Appears in
Asian Conference on Computer Vision (ACCV), Singapore, 2014

3D Human Pose Estimation from Monocular Images with Deep Convolutional Neural Network

Sijin Li Antoni B. Chan

Department of Computer Science
City University of Hong Kong

Abstract. In this paper, we propose a deep convolutional neural network for 3D human pose estimation from monocular images. We train the network using two strategies: 1) a multi-task framework that jointly trains pose regression and body part detectors; 2) a pre-training strategy where the pose regressor is initialized using a network trained for body part detection. We compare our network on a large data set and achieve significant improvement over baseline methods. Human pose estimation is a structured prediction problem, i.e., the locations of each body part are highly correlated. Although we do not add constraints about the correlations between body parts to the network, we empirically show that the network has disentangled the dependencies among different body parts, and learned their correlations.

1 Introduction

Human pose estimation is an active area in computer vision due to its wide potential applications. In this paper, we focus on estimating 3D human pose from monocular RGB images [1–3]. In general, recovering 3D pose from 2D RGB images is considered more difficult than 2D pose estimation, due to the larger 3D pose space, more ambiguities, and the ill-posed problem due to the irreversible perspective projection. Although using depth maps has been shown to be effective for 3D human pose estimation [4], the majority of the media on the Internet is still in 2D RGB format. In addition, monocular pose estimation can be used to aid multi-view pose estimation.

Human pose estimation approaches can be classified into two types—model-based generative methods and discriminative methods. The pictorial structure model (PSM) is one of the most popular generative models for 2D human pose estimation [5, 6]. The conventional PSM treats the human body as an articulated structure. The model usually consists of two terms, which model the appearance of each body part and the spatial relationship between adjacent parts. Since the length of a limb in 2D can vary, a mixture of models was proposed for modeling each body part [7]. The spatial relationships between articulated parts are simpler for 3D pose, since the limb length in 3D is a constant for one specific subject. [8] proposes to apply PSM to 3D pose estimation by discretizing the space.

However, the pose space grows cubicly with the resolution of the discretization, i.e., doubling the resolution in each dimension will octuple the pose space.

Discriminative methods view pose estimation as a regression problem [4, 9–11]. After extracting features from the image, a mapping is learned from the feature space to the pose space. Because of the articulated structure of the human skeleton, the joint locations are highly correlated. To consider the dependencies between output variables, [11] proposes to use structured SVM to learn the mapping from segmentation features to joint locations. [9] models both the input and output with Gaussian processes, and predicts target poses by minimizing the KL divergence between the input and output Gaussian distributions.

Instead of dealing with the structural dependencies manually, a more direct way is to "embed" the structure into the mapping function and learn a representation that disentangles the dependencies between output variables. In this case models need to discover the patterns of human pose from data, which usually requires a large dataset for learning. [4] uses approximately 500,000 images to train regression forests for predicting body part labels from depth images, but the dataset is not publicly available. The recently released Human3.6M dataset [12] contains about 3.6 million video frames with labeled poses of several human subjects performing various tasks. Such a large dataset makes it possible to train data-driven pose estimation models.

Recently, deep neural networks have achieved success in many computer vision applications [13, 14], and deep models have been shown to be good at disentangling factors [15, 16]. Convolutional neural networks are one of the most popular architectures for vision problems because it reduces the number of parameters (compared to fully-connected deep architectures), which makes training easier and reduces overfitting. In addition, the convolutional and max-pooling structure enables the network to extract translation invariant features.

In this paper, we consider two approaches to train deep convolutional neural networks for monocular 3D pose estimation. In particular, one approach is to jointly train the pose regression task with a set of detection tasks in a heterogeneous multi-task learning framework. The other approach is to pre-train the network using the detection tasks, and then refine the network using the pose regression task alone. To the best of our knowledge, we are the first to show that deep neural networks can be applied to 3D human pose estimation from single images. By analyzing the weights learned in the regression network, we also show that the network has discovered correlation patterns of human pose.

2 Related Work

There is a large amount of literature on pose estimation, and we refer the reader to [17] for a review. In the following, we will briefly review recent regression networks and pose estimation techniques.

[18] trains convolutional neural networks to classify whether a given window contains one specific body-part, and then detection maps for each body-part are calculated by sliding the detection window over the whole image. A spatial

model is applied to enforce consistencies among all detection results. [19] applies random forests for joint point regression on depth maps. The tree structures are learned by minimizing a classification cost function. For each leaf node, a distribution of 3d offsets to the joints is estimated for pixels reaching that node. Given a test image, all the pixels are classified into leaf nodes, and offset distributions are used for generating the votes for joint locations.

In [20], a cascade neural network is proposed for stage-by-stage prediction of facial points. Networks in the later stages will take inputs centered at the predictions of the previous stage, and it was shown that cascading the networks helps to improve the accuracy. Similarly, [21] cascades 3 stages of neural networks for estimating 2D human pose from RGB images. In each stage, the network architecture is similar to the classification network in [13], but is applied to joint point prediction in 2D images The networks in the later stages take higher resolution input windows around the previous predictions. In this way, more details can be utilized to refine the previous predictions. The cascading process assumes that the prediction can be made accurately by only looking at a relatively small local window around the target joints. However, this is not the case for 3D pose estimation. To estimate the joint locations in 3D, the context around the target joints must be considered. For example, by looking at the local window containing an elbow joint, it is very difficult to estimate its position in 3D. In addition, when body parts are occluded, local information is insufficient for accurate estimation. Therefore, our networks only contain one stage. To take into account contextual features, we design the network so that each node in the output layer receives contributions from all the pixels in the input image.

Previous works on using neural networks for 3D pose estimation from images mainly focuses on rigid objects or head pose. [22] uses fully connected networks for estimating the pose parameters of 3D objects in single images. However, [22] is only applicable to 3D rigid objects, such as cups and plates, which are very different from 3D articulated objects such as humans. [23] uses convolutional neural networks to detect faces, and estimates the head pose using a manually-designed low-dimensional manifold of head pose. In contrast to these previous works, we train our network to estimate the 3D pose of the whole human, which is a complex 3D articulated object. Finally, [24] uses an implicit mixture of conditional restricted Boltzmann machines to model the motion of 3D human poses (i.e., predicting the next joint points from the previous joint points), and applies it as the transition model in a Bayesian filtering framework for 3D human pose tracking. In contrast, here we focus on estimating the 3D pose directly from the image, and do not consider temporal information.

Previous works have demonstrated that learning body part labels could help to find better features for pose estimation [4, 25]. In [4], random forests are used for estimating the body part labels from depth images. Given the predictions of labels, mean shift is applied to obtain the part locations. [25] trains a multi-task deep convolutional neural network for 2D human pose estimation, consisting of the pose regression task and body part detection tasks. All tasks share the same convolutional feature layers, and it was shown that the regression network

benefits from sharing features with the detection network. In this work, we also introduce an intermediate representation, body joint labels, for learning intermediate features within a multi-task framework. In contrast to [25], here we focus on 3D pose estimation.

Pre-training has also been shown to be effective in training deep neural networks [26, 27]. [26] empirically shows that the early stages of training with stochastic gradient descent have a large impact on the network's final performance. Pre-training "regularizes" the network by leaving it in a basin of attraction with better generalization. In this work, we propose a strategy to pre-train the regression network using the detection network.

In the literature, deep convolutional neural networks have mainly been used for classification tasks [13, 14, 28], and have achieved state-of-art performances on many vision problems. Importantly, given sufficient data, deep convolutional neural networks can learn good features from randomly initialized weights. In addition, features learned by classification networks can also be used for other tasks – [29] feeds the output of the last convolutional layer of a trained classification neural network into a regression network for predicting bounding boxes for object detection.

3 Deep network for 3D pose estimation

In this paper, we use two strategies to train a deep convolutional neural network for 3D pose estimation. Our framework consists of two types of tasks: 1) a joint point regression task; and 2) joint point detection tasks. The input for both tasks are the bounding box images containing human subjects. The goal of the regression task is to estimate the positions of joint points relative to the root joint position. We define a set of detection tasks, each of which is associated with one joint point and one local window. The aim of each detection task is to classify whether one local window contains the specific joint or not.

3.1 Notation

Let $J_i = (J_{i,x}, J_{i,y}, J_{i,z})$ be the location of the i-th joint in the camera coordinate system. Let P be the articulated skeleton model for the human body, which specifies the parent-child relationship between joints. For example, $P(i)$ specifies the parent joint of the i-th joint. To simplify notation, we let the parent of the root joint to be itself.

3.2 Joint point regression task

The goal of joint point regression is to predict the positions of the joints relative to the root location, $\tilde{J}_i = J_i - J_{root}$. Similar to [12, 9], we assume that the bounding box of the human is provided, and hence it is not necessary to estimate the root location of the person. However, rather than predict the relative joint

positions with respect to root joint, which is a common formulation as in [12, 9, 11], we instead aim to predict the joint positions relative to their *parents* joints,

$$R_i = J_i - J_{P(i)}. \tag{1}$$

This representation can be interpreted as the unnormalized orientation of limbs. There are several reasons why this representation may be advantageous: 1) the variance of R_i is much smaller than $\tilde{J}_i$, which makes it easier to learn – for example, the distance between the wrist and elbow (i.e., $\|R_{wrist}\|$) is constant (for the same person), whereas the distance between the wrist and root position (i.e., $\|\tilde{J}_i\|$) has a wide range of possible values; 2) Since the human body is symmetric, information can be shared between different joints, e.g., the left arm and right arm have the same length. In addition, this representation makes it easier to infer the locations of occluded joints given its opposite part.

The joint point regressor is trained by minimizing the squared difference between the prediction and the ground-truth position,

$$E_r(R_i, \hat{R}_i) = \|R_i - \hat{R}_i\|_2^2 \tag{2}$$

where R_i and $\hat{R}_i$ are the ground-truth and estimated relative position for the i-th joint.

3.3 Joint point detection task

Inspired by [25], we define a set of detection tasks for each joint i and each window l, where the goal is to predict the indicator variable,

$$h_{i,l} = \begin{cases} 1, & \text{if } B_i \text{ is inside window } l, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where B_i is the 2D image location of the i-th joint in the input bounding box. B_i is calculated by projecting J_i into the image and calculating its relative positions with respect to the bounding box. In this work, we do not consider whether the joints are visible or not, i.e., the indicator variables are calculated regardless if the joint is occluded. The reason for doing this is to train the network to learn features for pose estimation even in the presence of occlusions, which might enable the network to predict valid poses when occlusion occur.

As in [25], the detection tasks are trained by minimizing the cross-entropy between the ground-truth label $h_{i,l}$ and the estimated label $\hat{h}_{i,l}$,

$$E_d(\hat{h}_{i,l}, h_{i,l}) = -h_{i,l} \log(\hat{h}_{i,l}) - (1 - h_{i,l}) \log(1 - \hat{h}_{i,l}). \tag{4}$$

The relationship between the regression tasks and detection tasks is illustrated in Figure 1.

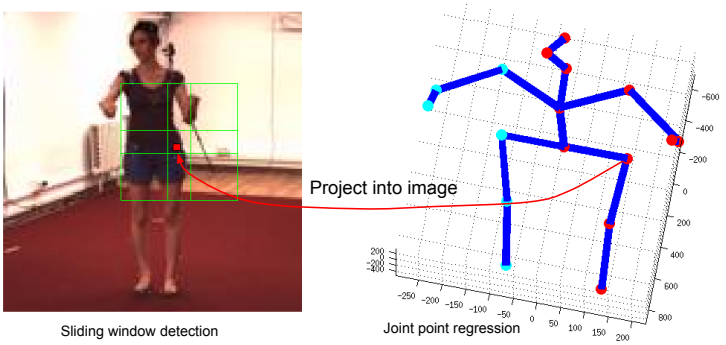

Fig. 1. Illustration of detection tasks and regression task.

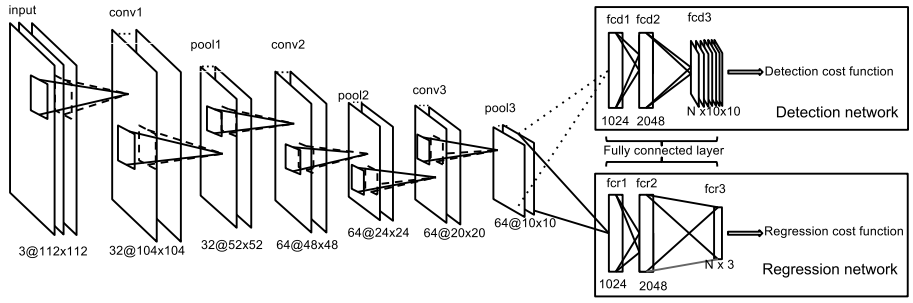

Fig. 2. Network architecture. For network training using multi-task learning, the `pool3` layer is connected to both the `fcd1` and `fcr1` layers. For pre-training with detection tasks, `pool3` is only connected to `fcd1` layer. After pre-training, this connection is removed and `pool3` is connected to `fcr1`. N is the number of joints ($N = 17$ for Human3.6M).

3.4 Network architecture and multi-task training

Our network architecture for 3D pose estimation is displayed in Figure 2. The whole network consists of 9 trainable layers – 3 convolutional layers that are shared by both regression and detection networks, 3 fully connected layers for the regression network, and 3 fully connected layers for the detection network. Rectified linear units (ReLu) [30] are used for `conv1`, `conv2`, and the first two fully connected layers for both regression and detection networks. We use tanh as the activation function for the last regression layer. To make the network robust to pixel intensity, we add a local response normalization layer after `conv2`, which applies the following function to calculate the output values,

$$f(u_{x,y}) = \frac{u_{x,y}}{(1 + \frac{\alpha}{|W_x| \cdot |W_y|} \sum_{x' \in W_x} \sum_{y' \in W_y} u^2_{x',y'})^{\beta}} \tag{5}$$

where $u_{x,y}$ is the value of the previous layer at location (x, y), (W_x, W_y) are the neighborhood of locations (x, y), $|W|$ represents the number of pixels within the neighborhood, and $\{\alpha, \beta\}$ are hyper-parameters.

We train the network within a multi-task learning framework. As in [25], we allow features in the lower layers to be shared between the regression and detection tasks during joint training. During the training, the gradients from both networks will be back-propagated to the same shared feature network, i.e., the network with layers from `conv1` to `pool3`. In this case, the shared network tends to learn features that will benefit both tasks. The global cost function for multi-task training is

$$\Phi_{\mathrm{M}} = \frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{N} E_r(R_i^{(t)}, \hat{R}_i^{(t)}) + \frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{l=1}^{L} E_d(h_{i,l}^{(t)}, \hat{h}_{i,l}^{(t)}), \tag{6}$$

where the superscript t is the index of the training sample, N is the number of joints, and T is the number of training samples.

3.5 Pre-training with the detection task

As an alternative to the multi-task training discussed earlier, another approach is to train the pose regression network using pre-trained weights from the detection network. Firstly we train the detection network alone, i.e., the connections between the `pool3` layer and the `fcr1` layer are blocked. In this stage, we only minimize the second term in (6).

After training the detection tasks, we block the connection between `pool3` and `fcd1` (thus removing the detection task), and reconnect `pool3` to `fcr1` layer. Using this strategy, the training for pose regression is initialized using the feature layer weights (`conv1-conv3`) learned from the detection tasks. Finally, the pose regression is trained using the first term in (6) as the cost function. Note that we do not use the weights of the fully-connected layers of the detectors (`fcd1` and `fcd2`) to initialize fully-connected layers of the regression task (`fcr1` and `fcr2`). The reason is that the target for the detection and regression tasks are quite different, so that the higher-level features used by the detection tasks might not be useful for regression.

3.6 Training details

For both the multi-task and pre-training approaches, we use back-propagation [31] to update the weights during training. In multi-task training, the `pool3` layer forwards its values to both `fcd1` and `fcr1`, and receives the average of the gradients from `fcd1` and `fcr1` when updating the weights. To reduce overfitting, we use "dropout" [32] in `fcr1` and `fcd1`, and set the dropout rate to 0.25. The hyper-parameters for the local response normalization layer are set to $\alpha = 0.0025$ and $\beta = 0.75$. More training details can be found in [13].

4 Experiment

In this section we present experiments using our deep convolutional neural network for monocular 3d pose estimation (DconvMP).

4.1 Human3.6M dataset

The Human3.6M dataset [12] contains around 3.6 million frames of video with 11 human subjects performing 15 actions. The subjects are recorded from 4 different views with RGB cameras, and the joint positions of the subjects were measured by a MoCap system. The calibration parameters are available for the RGB cameras and MoCap system. In our experiments, we use 5 subjects (S1, S5, S6, S7, S8) for training and validation, and 2 subjects (S9, S11) for testing.

Since our method is based on monocular images, we treat the 4 camera views of each pose as separate examples. The ground-truth poses for each camera view are obtained by transforming the joint locations into that camera's coordinate system. Therefore, the input for our method is a single image from one view, and the targets are the joint locations under that view. Test samples from the same frame but different views are evaluated separately, which follows the same setup as [12].

4.2 Data augmentation

We use data augmentation to improve the robustness of the network. After obtaining the bounding box of the human subject (provided by the Human3.6M dataset), we resize the bounding box to 128×128, such that the aspect ratio of the image is maintained. In order to make the network robust to the selection of the bounding box, in each iteration, a sub-window of size 112×112 is randomly selected as the training image (the 2D joint point projections are also adjusted accordingly).

Random pixel noise is also added to each input image during training to make the network robust to small perturbations of pixel values. As in [13] we apply PCA on the RGB channels over the whole training samples. In the training stage, we add random noise to all the pixels in each image,

$$\hat{\mathbf{p}} = \mathbf{p} + [\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}] \cdot [g_1\sqrt{\alpha_1}, g_2\sqrt{\alpha_2}, g_3\sqrt{\alpha_3}]^T, \tag{7}$$

where $[\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}]$ and $[\alpha_1, \alpha_2, \alpha_3]$ are the eigenvectors and eigenvalues of the 3x3 RGB covariance matrix of the training set, and $\{g_c\}_{c=1}^3$ are each Gaussian distributions with zero mean and variance 0.1. In each iteration, all the pixels within one training sample will share the same random values $\{g_c\}_{c=1}^3$.

4.3 Experiment setup

To generate the sliding window for joint point detection, we set the window size to 10×10 and the step size to 10 pixels. Experiments are run on a machine with

two 6-core Intel(R) Xeon(R) CPUs and an Nvidia Tesla K20. It takes 1-2 days to train one action in the Human3.6M dataset (around 100,000 samples after augmentation).

4.4 Evaluation on Human3.6M

Since there is sufficient data in Human3.6M, we train the network for each action separately, which follows the same action-specific protocol as [12]. We selected six representative actions that ranged from easy to difficult (according to previous results in [12]), which include "Walking", "Discussion", "Eating", "Walking Dog", "Greeting" and "Taking Photos". There were 132,744 training samples for "Walking", 158,788 for "Discussion", 109,424 for "Eating", 79,412 for "Walking Dog", 72,436 for "Greeting" and 76,048 for "Taking Photos".

We test networks trained with the heterogeneous multi-task learning framework (denoted as DconvMP-HML), and using pre-training (denoted as DconvMP). Since training detection and regression separately takes more time, we only run DconvMP for 3 actions, namely "Walking", "Eating" and "Taking Photo". We compare against the best performing method, LinKDE, from [12], using the code provided in Human3.6M. We use the code provided by Human3.6M to generate the bounding box for each image sample. Pose predictions are evaluated using the mean per joint position error (MPJPE) [12],

$$\text{MPJPE} = \frac{1}{T}\frac{1}{N}\sum_{t=1}^{T}\sum_{i=1}^{N}\|(J_i^{(t)} - J_{root}^{(t)}) - (\hat{J}_i^{(t)} - \hat{J}_{root}^{(t)})\|_2. \tag{8}$$

The pose prediction results are reported in Table 1. We also show the MPJPE accuracy versus error threshold for all methods in Figure 3. Compared with the baseline method LinKDE [12], our network obtains significant improvement for all the actions evaluated. In our experiments, the DconvMP network achieves roughly the same performance as DconvMP-HML, but takes longer to train.

Figure 4 shows several examples of pose estimation (also see supplemental). We observe that our methods perform better when there are occlusions (e.g., row 3 in Figure 4). In addition, our model performs well at distinguishing between the left and right body parts (e.g., row 2 in Figure 4). In the case where the error is large, our model still outputs a "valid" rough pose.

Table 1. The MPJPE results on Human3.6 dataset. The unit is millimeters. The numbers in parenthesis is the standard deviation of the MPJPE for samples in the testing set.

Action	Walking	Discussion	Eating	Taking Photo	Walking Dog	Greeting
DconvMP	80.09 (23.45)	-	103.31 (37.33)	190.37 (90.64)	-	-
DconvMP-HML	77.60 (23.54)	148.79 (100.49)	104.01 (39.20)	189.08 (93.99)	146.59 (75.38)	127.17 (51.10)
LinKDE (BS) [12]	97.07 (37.14)	183.09 (116.74)	132.50 (72.53)	206.45 (112.61)	177.84 (122.65)	162.27 (88.43)

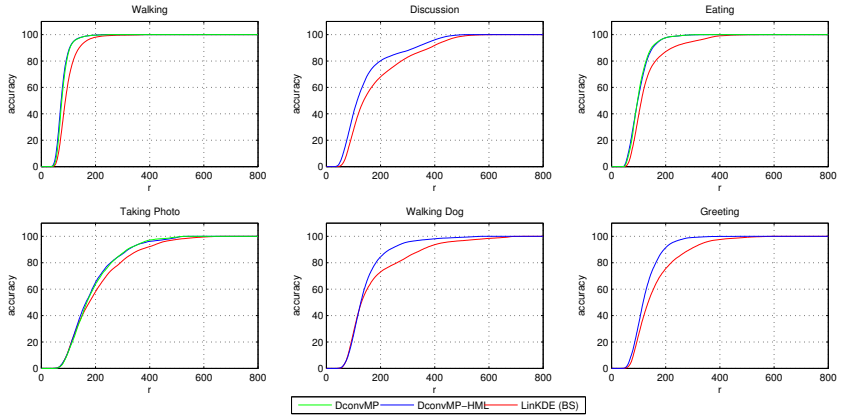

Fig. 3. MPJPE accuracy for error threshold r.

5 Visualization of learned structures

In this section, we explore whether the network has learned the structure of the human skeleton by analyzing the weights in the last layers of the joint regression network. Let F be a $m \times 3 \times N$ matrix of weights between the penultimate and last layers, where m is the dimension of the penultimate layer. We use F_i to denote the $m \times 3$ matrix for predicting the i-th joint, and $F_{i,x}$ to denote the weights for predicting the x-coordinate of the i-th joint (see Figure 5). In the following section, we examine the weights learned on the "Walking" action in the Human3.6M dataset. The weights learned for other actions show similar patterns.

5.1 Pearson correlation between joints

We first examine the correlation between the weights of pairs of joints, i.e., whether two joints use the same high-level features. To this end, we calculate the Pearson correlation between the weights for each pair of joints (i, j),

$$\rho^x_{i,j} = \frac{\mathrm{cov}(F_{i,x}, F_{j,x})}{\sigma(F_{i,x})\sigma(F_{j,x})}, \tag{9}$$

where $\mathrm{cov}(X, Y)$ is the covariance of X and Y, $\sigma(X)$ is the standard derivation of X. The Pearson correlation is calculated for each dimension (x, y, and z).

Figure 6 (top) shows the correlation matrices between the regression weights. Firstly, the correlation matrices show that the learned weights for the left and right hips (also left and right shoulders) are negative correlated. This explains why the network can correctly predict the left hip when only the right side of the body is visible (see row 1 of Figure 4). Also note that the left hip and left shoulder (also right hip and right shoulder) are positively correlated in the x- and z-dimensions, but not the y-dimension; i.e., the left hip and left shoulder share the same internal feature for predicting their x- and z-coordinates. This

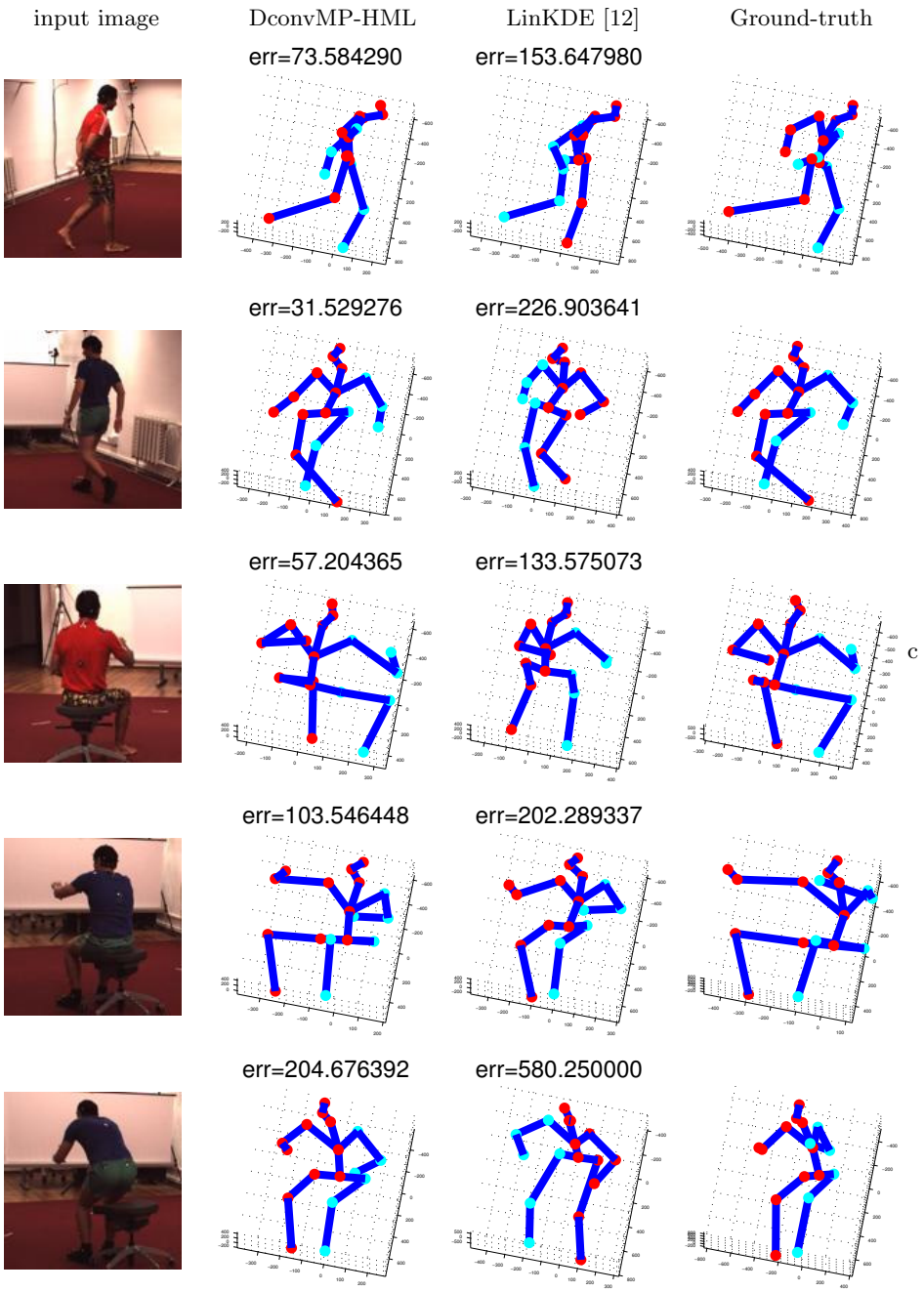

Fig. 4. Examples of pose estimation on Human3.6M. The first two rows are taken from the "Walking" action. The last three rows are taken from "Eating" action. The joints on the right-side are represented by blue balls, while the remaining joints are represented by red balls.

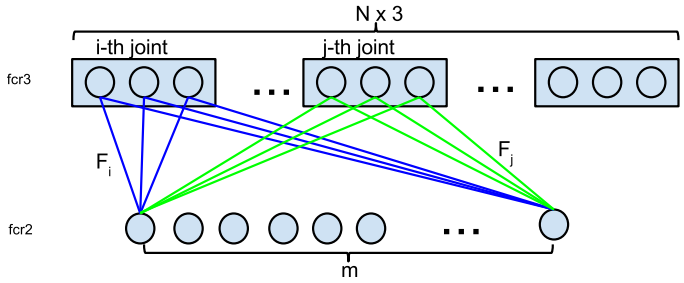

Fig. 5. Illustration of weights F_i and F_j in the final fully-connected layer for regression. For clarity, all the connections are not shown.

suggests that the network has an internal representation for the positions of the left (or right) side of the person, as delineated by the hip and shoulders.

For comparison, Figure 6 (bottom) shows the correlation matrices between the ground-truth relative joint positions. Interestingly, these correlation matrices share similar patterns to those calculated using the regression weights.

5.2 Sparsity measure with LP norm

We next examine the degree to which the internal features are shared in the regression network. To do this, we measure the sparsity (number of zero entries) of weight pairs. [33] showed that the negative l^p norm can be used for measuring the sparsity of a vector. We calculate the "co-sparsity" between the regression weights of two joints (i, j) using

$$S_p(F_{i,x}, F_{j,x}) = -\sum_k \left(\left| \frac{F_{i,x,k}}{\sigma(F_{i,x})} \right|^p + \left| \frac{F_{j,x,k}}{\sigma(F_{j,x})} \right|^p \right)^{\frac{1}{p}}, \tag{10}$$

where $(F_{i,x,k}, F_{j,x,k})$ are the weights for joints i and j corresponding to the same feature k. The S_p measure will be high if entries in the weight pair $(F_{i,x,k}, F_{j,x,k})$ are zero.

The sparsity measures for each pair of joints are shown in Figure 7. We observe that pairing wrists (or ankles) with other parts yields sparser weight pairs, i.e., the prediction for wrists (or ankles) do not share high-level features with other parts. One possible reason is that these extremal joints have the most variance, and thus are the most difficult to predict. As a result, the network has learned specific features for ankles and wrists, which are not shared with other body parts.

6 Conclusion

In this work, we used a deep convolutional neural network for estimating 3D human pose from monocular images. We considered two strategies for training the network: 1) multi-task framework that simultaneously trains the regression

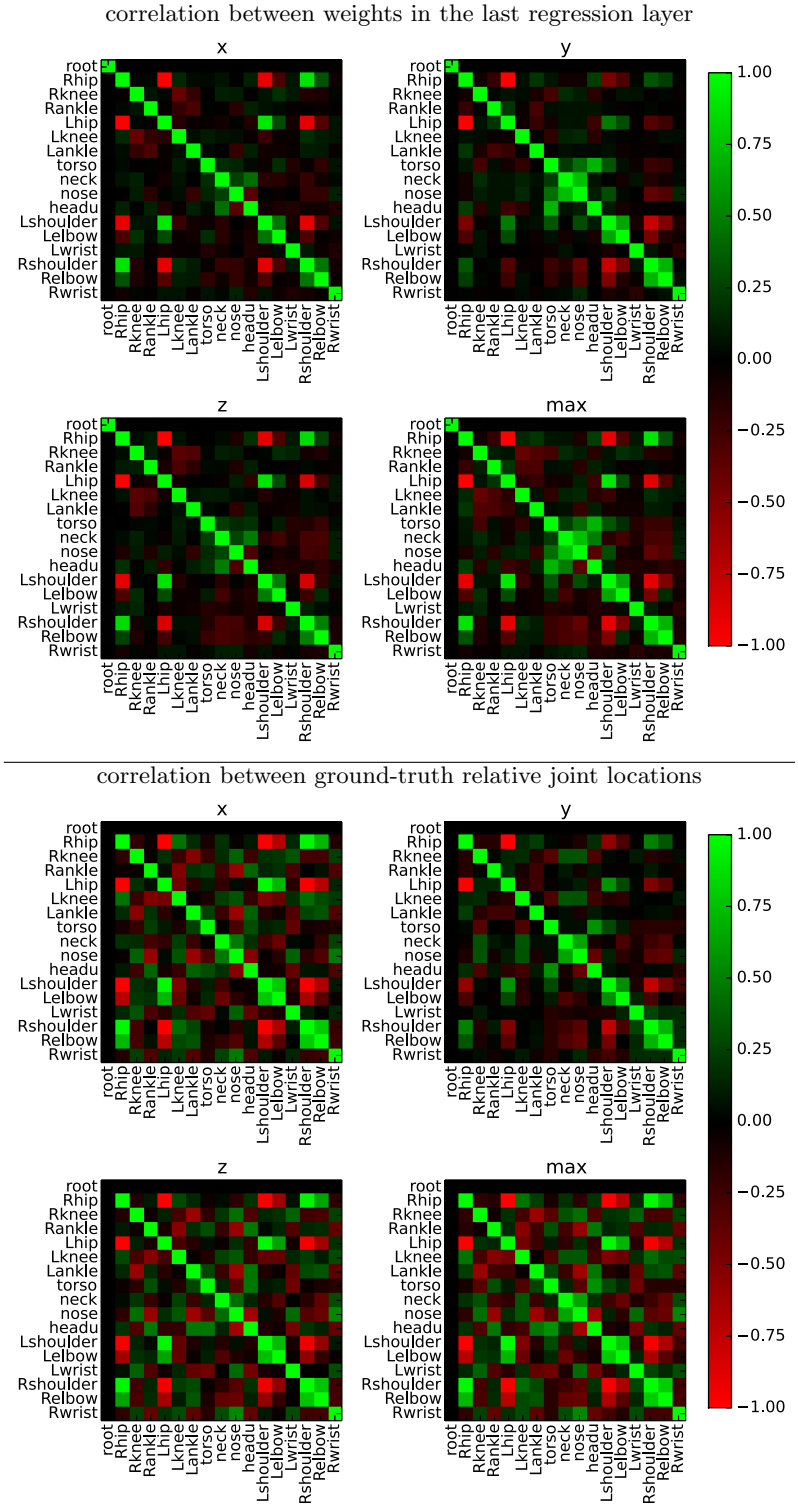

Fig. 6. Pairwise Pearson correlation. Each group of four matrices shows the correlation for the x-, y-, and z-dimensions, as well as the maximum magnitude over all dimensions.

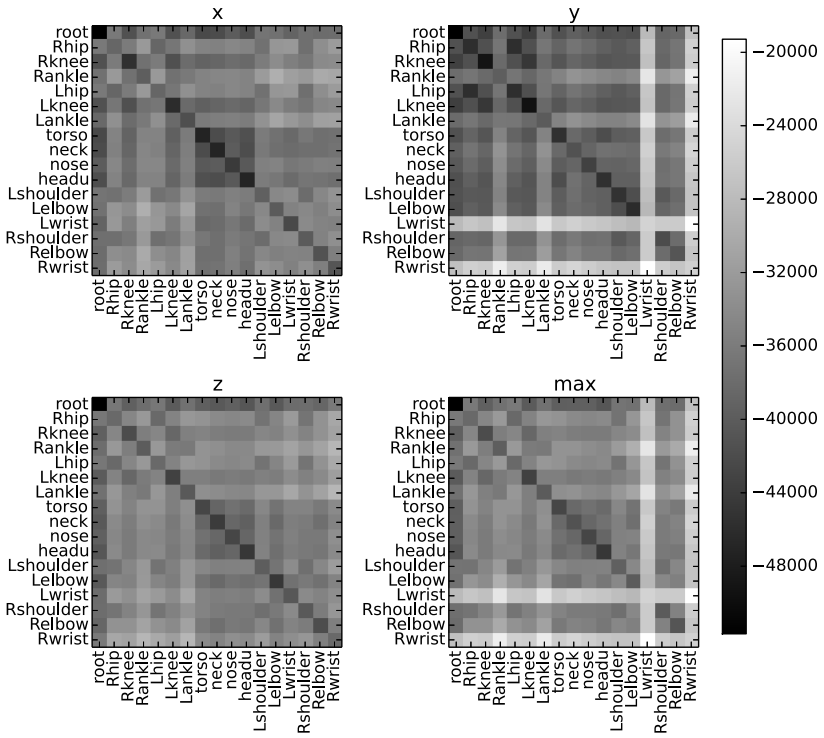

Fig. 7. The LP norm ($p = 0.2$) of the weights for joint pairs.

and detection tasks; 2) and pre-training the regression task with detection tasks. These two strategies yield networks that achieve approximately the same performance, although pre-training has longer running time. When either using pre-training or sharing features, the detection tasks helps to regularize the training of the regression network and guides it to better local minimums. We evaluated our methods on the Human3.6M dataset, and the network achieves significant improvement over baseline methods in [12]. We empirically showed that the deep convolutional network has disentangled dependencies between body parts and learned the correlation between output variables. In this work we have examined how the network encodes structural dependencies in its weights. Future work will explore how such structural dependencies can be induced in the network a priori.

Acknowledgement. This work was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (CityU 123212 and CityU 110513).

References

1. Andriluka, M., Roth, S., Schiele, B.: Monocular 3d pose estimation and tracking by detection. In: CVPR. (2010)
2. Wei, X.K., Chai, J.: Modeling 3d human poses from uncalibrated monocular images. In: ICCV. (2009) 1873–1880
3. Agarwal, A., Triggs, B.: Recovering 3d human pose from monocular images. IEEE Trans. Pattern Anal. Mach. Intell. **28** (2006) 44–58
4. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. CVPR (2011)
5. Felzenszwalb, P.F., Huttenlocher, D.P.: Pictorial structures for object recognition. IJCV (2005) 55–79
6. Eichner, M., Marin-Jimenez, M., Zisserman, A., Ferrari, V.: 2d articulated human pose estimation and retrieval in (almost) unconstrained still images. IJCV (2012)
7. Yang, Y., Ramanan, D.: Articulated pose estimation with flexible mixtures-of-parts. In: CVPR. (2011)
8. Burenius, M., Sullivan, J., Carlsson, S.: 3d pictorial structures for multiple view articulated pose estimation. In: CVPR. (2013) 3618–3625
9. Bo, L., Sminchisescu, C.: Twin gaussian processes for structured prediction. Int. J. Comput. Vision (2010)
10. Dantone, M., Gall, J., Leistner, C., van Gool, L.: Human pose estimation from still images using body parts dependent joint regressors. In: CVPR. (2013)
11. Ionescu, C., Li, F., Sminchisescu, C.: Latent structured models for human pose estimation. In: ICCV. (2011) 2220–2227
12. Ionescu, C., Papava, D., Olaru, V., Sminchisescu, C.: Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE Transactions on Pattern Analysis and Machine Intelligence (2014)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS 25. (2012)
14. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. IEEE TPAMI (2013)
15. Bengio, Y.: Deep learning of representations: Looking forward. CoRR **abs/1305.0445** (2013)
16. Glorot, X., Bordes, A., Bengio, Y.: Domain adaptation for large-scale sentiment classification: A deep learning approach. In: ICML. (2011)
17. Moeslund, T.B., Hilton, A., Krüger, V.: A survey of advances in vision-based human motion capture and analysis. CVIU **104** (2006) 90–126
18. Jain, A., Tompson, J., Andriluka, M., Taylor, G.W., Bregler, C.: Learning human pose estimation features with convolutional networks. In: International Conference on Learning Representations (ICLR). (2014)
19. Girshick, R., Shotton, J., Kohli, P., Criminisi, A., Fitzgibbon, A.: Efficient regression of general-activity human poses from depth images. In: ICCV. (2011) 415–422
20. Sun, Y., Wang, X., Tang, X.: Deep convolutional network cascade for facial point detection. In: CVPR. (2013)
21. Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: IEEE Conf. Computer Vision and Pattern Recognition. (2014)
22. Yuan, C., Niemann, H.: Neural networks for the recognition and pose estimation of 3d objects from a single 2d perspective view. Image Vision Comput. **19** (2001) 585–592

23. Osadchy, M., Cun, Y.L., Miller, M.L.: Synergistic face detection and pose estimation with energy-based models. JMLR **8** (2007) 1197–1215
24. Taylor, G.W., Sigal, L., Fleet, D.J., Hinton, G.E.: Dynamical binary latent variable models for 3d human pose tracking. In: CVPR. (2010) 631–638
25. Li, S., Liu, Z.Q., Chan, A.B.: Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network. In: CVPR: DeepVision workshop. (2014)
26. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. **11** (2010) 625–660
27. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18** (2006) 1527–1554
28. Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J., Ng, A.: Building high-level features using large scale unsupervised learning. In: ICML. (2012)
29. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. CoRR **abs/1312.6229** (2013)
30. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: ICML. (2010)
31. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. (1998) 2278–2324
32. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Improving neural networks by preventing co-adaptation of feature detectors. CoRR (2012)
33. Hurley, N., Rickard, S.: Comparing measures of sparsity. IEEE Trans. Inf. Theor. **55** (2009) 4723–4741