Structured Prediction of 3D Human Pose with Deep Neural Networks

Bugra Tekin*[1]
bugra.tekin@epfl.ch
Isinsu Katircioglu*[1]
isinsu.katircioglu@epfl.ch
Mathieu Salzmann[1]
mathieu.salzmann@epfl.ch
Vincent Lepetit[2]
lepetit@icg.tugraz.at
Pascal Fua[1]
pascal.fua@epfl.ch

[1] CVLab
EPFL,
Lausanne, Switzerland

[2] CVARLab
TU Graz,
Graz, Austria

Abstract

Most recent approaches to monocular 3D pose estimation rely on Deep Learning. They either train a Convolutional Neural Network to directly regress from image to 3D pose, which ignores the dependencies between human joints, or model these dependencies via a max-margin structured learning framework, which involves a high computational cost at inference time.

In this paper, we introduce a Deep Learning regression architecture for structured prediction of 3D human pose from monocular images that relies on an overcomplete auto-encoder to learn a high-dimensional latent pose representation and account for joint dependencies. We demonstrate that our approach outperforms state-of-the-art ones both in terms of structure preservation and prediction accuracy.

1 Introduction

3D human pose can now be estimated reliably by training algorithms to exploit depth data [7, 27] or video sequences [3, 11, 30]. However, estimating such a 3D pose from single ordinary images remains challenging because of the many ambiguities inherent to monocular 3D reconstruction, including occlusions, complex backgrounds, and, more generally, the loss of depth information resulting from the projection from 3D to 2D.

These ambiguities can be mitigated by exploiting the structure of the human pose, that is, the dependencies between the different body joint locations. This has been done by explicitly enforcing physical constraints at test time [25, 29] and by data-driven priors over the pose space [6, 28, 33]. Recently, dependencies have been modeled within a Deep Learning framework using a max-margin formalism [20], which resulted in state-of-the-art prediction accuracy. While effective, these methods suffer from the fact that they require solving a computationally expensive optimization problem to estimate the 3D pose.

* indicates equal contribution.

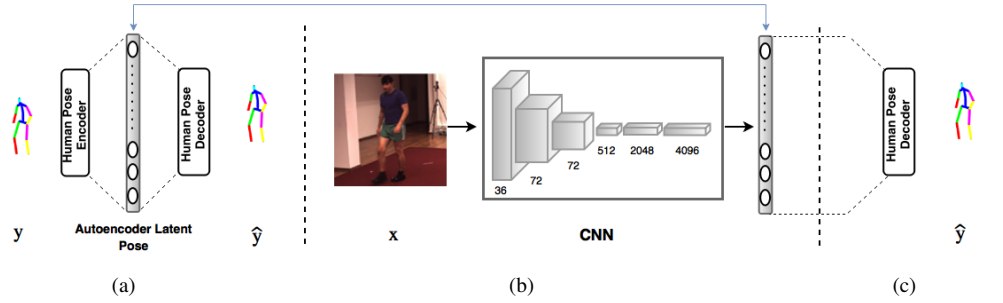


Figure 1: Our architecture for the structured prediction of the 3D human pose. **(a)** An auto-encoder whose hidden layers have a larger dimension than both its input and output layers is pretrained. In practice we use either this one or more sophisticated versions that are described in more detail in Section 3.1 **(b)** A CNN is mapped into the latent representation learned by the auto-encoder. **(c)** the latent representation is mapped back to the original pose space using the decoder.

By contrast, regression-based methods, such as [19], directly and efficiently predict the 3D pose given the input image. While this often comes at the cost of ignoring the underlying structure, several methods have been proposed to account for it [14, 26]. In [14], this was achieved by making use of Kernel Dependency Estimation (KDE) [5, 36], which maps both input and output to high-dimensional Hilbert spaces and learns a mapping between these spaces. Because this approach relies on handcrafted features and does not exploit the power of Deep Learning, it somewhat under-performs more recent CNN-based techniques [19, 20].

In this paper, we demonstrate that we can account for the human pose structure within a deep learning framework by first training an overcomplete auto-encoder that projects body joint positions to a high dimensional space represented by its middle layer, as depicted by Fig. 1(a). We then learn a CNN-based mapping from the input image to this high-dimensional pose representation as shown in Fig. 1(b). This is inspired by KDE in that it can be understood as replacing kernels by the auto-encoder layers to predict the pose parameters in a high dimensional space that encodes complex dependencies between different body parts. As a result, it enforces implicit constraints on the human pose, preserves the human body statistics, and improves prediction accuracy, as will be demonstrated by our experiments. Finally, as illustrated in Fig. 1(c), we connect the decoding layers of the auto-encoder to this network, and fine-tune the whole model for pose estimation.

In short, our contribution is to show that combining traditional CNNs for supervised learning with auto-encoders for structured learning preserves the power of CNNs while also accounting for dependencies, resulting in increased performance. In the remainder of the paper, we first briefly discuss earlier approaches. We then present our structured prediction approach in more detail and finally demonstrate that it outperforms state-of-the-art methods on the Human3.6m dataset.

2 Related Work

Following recent trends in Computer Vision, human pose estimation is now usually formulated within a Deep Learning framework. The switch away from earlier representations

started with 2D pose estimation by learning a regressor from an input image to either directly the pose vectors [32] or the heatmaps encoding 2D joint locations [15, 22, 31]. Recently, this trend has extended to 3D pose estimation [19], where the problem is typically formulated in terms of continuous 3D joint locations, since discretizing the 3D space is more challenging than in the case of 2D.

Another important difference between 2D and 3D pose estimation comes from the additional ambiguities in the latter one due to the fact that the input only shows a projection of the output. To overcome these ambiguities, recent algorithms have attempted to encode the dependencies between the different joints within Deep Learning approaches, thus effectively achieving structured prediction. In particular, [10] uses auto-encoders to learn a shared representation for 2D silhouettes and 3D poses. This approach, however, relies on accurate foreground masks and exploits handcrafted features, which mitigate the benefits of Deep Learning. In the context of hand pose estimation, [21] introduces a bottleneck, low-dimensional layer that aims at accounting for joint dependencies. This layer, however, is obtained directly via PCA, which limits the kind of dependencies it can model.

To the best of our knowledge, the work of [20] constitutes the most effective approach to encoding dependencies within a Deep Learning framework for 3D human pose estimation. This approach extends the structured SVM model to the Deep Learning setting by learning a similarity score between feature embeddings of the input image and the 3D pose. This process, however, comes at a high computational cost at test time, since, given an input image, the algorithm needs to search for the highest-scoring pose. Furthermore, the final results are obtained by averaging over multiple high-scoring ground-truth training poses, which might not generalize well to unseen data since the prediction can thus only be in the convex hull of the ground-truth training poses. By contrast, we draw inspiration from the KDE-based approaches [12, 14], that map both image and 3D pose to high-dimensional Hilbert spaces and learn a mapping between these spaces. Here, however, we show how to do this in a Deep Learning context with CNNs and auto-encoders. The benefits are twofold: We can leverage the power of learned features that have proven more effective than handcrafted ones such as HOG [1], and our framework relies on a direct and efficient regression between the two spaces, thus avoiding the computational burden of the state-of-the-art approach of [20].

Using auto-encoders for unsupervised feature learning has proven effective in several recognition tasks [16, 18, 35]. In particular, denoising auto-encoders [34] that aim at reconstructing the perfect data from a corrupted version of it have demonstrated good generalization ability. Similarly, contractive auto-encoders have been shown to produce intermediate representations that are robust to small variations of the input data [24]. All these methods, however, rely on auto-encoders to learn features for recognition tasks. By contrast, here, we exploit auto-encoders to model the output structure for regression purposes.

3 Method

In this work, we aim at directly regressing from an input image x to a 3D human pose. As in [9, 13, 19], we represent the human pose in terms of the 3D locations $y \in \mathbb{R}^{3J}$ of J body joints relative to a root joint. An alternative would have been to predict the joint angles and limb lengths, however this is a less homogeneous representation and is therefore rarely used for regression.

As discussed above, a straightforward approach to creating a regressor is to train a conventional CNN such as the one used in [19]. However, this fails to encode dependencies

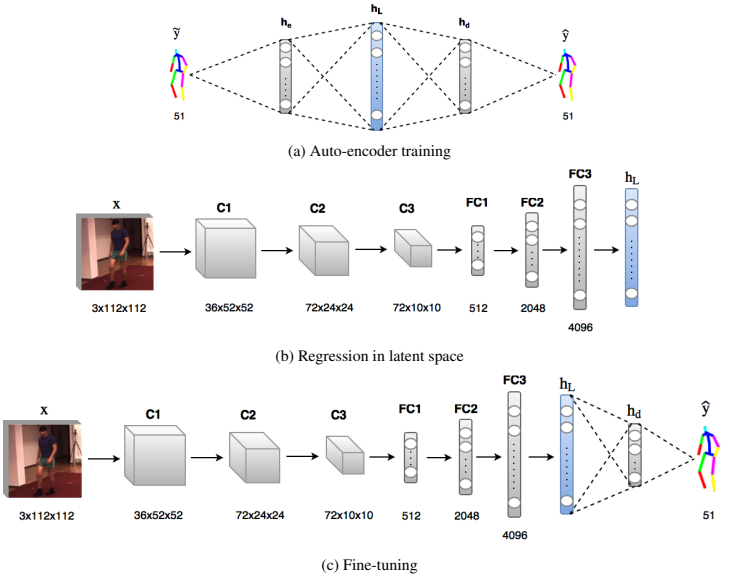


Figure 2: Our approach. **(a)** We train a stacked denoising auto-encoder that learns and enforces implicit constraints about human body in its latent middle layer h_L. **(b)** Our CNN architecture maps the image to the latent representation h_L learned by the auto-encoder. **(c)** We stack the decoding layers of the auto-encoder on top of the CNN for reprojection from the latent space to the original pose space and fine-tune the entire network by updating the parameters of all layers.

between joint locations. In [20], this limitation was overcome by introducing a substantially more complex, deep architecture for maximum-margin structured learning. Here, we encode dependencies in a simpler, more efficient, and ultimately more accurate way by learning a mapping between the output of a conventional CNN and a latent representation obtained using an overcomplete auto-encoder, as illustrated in Fig. 2. The auto-encoder is pre-trained on human poses and comprises a hidden layer of *higher dimension* than its input and output. In effect, this hidden layer and the CNN-based representation of the image play the same role as the kernel embeddings in KDE-based approaches [5, 12, 14], thus allowing us to account for structure within a direct regression framework. Once the mapping between these two high-dimensional embeddings is learned, we further fine-tune the whole network for the final pose estimation task, as depicted at the bottom of Fig. 2.

In the remainder of this section, we describe the different stages of our approach.

3.1 Using Auto-Encoders to Learn Structured Latent Representations

We encode the dependencies between human joints by learning a mapping of 3D human pose to a high-dimensional latent space. To this end, we use a denoising auto-encoder that can have one or more hidden layers.

Following standard practice [35], given a training set of pose vectors $\{y_i\}$, we add isotropic Gaussian noise to create noisy versions $\{\tilde{y}_i\}$ of these vectors. We then train our auto-encoder to take as input a noisy $\tilde{y}_i$ and return a denoised y_i as output. The corresponding reconstruction function $f_{ae}(\cdot)$ must satisfy

$$\hat{y} = f_{ae}(\tilde{y}, \theta_{ae}) \,, \tag{1}$$

where $\hat{y}$ is the reconstruction and $\theta_{ae} = (W_{enc,j}, b_{enc,j}, W_{dec,j}, b_{dec,j})_{j=1}^{L}$ contains the model parameters, that is, the weights and biases for L encoding and decoding layers. We take the middle layer to be our latent pose representation and denote it by h_L. We use ReLU as the activation function of the encoding layer. This favors a sparse hidden representation [8], which has been shown to be effective at modeling a wide range of human poses [2, 23]. A linear activation function is used at the decoding layer of the auto-encoder to reproject to both negative and positive joint coordinates. To keep the number of parameters small and reduce overfitting, we use tied weights for the encoder and the decoder, that is, $W_{dec,j} = W_{enc,j}^{T}$.

To learn the parameters θ_{ae}, we rely on the square loss between the reconstruction, $\hat{y}$, and the original input, y, over the N training examples. To increase robustness to small pose changes, we regularize the cost function by adding the squared Frobenius norm of the Jacobian of the hidden mapping $g(\cdot)$, that is, $J(\tilde{y}) = \frac{\partial g}{\partial \tilde{y}}(\tilde{y})$ where $g(\cdot)$ is the encoding function that maps the input $\tilde{y}$ to the middle hidden layer, h_L. Training can thus be expressed as finding

$$\theta_{ae}^{*} = \underset{\theta_{ae}}{\operatorname{argmin}} \sum_{i}^{N} ||y_i - f_{ae}(\tilde{y}_i, \theta_{ae})||_2^2 + \lambda \|J(\tilde{y}_i)\|_F^2 \,, \tag{2}$$

where λ is the regularization weight. Unlike when using KDE, we do not need to solve a complex pre-image problem to go from the latent pose representation to the pose itself. This mapping, which corresponds to the decoding part of our auto-encoder, is learned directly from data.

3.2 Regression in Latent Space

Once the auto-encoder is trained, we aim to learn a mapping between the input image and the latent representation of the human pose. To this end, and as shown in Fig. 2(b), we make use of a CNN to regress the image to a high-dimensional representation, which is itself mapped to the latent pose representation.

More specifically, let θ_{cnn} be the parameters of the CNN, including the mapping to the latent pose representation. Given an input image x, we consider the square loss function between the representation predicted by the CNN, $f_{cnn}(x, \theta_{cnn})$, and the one that was previously learned by the auto-encoder, h_L. Given our N training samples, learning amounts to finding

$$\theta_{cnn}^{*} = \underset{\theta_{cnn}}{\operatorname{argmin}} \sum_{i}^{N} ||f_{cnn}(x_i, \theta_{cnn}) - h_{L,i}||_2^2 \,. \tag{3}$$

In practice, as shown in Fig. 2(b), we rely on a standard CNN architecture similar to the one of [19, 32]. It comprises three convolutional layers—C1, C2 and C3—each followed by a pooling layer—P1, P2, and P3. In our implementation, the input volume is a three channel image of size 128×128. $P3$ is directly connected to a cascade of fully-connected layers—FC1, FC2 and FC3—that produces a 4096-dimensional image representation, which is then mapped linearly to the latent pose embedding. Except for this last linear layer, each layer uses a ReLU activation function.

As in [19], prior to training our CNN, we first initialize the convolutional layers using a network trained for the detection of body joints in 2D. We then replace the fully-connected layers of the detection network with those of the regressor to further train for the pose estimation task.

3.3 Fine-Tuning the Whole Network

Finally, as shown in Fig. 2(c), we append the decoding layers of the auto-encoder to the CNN discussed above, which reprojects the latent pose estimates to the original pose space. We then fine-tune the resulting complete network for the task of human pose estimation. We take the cost function to be the squared difference between the predicted and ground-truth 3D poses, which yields the optimization problem

$$\theta_{ft}^{*} = \underset{\theta_{ft}}{\operatorname{argmin}} \sum_{i}^{N} ||f_{ft}(x_i, \theta_{ft}) - y_i||_2^2 \,, \tag{4}$$

where θ_{ft} are the complete set of model parameters and $f_{ft}(\cdot)$ is the mapping function.

4 Results

In this section, we first describe the large-scale dataset we tested our approach on. We then give implementation details and describe the evaluation protocol. Finally, we compare our results against those of the state-of-the-art methods.

4.1 Dataset

We evaluate our method on the Human3.6m dataset [14], which comprises 3.6 million image frames with their corresponding 2D and 3D poses. The subjects perform complex motion scenarios based on typical human activities such as discussion, eating, greeting and walking. The videos were captured from 4 different camera viewpoints. Following the standard procedure of [19], we collect the input images by extracting a square region around the subject using the bounding box present in the dataset and resize it to 128×128. The output pose is a vector of 17 3D joint coordinates.

4.2 Implementation Details

We trained our auto-encoder using a greedy layer-wise training scheme followed by fine-tuning as in [9, 35]. We set the regularization weight of Eq. 2 to $\lambda = 0.1$. We experimented with single-layer auto-encoders, as well as with 2-layer ones. The sizes of the hidden layers were set to 2000 and 300-300 for the 1-layer and 2-layer cases, respectively, meaning that $h_L = 2000$ or $h_L = 300$. We corrupted the input pose with zero-mean Gaussian noise with standard deviation of 40 for 1-layer and 40-20 for 2-layer auto-encoders. In all cases, we used the ADAM optimization procedure [17] with a learning rate of 0.001 and a batch size of 128.

The number and individual sizes of the layers of our CNNs are given in Fig. 2. The filter sizes for the convolutional layers are consecutively 9×9, 5×5 and 5×5. Each convolutional layer is followed by a 2×2 max-pooling layer. The activation function is the ReLU in all the layers except for the last one that uses linear activation. As for the auto-encoders, we used ADAM [17] with a learning rate of 0.001 and a batch size of 128. To prevent overfitting, we applied dropout with a probability of 0.5 after each fully-connected layer and augment the data by randomly cropping 112×112 patches from the 128×128 input images.

4.3 Evaluation Protocol

For a fair comparison, we used the same data partition protocol as in earlier work [19, 20] for training and test splits. The data from 5 subjects (S1,S5,S6,S7,S8) was used for training and

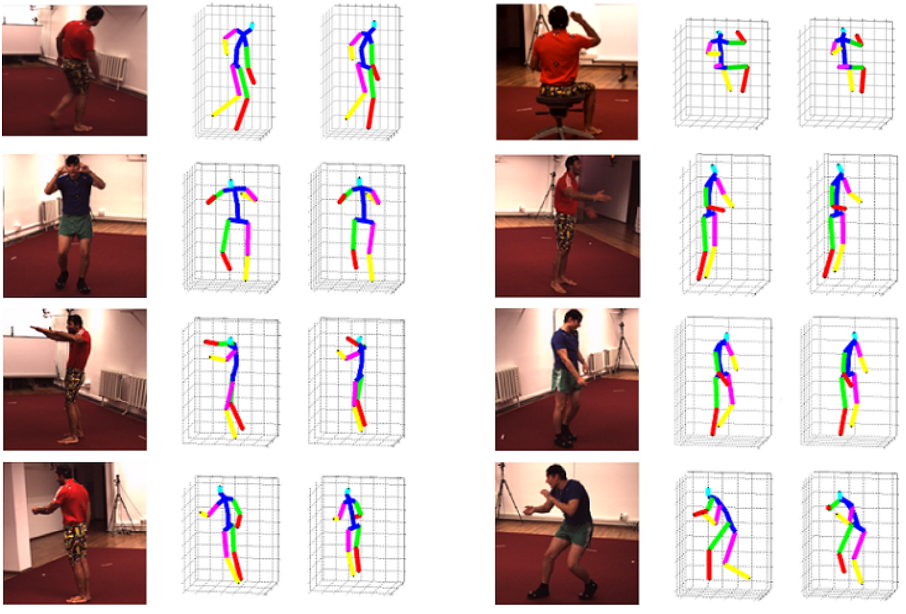


Figure 3: 3D poses for the *Walking, Eating, Taking Photo, Greeting, Discussion* and *Walking Dog* actions of the Human3.6m database. In each case, the first skeleton depicts the ground-truth pose and the second one the pose we recover. Note that our method can recover the 3D pose in these challenging scenarios. Bottom row shows examples of failure cases where the prediction slightly deviates from the ground-truth due to self-occlusion and complex poses that are underrepresented in the training set. Best viewed in color.

Model	Discussion	Eating	Greeting	Taking Photo	Walking	Walking Dog
LinKDE([14]	183.09	132.50	162.27	206.45	97.07	177.84
DconvMP-HML [19]	148.79	104.01	127.17	189.08	77.60	146.59
StructNet-Max [20]	149.09	109.93	136.90	179.92	83.64	147.24
StructNet-Avg [20]	134.13	97.37	122.33	166.15	68.51	132.51
OURS	**129.06**	**91.43**	**121.68**	**162.17**	**65.75**	**130.53**

Table 1: Average Euclidean distance in mm between the ground-truth 3D joint locations and those predicted by competing methods [14, 19, 20] and ours.

the data from 2 different subjects (S9,S11) was used for testing. We evaluate the accuracy of 3D human pose estimation in terms of average Euclidean distance between the predicted and ground-truth 3D joint positions as in [19, 20]. The accuracy numbers are reported in milimeters for all actions on which the authors of [19, 20] provided results. Training and testing were carried out monocularly in all camera views for each separate action.

4.4 Experimental Results

Fig. 3 depicts selected pose estimation results on Human3.6m. In Table 1, we report our results on this dataset along with those of three state-of-the-art approaches: KDE regression from HOG features to 3D poses [14], jointly training a body part detector and a 3D pose regressor network [19], and using a maximum-margin formalism within a Deep Learning framework for structured prediction [20]. For the latter, the estimation is taken to be either the highest-scoring pose or the average of the 500 highest-scoring training poses. Our method consistently outperforms all the baselines.

Model	Discussion	Eating	Greeting	Taking Photo	Walking	Walking Dog
CNN-Direct	135.36	105.98	133.35	177.62	77.73	153.02
OURS, 1 layer no FT	134.02	96.01	127.58	158.73	68.55	146.28
OURS, 2 layer no FT	129.67	98.57	124.80	162.69	73.47	146.46
OURS, 1 layer with FT	130.07	94.08	121.96	**158.51**	65.83	135.35
OURS, 2 layer with FT	**129.06**	**91.43**	**121.68**	162.17	**65.75**	**130.53**

(a)

Model	Taking Photo
CNN-Direct	177.62
CNN-ExtraFC[2000]	179.29
CNN-PCA[30]	170.74
CNN-PCA[40]	167.62
CNN-PCA[51]	182.64
OURS[40]	165.11
OURS[2000]	**158.51**

(b)

Table 2: Average Euclidean distance in mm between the ground-truth 3D joint locations and those computed **(a)** using either no auto-encoder at all (CNN) or 1-layer and 2-layer encoders (OURS), with or without fine-tuning (FT), **(b)** by replacing the auto-encoder by either an additional fully-connected layer (*CNN-ExtraFC*) or a PCA layer (*CNN-PCA*). The bracketed numbers denote the various dimensions of the additional layer we tested. Our approach again yields the most accurate predictions.

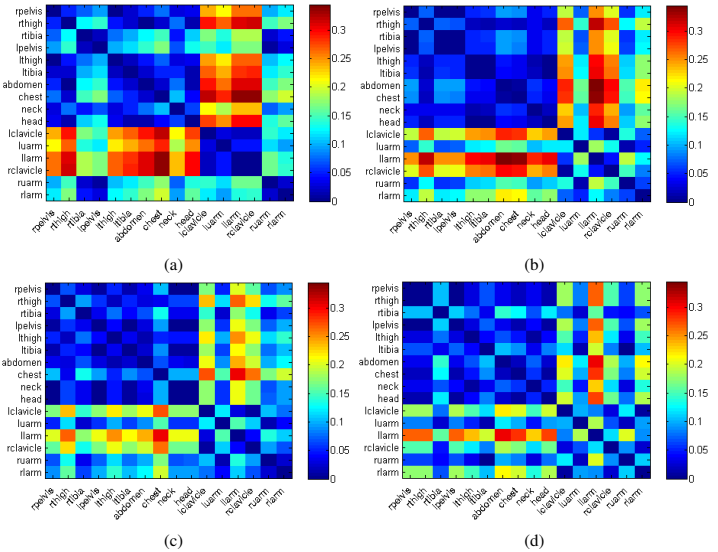
The results reported in Table 1 were obtained using a two layer auto-encoder. However, as discussed in Section 3.1 our formalism applies to auto-encoders of any depth. Therefore, in Table 2 (a), we report results obtained using a single layer one as well as by turning off the final fine-tuning of Section 3.3. For completeness, we also report results obtained by using a CNN similar to the one of Fig. 2(b) to regress directly to a 51-dimensional 3D pose vector *without* using an auto-encoder at all. We will refer to it as *CNN-Direct*. We found that both kinds of auto-encoders perform similarly and better than *CNN-Direct*, especially for actions such as *Taking Photo* and *Walking Dog* that involve interactions with the environment and are thus physically more constrained. This confirms that the power of our method comes from auto-encoding. Furthermore, as expected, fine-tuning consistently improves the results.

During fine-tuning, our complete network has more fully-connected layers than *CNN-Direct*. One could therefore argue that the additional layers are the reason why our approach outperforms it. To disprove this, we evaluated the baseline, *CNN-ExtraFC*, in which we simply add one more fully-connected layer. We also evaluated another baseline, *CNN-PCA*, in which we replace our auto-encoder latent representation by a PCA-based one. In Table 2(b), we show that our approach significantly outperforms these two baselines on the *Taking Photo* action. This suggests that our overcomplete auto-encoder yields a representation that is more discriminative than other latent ones. Among the different PCA configurations, the one with 40 dimensions performs the best. However, training an auto-encoder with 40 dimensions outperforms it.

Following [12], we show in Fig. 4 the differences between the ground-truth limb ratios and the limb ratios obtained from predictions based on KDE, *CNN-Direct* and our approach. These results evidence that our predictions better preserve these limb ratios, and thus better model the dependencies between joints.

4.5 Parameter Choices

In Table 3, we compare the results of different auto-encoder configurations in terms of number of layers and channels per layer on the *Greeting* action. Similarly to what we did in Table 2(b), the bracketed numbers denote the dimension of the auto-encoder's hidden layers. We obtained the best result for 1 layer with 2000 channels or 2 layers with 300-300 channels. These values are those we used for all the experiments described above. They were chosen for a single action and used unchanged for all others, thus demonstrating the versatility of our approach.



Model	Lower Body	Upper Body	Full Body
KDE [14]	1.02	7.18	16.43
CNN	0.57	6.86	14.97
OURS no FT	0.62	5.30	11.99
OURS with FT	0.77	5.43	11.90

(e)

Figure 4: Matrix of differences between estimated log of limb length ratios and those computed from ground-truth poses. The rows and columns correspond to individual limbs. For each cell, the ratios are computed by dividing the limb length in the horizontal axis by the one in the vertical axis as in [12] for **(a)** KDE [14], **(b)** CNN-Direct as in Table 2, and **(c,d)** our method without and with fine-tuning. An ideal result would be one in which all cells are blue, meaning the limb length ratios are perfectly preserved. Best viewed in color. **(e)** Sum of the log of limb length ratio errors for different parts of the human body. All methods perform well on the lower body. However, ours outperforms the others on the upper body and when considering all ratios in the full body.

Layer Configuration	Greeting
[40]	129.49
[500]	123.95
[1000]	121.96
[2000]	**121.96**
[3000]	123.49
[250-250]	125.61
[300-300]	**121.68**
[250-500]	128.98
[500-1000]	126.52
[200-200-200]	126.78
[500-500-500]	127.73

Table 3: Average Euclidean distance in mm between the ground-truth 3D joint locations and the ones predicted by our approach trained using auto-encoders in various configurations, with different number of layers and number of channels per layer as indicated by the bracketed numbers. This validation was performed on the *Greeting* action and the optimal values used for all other actions.

5 Conclusion

We have introduced a novel Deep Learning regression architecture for structured prediction of 3D human pose from a monocular image. We have shown that our approach to combining auto-encoders with CNNs accounts for the dependencies between the human body parts efficiently and yields better prediction accuracy than state-of-the-art methods. Since our framework is generic, in future work, we intend to apply it to other structured prediction problems, such as deformable surface reconstruction.

Acknowledgments. This work was supported in part by the EUROSTARS Project CLASS.

References

[1] A. Agarwal and B. Triggs. 3D Human Pose from Silhouettes by Relevance Vector Regression. In *CVPR*, 2004.

[2] I. Akhter and M. J. Black. Pose-Conditioned Joint Angle Limits for 3D Human Pose Reconstruction. In *CVPR*, 2015.

[3] M. Andriluka, S. Roth, and B. Schiele. Monocular 3D Pose Estimation and Tracking by Detection. In *CVPR*, 2010.

[4] L. Bo and C. Sminchisescu. Twin Gaussian Processes for Structured Prediction. *IJCV*, 2010.

[5] C. Cortes, M. Mohri, and J. Weston. A General Regression Technique for Learning Transductions. In *ICML*, 2005.

[6] J. Gall, B. Rosenhahn, T. Brox, and H.-P. Seidel. Optimization and Filtering for Human Motion Capture. *IJCV*, 2010.

[7] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient Regression of General-Activity Human Poses from Depth Images. In *ICCV*, 2011.

[8] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *AISTATS*, 2011.

[9] G. Hinton and R. Salakutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 2006.

[10] C. Hong, J. Yu, J. Wan, D. Tao, and M. Wang. Multimodal Deep Autoencoder for Human Pose Recovery. *TIP*, 2014.

[11] C. H. Huang, E. Boyer, N. Navab, and S. Ilic. Human Shape and Pose Tracking Using Keyframes. In *CVPR*, 2014.

[12] C. Ionescu, F. Li, and C. Sminchisescu. Latent Structured Models for Human Pose Estimation. In *ICCV*, 2011.

[13] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *PAMI*, 2014.

[14] C. Ionescu, I. Papava, V. Olaru, and C. Sminchisescu. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *PAMI*, 2014.

[15] A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler. Learning Human Pose Estimation Features with Convolutional Networks. In *ICLR*, 2014.

[16] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2014.

[17] D.P. Kingma and J. Ba. Adam: A Method for Stochastic Optimisation. In *ICLR*, 2015.

[18] K. Konda, R. Memisevic, and D. Krueger. Zero-bias autoencoders and the benefits of co-adapting features. In *ICLR*, 2015.

[19] S. Li and A.B. Chan. 3D Human Pose Estimation from Monocular Images with Deep Convolutional Neural Network. In *ACCV*, 2014.

[20] S. Li, W. Zhang, and A. B. Chan. Maximum-Margin Structured Learning with Deep Networks for 3D Human Pose Estimation. In *ICCV*, 2015.

[21] M. Oberweger, P. Wohlhart, and V. Lepetit. Hands Deep in Deep Learning for Hand Pose Estimation. *arXiv Preprint*, 2015.

[22] T. Pfister, J. Charles, and A. Zisserman. Flowing ConvNets for Human Pose Estimation in Videos. In *ICCV*, 2015.

[23] V. Ramakrishna, T. Kanade, and Y. Sheikh. Reconstructing 3D Human Pose from 2D Image Landmarks. In *ECCV*, 2012.

[24] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive Auto-encoders: Explicit invariance during feature extraction. In *ICML*, 2011.

[25] M. Salzmann and R. Urtasun. Combining Discriminative and Generative Methods for 3D Deformable Surface and Articulated Pose Reconstruction. In *CVPR*, June 2010.

[26] M. Salzmann and R. Urtasun. Implicitly Constrained Gaussian Process Regression for Monocular Non-Rigid Pose Estimation. In *NIPS*, December 2010.

[27] J. Shotton, A. Fitzgibbon, M. Cook, and A. Blake. Real-Time Human Pose Recognition in Parts from a Single Depth Image. In *CVPR*, 2011.

[28] H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic Tracking of 3D Human Figures Using 2D Image Motion. In *ECCV*, 2000.

[29] C. Sminchisescu and B. Triggs. Kinematic Jump Processes for Monocular 3D Human Tracking. In *CVPR*, 2003.

[30] B. Tekin, A. Rozantsev, V. Lepetit, and P. Fua. Direct Prediction of 3D Body Poses from Motion Compensated Sequences. In *CVPR*, 2016.

[31] 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.

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

[33] R. Urtasun, D. Fleet, A. Hertzman, and P. Fua. Priors for People Tracking from Small Training Sets. In *ICCV*, 2005.

[34] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

[35] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *JMLR*, 2010.

[36] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel Dependency Estimation. In *NIPS*, 2002.