An Efficient Convolutional Network for Human Pose Estimation

Umer Rafi[1]
rafi@vision.rwth-aachen.de
Juergen Gall[2]
gall@informatik.uni-bonn.de
Bastian Leibe[1]
leibe@vision.rwth-aachen.de

[1] Visual Computing Institute
RWTH Aachen University
Aachen, Germany

[2] Computer Vision Group
Institute of Computer Science III
University of Bonn, Germany

Abstract

In recent years, human pose estimation has greatly benefited from deep learning and huge gains in performance have been achieved. However, to push for maximum performance recent approaches exploit computationally expensive deep network architectures, train on multiple datasets, apply additional post-processing and provide limited details about used design choices. This makes it hard not only to compare different methods and but also to reproduce existing results .

In this work, we propose an efficient deep network architecture that is trained efficiently with a transparent procedure and exploits the best available ingredients from deep learning with low computational budget. The network is trained only on the same dataset without pre-training and achieves impressive performance on popular benchmarks in human pose estimation.

1 Introduction

Convolutional networks have raised the bar substantially for many computer vision benchmarks. Human pose estimation is such an example where methods based on convolutional networks dominate the leader boards [5, 14]. Despite of the recent huge success in human pose estimation, a direct comparison between the architectures remains difficult. For architectures that do not provide the source code for training and testing, the reproducibility of the results can be become very difficult due to small details that might be essential for the performance like the used image resolution or the exact form of data augmentation. Very often pre-trained models are used that are fine-tuned on the benchmark datasets, making it difficult to compare them with methods that are trained from scratch on benchmarks and therefore on less training data. Another issue is the increasing complexity of the models for human pose estimation. Despite of the impressive accuracy they achieve, computationally expensive network architectures with a large memory footprint can be impractical for many applications since they require high-end graphics cards.

In this work, we propose an efficient network architecture for human pose estimation that exploits the best design choices for network architectures with a low memory footprint and train it by using the best ingredients for efficient learning. An important design choice

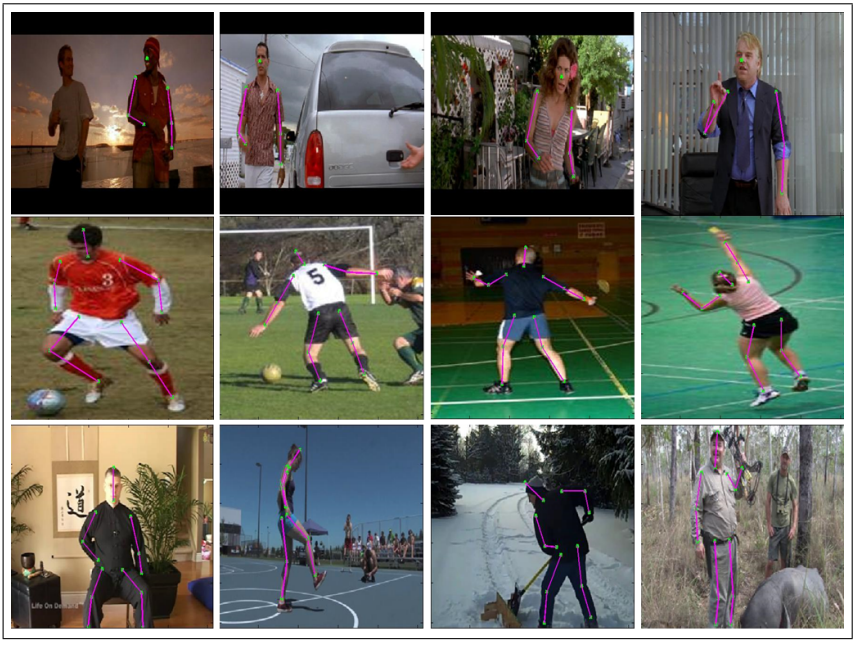

Figure 1: Our Qualitative results for FLIC, LSP and MPI datasets respectively.

is to learn features in different layers at multiple scales. This has been recently exploited in the context of classification by Szegedy et al. [25] through the introduction of inception layers. Surprisingly, very little attention has been paid on exploiting this concept for human pose estimation. Similarly, learning features at multiple image resolutions has been shown to be very effective in [27] for human pose estimation as it helps the network to use a larger context for difficult body joints like wrists and ankles. In our network, we combine both ideas to achieve maximum performance. Another important aspect is the use of features from the middle layers in addition to features from the last layer. Coarse features from the last layer contain information about semantics but are poor at localization due to pooling. Features from the middle layers contain good localization information due to less pooling. Long et al. [17] exploited it for semantic segmentation and [1] used it for object localization. Similarly, using context around features from the last layer has shown to be very effective in the context of semantic segmentation [16]. Our network also exploits context around features from last layers along with features from a middle layer. Other recent advances in deep learning, e.g, adam optimizer, exponential learning rate decay, batch normalization and extensive data augmentation, also have shown to provide further benefits for the overall performance [13]. We therefore exploit the above ingredients to add additional gains to the performance.

Based on the design choices, we propose a network architecture for human pose estimation that is efficient to train and has a low memory footprint such that a mid-range GPU is sufficient. Yet, our network architecture achieves state-of-the-art accuracy on the most popular benchmarks for human pose estimation, indicating that very complex architectures might not be needed for the task. For best comparison and reproducibility, we evaluated the network using the protocols of the benchmarks without any pre-training or post-processing.

We will further release the learned models for all benchmarks and the source code for training and testing such that the proposed network serves as an up-to-date baseline for more complex models.

2 Related Work

Human Pose Estimation has generated a lot of research literature. The research can be divided into CNN and Non-CNN approaches

Non-CNN approaches. The classical approaches are based on the pictorial structures model [1, 2, 7, 10, 14, 18, 19, 32] that uses a tree-structured graphical model to encode spatial dependencies between neighbouring joints. The approaches have shown to be successful when all joints on the person are visible and may suffer from double counting in case of occlusions. Another line of research is based on Hierarchical models [24, 26] that first detect larger parts in the image and then condition the detection of difficult smaller parts on detected larger parts. Complex Non-tree Models [23] have also been used that model spatial dependencies between non-neighbouring parts and then rely on loopy belief propagation with approximate to reason about different body joint positions in the image to make inference tractable. Recently sequentail prediction machines [21] have been proposed that enjoy the benefit of modelling complex spatial dependencies between joints with an efficient inference procedure.

CNN Approaches. Toshev et al [29] first used CNNS to directly regress point estimates for different body joints. Tompson et al [27] showed that predicting belief maps compared to point estimates has a huge impact on performance. In their later work, Tompson et al [28] further improved the performance by introducing a cascade architecture that counters the effect of pooling. [4] showed that predicting fixed corrections to current estimates for different joints when provided with initial estimates can be superior to directly predicting the point estimates for joints. Very recently [30] used inference machines architecture with features from cnns. [20] go one step further and use cnns for multi-person pose estimation.

The closest to our method is [27] since it also uses a fully convolutional architecture that is trained from scratch on multiple image resolutions on the same dataset . Contrary to their method, we don't use any graphical model on the top. We don't use any cascades as in [28, 29]. We train our model from scratch on the same dataset as compared to [4, 20, 30]. We don't use background model and implicit spatial dependencies as in [30]. Our network has a low memory footprint of 3GB compared to 6GB from [30] which allows our network to be re-used on a mid range GPU like GTX980.

3 Fully Convolutional Deep Network for Human Pose Estimation

In 2D human pose estimation the goal is to localize the (x, y) positions of different body joints. Recent work [27, 29] has shown that regressing point estimates for different body joints may be sub-optimal and a better strategy is to use fully convolutional deep architectures to predict dense belief maps for each body joint that contain per-pixel likelihood of

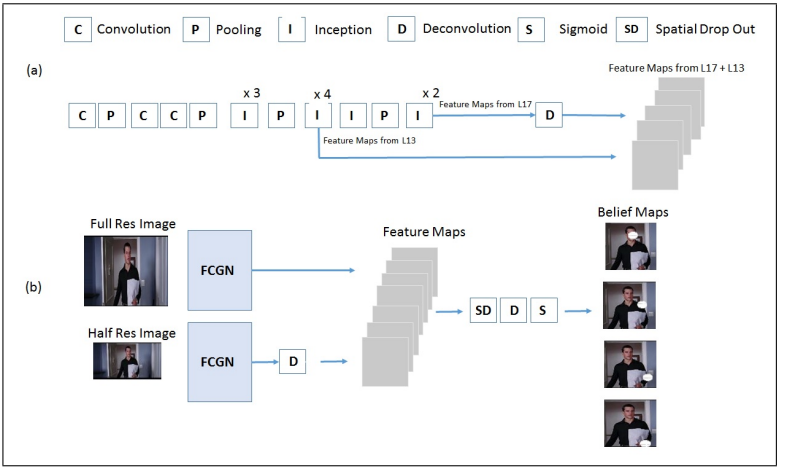

Figure 2: (a) FCGN : Our adaptation of BN inception [13] network. (b) Our multi resolution network consisting of HalfRes FCGN and FullRes FCGN.

each body joint. Similar approach has been applied in [8, 17] for segmentation and depth estimation and has produced promising results.

One caveat with fully convolutional networks is their hunger for memory. Any contemporary network, that is not well-optimised for memory usage, when converted to fully convolutional network will have a requirement of large computational budget and will require an expensive GPU with training time in days even for a relatively small dataset of few thousand images. The contemporary BN-Inception network [13] for image classification is very well optimised for utilising computational resources and at the same time has impressive performance. To this end, we adapt [13] into a fully convolutional network for predicting belief maps for body joints. This section is organized as follows. In Section 3.1 we describe our adaptation of the batch normalized GoogleNet architecture [13]. Sections 3.2 and 3.3 describe the training and data augmentation procedures respectively.

3.1 Network Architecture

Our adaptation of [13] is shown in Figure 2(a) which we refer in this work as Fully Convolutional Google Net (FCGN). We use the first 17 layers and remove the average pooling, drop-out, linear and soft-max layers from the last stages of the network. We add a skip connection to combine feature maps from layer 13 with feature maps from layer 17. We upsample the feature maps from layer 17 to the resolution of the feature maps from layer 13 by a deconvolution filter of both size and stride 2. The output of FCGN consists of coarse feature maps from layer 13 and 17 that have 16 times lesser resolution than the input image due to max/average pooling by a factor of 16.

Our main multi-resolution network is shown in Figure 2(b). It uses two FCGN with shared weights, where each FCGN takes the same image at a different resolution and produces coarse feature maps as described above. The feature maps from the Half Res Image FCGN are upsampled to the resolution of Full Res Image FCGN feature maps by a deconvolution filter of both stride and filter size of 2.

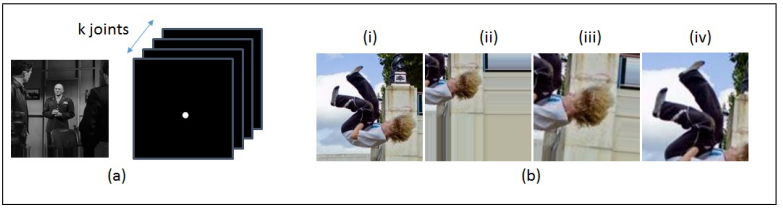

Figure 3: (a) Image with ground truth binary belief maps for joints. (b) Illustration of data augmentation procedure : (i) Original Image (ii) Person shifted to upper left corner (iii) Person warped (iv) Person shifted back to center of cropped image.

The coarse feature maps from HalfRes FCGN and FullRes FCGN are then directly upsampled to belief maps for different body joints by using a larger deconvolution filter of size 32 and stride 16. By using deconvolution filter of size 32 we automatically exploit the context of neighbouring pixels in coarse feature maps for predicting belief maps for joints. The belief maps are then normalized by using a sigmoid. We also use spatial drop out [28] before upsampling to further regularize our network. The details for the network are provided in supplementary material.

3.2 Training

We denote a training example as $(I, \{b_j(x,y)\})$, where $b_j(x,y) \in \mathbb{R}^{w \times h}$ is the ground-truth 2D belief map for joint j and contains the per-pixel likelihood of joint j for each pixel (x,y) in image I of width w and height h. The belief map represents a binary image containing a disk of 1's with radius 'r' centred at the ground-truth (x,y) location of joint j and zero elsewhere, as shown in Figure 3 (a).

We are mainly interested in learning the appearances of different body joints and not background. Since our belief maps consist of 0's (background) and 1's (body joints), we want to use an error function with an inherent property of focussing on target values 1's only. To this end, given training samples $N = \{(I, \{b_j(x,y)\})\}$, we minimize the binary cross entropy between the ground-truth and predicted belief maps for k joints in each training image I as follows:

$$\arg\min_{w} \quad -\sum_{j=1}^{k} \sum_{x,y} b_j(x,y) \log(\hat{b_j}(x,y)) + (1 - b_j(x,y)) \log(1 - \hat{b_j}(x,y)), \qquad (1)$$

where w and $\hat{b_j}(x,y)$ are the parameters of our network and the predicted belief maps, respectively. The weights of the network are then learned using back propagation and the Adam optimizer.

3.3 Data Augmentation

Data Augmentation is an essential ingredient in Deep networks and has significant impact on performance. Data augmentation for classification is different from human pose estimation since in the later one also needs to warp the joints accordingly. In recent work on Human Pose Estimation with Deep Networks [27, 30] only the type and amount of augmentation is mentioned.

In this section we explain our procedure[1] of data augmentation. Each form of augmentation, i.e, scaling, rotation, translation, reflection is an image warp operation. The warps are represented by a warp matrix containing random values of scaling, rotation, flip and translation. The matrix is then used to compute the pixel values in warped image through a combination of backward warping and an interpolation method. The warped joints are computed by multiplication of warp matrix with unwarped joints.

The naive application of the above procedure results in applying the warp around the upper left image corner and the person of interest may be located elsewhere in the image. Our goal is to apply the warp around the center position of the person of interest. To this end, we apply a warp W that is computed as follows:

$$W = \begin{pmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} rscos\theta & -ssin\theta & 0 \\ ssin\theta & scos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -x_p + t_x \\ 0 & 1 & -y_p + t_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{2}$$

The warp on the right first shifts the perturbed center of person $(x_p + t_x, y_p + t_y)$ to the upper left corner of the image. The perturbed operation implicitly produces the effect of applying a random translation. The middle warp then applies random scaling s, rotation$(cos\theta, sin\theta)$ and reflection r . The left warp then shifts back the warped person to the center (c_x, c_y) of the cropped image. Figure 3(b) illustrates the procedure.

One caveat with extensive augmentation is that in many random warps some warped joints will be outside of the image boundary. One solution is to use less augmentation. However, this can have a significant impact on performance. To this end, we keep on randomly generating warps and then only select the warps for which all warped joints end up inside the image boundary.

4 Experiments

In this section we present our quantitative results on standard benchmarks including the MPII, LSP and FLIC datasets. Our experimental settings are as follows: (i) We crop the images in all datasets to a resolution of 256×256. For training images in all datasets, we crop around the person's center computed by using the ground-truth joint positions. For test images in all datasets we crop around the rough person location when available, otherwise we crop around the center of the image. (ii) We train the network from scratch without any pre-training with a learning rate of 0.001 with an exponential decay of 0.96 applied every 50 epochs. We train the network for 120 epochs for each dataset. (iii) We use scaling $\in \{0.5, 1.5\}$, translation $\in \{-20, 20\}$, rotation $\in \{-20°, 20°\}$ and horizontal flipping with probability 0.5 for data augmentation.

We use the Torch 7 [6] framework for our experiments. Unless otherwise stated we report results for the above-mentioned settings. For evaluation measure on LSP and FLIC we use the PCK measure from [22]. For MPII we use the PCKh measure from [3].

4.1 FLIC dataset

The FLIC dataset consists of 3'987 training images and 1'016 test images. Following the standard practice we report PCK @ **0.2** only for wrists and elbows. Our quantitative results

[1] The code will be released

Method	Elbows	Wrists
Tompson et al [28]	93.1	89.0
Toshev et al [29]	92.3	82.0
Chen et al [5]	95.3	92.4
Wei et al [30]	97.59	95.03
Ours	96.06	89.66
Ours scale normalised	98.62	94.88
Ours with limited augmentation	92.37	81.89
Ours without learning rate decay	94.53	86.51

Table 1: Comparison over Quantitative Results for FLIC over Elbows and Wrists with state-of-the-art using PCK @ **0.2**.

are shown in Table 1. Our model takes 10 hours to train on FLIC dataset using GTX 980 GPU.

Effect of Scale normalization. For FLIC rough torso detections for training and testing are also available. From these detections, we compute the height of the detected torso and use it for the scale approximation for each image. We then normalize all training and test images to the same scale by re-normalizing the height of the detected torso in each image with 200 pixels. Results are shown in Table 1. Those results show that using scale information, when available, can provide significant gains in performance especially for wrists from **89.66** to **94.88**. Wei *et al*. [30] also use scale information, still we outperform them for elbows with **98.62** compared to **97.59** and reach **94.88** for wrists, which is very close to their performance of **95.03** despite our not using any background model and implicit spatial dependencies. On the other hand, our model has a low memory footprint of 3GB and could be run on a moderate-range GPU, in contrast to their model that has a high memory footprint of 6GB.

Effect of limited Data Augmentation. We evaluate the effect of limited augmentation by reducing the ranges for scaling $\in \{0.7, 1.3\}$, translation $\in \{-5, 5\}$, rotation $\in \{-5°, 5°\}$ on FLIC dataset. Performance drops from **96.06** to **92.37** for elbows and from **89.66** to **81.89** for wrists (see Table 1). Those results show that extensive data augmentation is indeed important for achieving best performance.

Effect of exponential learning rate decay. In exponential learning rate decay the learning rate is reduced automatically as a part of the training procedure and helps for getting a better performance. We compare the effect of exponential decay *vs*. no decay at all within the same number of epochs. Again performance drops from **96.06** to **94.53** for elbows and from **89.66** to **86.51** for wrists, as shown in Table 1. This shows that exponential learning rate decay provides better performance *vs*. no decay at all within the same number of epochs.

4.2 Leeds Sports Pose (LSP) dataset

The LSP dataset [14] consists of 1'000 training and 1'000 test images. The extended LSP dataset [15] consists of an additional 10'000 training images. In this work, we use 11'000 training images from both LSP and extended LSP . Quantitative results at PCK @ **0.2** are

Method	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	PCK
Tompson et al [27]	90.6	79.2	67.9	63.4	69.5	71.0	64.2	72.0
Fan et al [9]	92.4	75.2	65.3	64.0	75.7	68.3	70.4	73.0
Carreira et al [4]	90.5	81.8	65.8	59.8	81.6	70.6	62.0	73.1
Chen et al [6]	91.8	78.2	71.8	65.5	73.3	70.2	63.4	73.4
Yang et al [31]	90.6	78.1	73.8	68.8	74.8	69.9	58.9	73.6
Wei et al [30]	.	.	.	.	.	.	.	84.32
Pishchulin et al [20] + MPII	97.0	91.0	83.8	78.1	91.0	86.7	82.0	87.1
Wei et al [30] + MPII	97.8	92.5	87.0	83.9	91.5	90.8	89.9	90.5
Ours with single resolution only	96.0	85.8	79.6	73.7	86.3	80.8	77.7	82.8
Ours with feature maps from last layer only	95.4	83.7	74.8	70.4	84.4	78.2	74.2	80.2
Ours								83.86

Table 2: Comparison over Quantitative Results for LSP with state-of-the-art using PCK @ **0.2**.

shown in Table 2. Our performance **83.86** is very close to **84.32** from [30] when they train only on the same dataset with an additional background model and implicit spatial model on top. Additionally, our model requires almost half as much memory as their model.

Effect of middle layer features. We evaluate the impact of using feature maps from the middle layers on performance. To that end, we remove the skip layer connection from FCGN in Section 3.1. Overall performance drops from **83.86** to **80**, as shown in Table 2. Those results show that features from the middle layer are essential for better performance for localization.

Effect of Multi-Resolution Features. We also evaluate the impact of using features at multiple resolutions. To that end, we remove the Half Res Image FCGN from our multi-resolution network and train it with Full Res Image FCGN only. Performance gets lower from **83.86** to **82.8**, as shown in Table 2.

4.3 MPII Human Pose Dataset

The MPII Human Pose Dataset [3] is a very challenging dataset and consists of around 40'000 images of people. We use 25'925 images for training and 2'958 validation images according to the train/validation split from [28]. We evaluate on the 7'247 Single Person test images by uploading our results on the MPII server. The dataset provides a rough scale and person location for both training and test images. We crop test images around the given rough person location. We normalize both training and test images to the same scale by using the provided rough scale information. Our model takes 3 days to train on the MPII dataset using GTX 980 GPU.

Our quantitative results are shown in Table 3. Our method outperforms most of the recent state-of-the art methods and is competitive to [30] without using any pre-training, post-processing, background models and any form of implicit/explicit spatial dependencies mainly due to the effectiveness of our network design and usage of best deep learning ingredients.

Method	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	PCKh
Hu et al [17]	95.0	91.6	83	76.6	81.9	74.5	69.5	82.4
Carreira et al [4]	95.7	91.7	81.7	72.4	82.8	73.2	66.4	81.3
Tompson et al [28]	96.1	91.9	83.9	77.8	80.9	72.3	64.8	82.0
Pischulin et al [20]	94.1	90.2	83.4	77.3	82.6	75.7	68.6	82.4
Wei et al [30] + LSP	97.8	95.0	88.7	84.0	88.4	82.8	79.4	88.5
Ours	96.8	93.1	85.2	79.7	85.8	78.7	71.8	85.1

Table 3: Comparison over Quantitative Results for MPI with recent state-of-the-art using PCKh @ **0.5**.

5 Conclusion

In this work, we have proposed a low memory footprint deep network for human pose estimation that can be trained efficiently on a mid-range GPU and achieves state-of-the-art accuracy on popular benchmarks for human pose estimation. We have exploited the best ingredients from deep learning and achieved impressive performance even on a data set like FLIC where limited training data is available without using pre-training and extra training data from MPII. We will release our source code for training and testing and trained models, so that others can also benefit from our design guidelines and use them for further progress in the field of human pose estimation.

6 Acknowledgement

The work in this paper is funded by the EU projects STRANDS (ICT-2011-600623) and SPENCER (ICT-2011-600877).

References

[1] M. Andriluka, S. Roth, and B. Schiele. Pictorial Structures revisited: People Detection and articulated pose estimation. In *CVPR*, 2009.

[2] M. Andriluka, S. Roth, and B. Schiele. Monocular 3D pose estimation and tracking by detection. In *CVPR*, 2010.

[3] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2D Human Pose Estimation: New BenchMark. In *CVPR*, 2014.

[4] J. Carreira, P. Agarwal, K. Fragkiadaki, and J. Malik. Human Pose Estimation with Iterative Error Feedback. In *CVPR*, 2016.

[5] X. Chen and A. Yuille. Articulated Pose Estimation by a Graphical Model with Image Dependent Pairwise Relations. In *NIPS*, 2014.

[6] R. Collobert, K. Kavukcuoglu, and C. Farabe. Torch7: A matlab-like environment for machine learning. In *NIPS'11 Workshop on BigLearn*, 2011.

[7] M Dantone, J Gall, C Leistner, and L. van Gool. Human Pose Estimation using Body Parts Dependent Joint Regressors. In *CVPR*, 2013.

[8] D. Eigen, C. Puhrsch, and R .Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. In *NIPS*, 2014.

[9] X. Fan, K. Zheng, Y. Lin, and S. Wang. Combining local appearance and holistic view: Dual Source deep neural networks for human pose estimation. In *CVPR*, 2015.

[10] P. Felzenswalb and D. Huttenlocher. Pictorial Structures for object recognition. In *CVPR*, 2005.

[11] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Hypercolumns for Object Segmentation and Fine-grained Localization. In *CVPR*, 2015.

[12] P. Hu and D. Ramanan. Bottom Up and Top Down Reasoning with Hierarchical Rectified Gaussians. In *CVPR*, 2016.

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

[14] S. Johnson and M. Everingham. Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation. In *BMVC*, 2010.

[15] S. Johnson and M. Everingham. Learning Effective Human Pose Estimation from Inaccurate Annotation. In *CVPR*, 2011.

[16] G. Lin, C. Shen, A. van dan Hengel, and I. Reid. Efficient piece wise training of deep strucutred models for semantic segmentation. In *CVPR*, 2016.

[17] J. Long, E. Shelhamer, and T. Darrel. Fully Convolutional Netowrks for Semantic Segmentation. In *CVPR*, 2015.

[18] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele. Poselet Conditioned Pictorial Structures. In *CVPR*, 2013.

[19] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele. Strong appearance and expressive spatial models for human pose estimation. In *ICCV*, 2013.

[20] 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. In *CVPR*, 2016.

[21] V. Ramakrishna, D. Munoz, M. Hebert, J. Bagnell, and Y. Sheikh. Articulated Pose Estimation via Inference Machines. In *ECCV*, 2014.

[22] B. Sapp and B. Taskar. MODEC : Multimodel Decomposable Models for Human Pose Estimation. In *CVPR*, 2013.

[23] L. Sigal and M. Black. Measure locally, reason globally:Occlusion-sensitive articulated pose estimation. In *CVPR*, 2006.

[24] M. Sun and S. Savarese. Articulated part-based model for joint object detetion and pose estimation. In *ICCV*, 2011.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *CVPR*, 2015.

[26] Y. Tian, L. Zitnick, and S. Narasimhan. Exploring the spatial hierarchy of mixture models for human pose estimation. In *ECCV*, 2012.

[27] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation. In *NIPS*, 2014.

[28] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient Object Localization Using Convolutional Networks. In *CVPR*, 2015.

[29] A. Toshev and C. Szegedy. Deep Pose: Human Pose Estimation via Deep Neural Network. In *CVPR*, 2014.

[30] S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional Pose Machines. In *CVPR*, 2016.

[31] W. Yang, W. Ouyang, H. Li, and X. Wang. End-to-End Learning of Deformable Mixture of Parts and Deep Convolutional Neural Networks for Human Pose Estimation. In *CVPR*, 2016.

[32] Y Yang and D Ramanan. Articulated Pose Estimation using Flexible Mixtures of Parts. In *CVPR*, 2011.