for the RNN-CNN model.

Our experiments show that the CNN base architecture performed much better than the CNN-RNN one. The RMSE over time during training, for the base model, are shown in Figure 2. We can see that RMSE error decreased for both the training and the validation set but levelled off to still a fairly high level. The training process for the CNN-RNN model followed the same trend but levelled off much earlier as shown in 3.

To get a better understanding of what the model learned we do a qualitative analysis of the output of each model. Sample output images where we connect all the joints to form the complete pose are shown in Figures 4 for the base model. We can see that even through there are high errors in the localization of the joint coordinates, the pose does adapt in reasonable ways to the underlying image. This however is not the case when we examine similar output form the CNN-RNN model as shown in Figure 5. It seems that the model here simply learned the average pose across all images, and makes minimal adjustments to that average pose at test time.

Adding attention to the LSTM did not, unfortunately, improve or change its performance. Thus, we did not include the attention results for sake of conciseness, and the results shown in the following sections are for the simple LSTM model.

Finally, PCKh measures and comparisons are provided in Table 2 to show the relative performance against a selection of the current state of the art models.

6. Discussion

We suspect that the recurrent network regressed toward the most common pose in the dataset. This might occur, for example, if the RNN had not learned to fully utilize the output of the CNN or the CNN was not well-trained, and thus the model was calculating:

$$\arg\max_{y_i} p(y_i|y_{i-1}, y_{i-2}, ..., y_1), i = 1, 2, ..., n$$

instead of:

$$\arg\max_{y_i} p(y_i|y_{i-1}, y_{i-2}, ..., y_1, x), i = 1, 2, ..., n$$

.

This would be problematic because it may be the case that knowing y_1 alone may be insufficient to

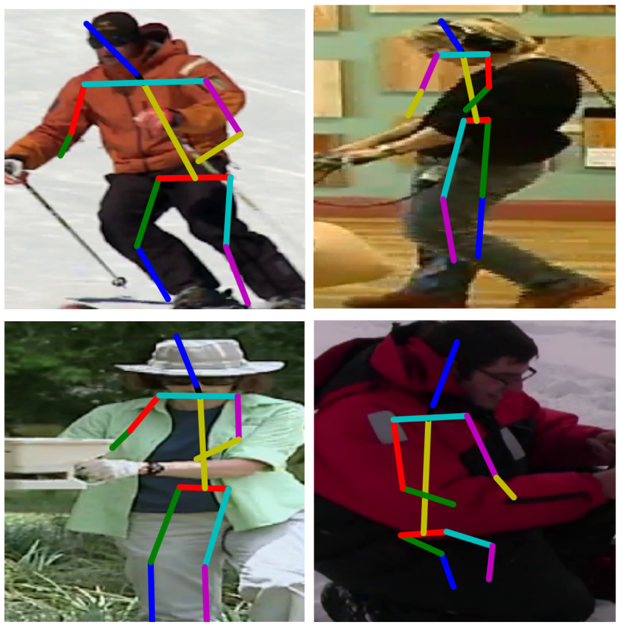

Figure 4: Sample predictions for the CNN base model. The model seems to be learning effectively.

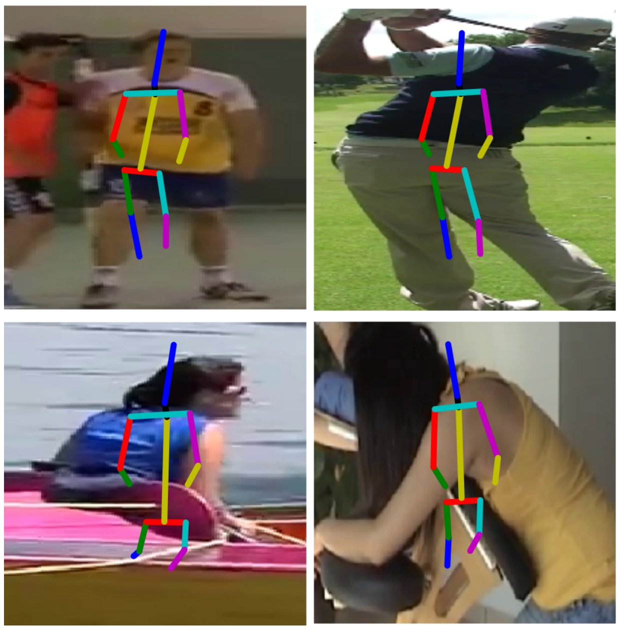

Figure 5: Sample predictions for the CNN-RNN model. The estimated poses seem to be shifting slightly toward ground truth, but seem to contain noticeable "inertia."

estimate the correct y_2. In this case, estimating the most common pose would make sense. Having a larger training set may have alleviated this problem. We also suspect that errors compounded through time in the recurrent network. Specifically, if $\arg\max_{y_2} p(y_2|y_1, x)$ is calculated incorrectly, then it is likely that $\arg\max_{y_3} p(y_3|y_2, y_1, x)$ would also be calculated incorrectly. The base model did not explicitly model these conditional probabilities so it would not suffer from this problem.

Another possible explantion for why the LSTM performed poorly is that modeling joints as a sequence placed too many constraints on the model. In other words, the recurrent formulation of the problem was incorrect. This might explain why adding attention did not improve the performance of the LSTM. Attention gives more information to the recurrent network, but if a recurrent structure was incorrect to begin with, then it would not help.

7. Conclusion

In this project, we investigated the possibility of modelling pose estimation as a sequence task. We tested this hypothesis via use of a convolutional network linked to a recurrent one. For comparison, we also tested just the convolutional network on the same task. Our preliminary results show that the CNN-RNN performed worse than the CNN. We also found that adding attention did not improve the result of the CNN-RNN. Further work can will expand upon specific reasons why the the model performed worse than the baseline with additional quantitative results to test our hypothesis. Code for this project can be found at `https://github.com/electric26/posernn`.

References

[1] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3686–3693. IEEE, 2014.

[2] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. *arXiv preprint arXiv:1507.06550*, 2015.

[3] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[4] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[5] A. Haque, B. Peng, Z. Luo, A. Alahi, S. Yeung, and L. Fei-Fei. Viewpoint invariant 3d human pose estimation with recurrent error feedback. *arXiv preprint arXiv:1603.07076*, 2016.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[7] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] I. Lifshitz, E. Fetaya, and S. Ullman. Human pose estimation using deep consensus voting. *arXiv preprint arXiv:1603.08212*, 2016.

[9] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *arXiv preprint arXiv:1603.06937*, 2016.

[10] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. *arXiv preprint arXiv:1511.06645*, 2015.

[11] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] A. Toshev and C. 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.

[14] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. *arXiv preprint arXiv:1602.00134*, 2016.

[15] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

[16] 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.