IJCV manuscript No.
(will be inserted by the editor)

Learning Latent Representations of 3D Human Pose with Deep Neural Networks

Isinsu Katircioglu*, Bugra Tekin*, Mathieu Salzmann, Vincent Lepetit, Pascal Fua

Received: February 1, 2017

Abstract Most recent approaches to monocular 3D pose estimation rely on Deep Learning. They either train a Convolutional Neural Network to directly regress from an image to a 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 or 2D joint location heatmaps that relies on an overcomplete autoencoder to learn a high-dimensional latent pose representation and accounts for joint dependencies. We further propose an efficient Long Short-Term Memory (LSTM) network to enforce temporal consistency on 3D pose predictions. We demonstrate that our approach achieves state-of-the-art performance both in terms of structure preservation and prediction accuracy on standard 3D human pose estimation benchmarks.

Keywords 3D Human Pose Estimation · Structured Prediction · Deep Learning

(*) I. Katircioglu and B. Tekin contributed equally as co-first authors.

· I. Katircioglu · B. Tekin · M. Salzmann · P. Fua
Computer Vision Laboratory (CVLab)
École Polytechnique Fédérale de Lausanne (EPFL)
1015, Lausanne, Switzerland
e-mail: firstname.lastname@epfl.ch

V. Lepetit
LaBRI
University of Bordeaux
33405 Talence, France
e-mail: vincent.lepetit@u-bordeaux.fr

1 Introduction

In spite of much recent progress, estimating 3D human pose from a single ordinary image remains challenging because of the many ambiguities inherent to monocular 3D reconstruction. They include occlusions, complex backgrounds, and, more generally, the loss of depth information resulting from the projection from 3D to 2D.

Recent regression-based methods can directly and efficiently predict the 3D pose given the input image [32] or images [55] but often ignore the underlying body structure and resulting joint dependencies, which makes them vulnerable to ambiguities. Several methods have recently been proposed to account for these dependencies [49,23,33]. In particular, by leveraging the power of Deep Learning, the method of [33] achieves high accuracy. However, it involves a computationally expensive search procedure to estimate the 3D pose.

Since pose estimation is much better-posed in 2D than in 3D, an alternative way to handle ambiguities is to use discriminative 2D pose regressors [6,8,11,15,24,38,42,44,58,61,63] to extract the 2D pose and then infer a 3D one from it [4,13,64,66]. This however also involves fitting a 3D model in a separate optimization step, and is thus more expensive than direct regression.

In this paper, we demonstrate that we can account for the human pose structure within a deep learning regression framework. To this end, we propose to first train an overcomplete autoencoder 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 image to this high-dimensional pose representation as shown in Fig. 1(b). Finally, as illustrated in Fig. 1(c), we connect the decoding layers of the autoencoder to the CNN, and fine-tune the whole model for pose estimation. This procedure is inspired by Kernel Dependency Estima-

tion (KDE) in that it can be understood as replacing the high-dimensional feature maps in kernel space by autoencoder layers that represent the pose in a high-dimensional space encoding complex dependencies between the different body parts. However, our approach has the advantage over KDE of directly providing us with a mapping back to the pose space, thus avoiding the need for a computationally expensive optimization at test time. Altogether, and as will be demonstrated by our experiments, our framework enforces implicit constraints on the human pose, preserves the human body statistics, and improves prediction accuracy.

With the growing availability of large training datasets, 2D pose estimation algorithms have achieved tremendous success [38,44,61] by relying on Deep Learning. They exploit the fact that finding 2D joint locations in a color image is easier than direct 3D pose prediction, which is fraught with depth ambiguities. To leverage the well-posedness of the 2D localization problem, we therefore use the reliable 2D joint location heatmaps produced by [38] as input to our autoencoder-based regression architecture. We show that this improves 3D pose accuracy upon direct regression from an RGB image. We further show that our autoencoder-based regression approach scales to very deep architectures and achieves state-of-the-art performance when used with ResNet architecture [18].

Because we can perform 3D pose-estimation using a single CNN, our approach can easily be extended to handling sequences of images instead of single ones. To this end, we introduce two LSTM-based architectures: one that acts on the pose predictions in consecutive images, and one that models temporal information directly at the feature level. Our experiments evidence the additional benefits of modeling this temporal information over our single-frame approach.

In short, our contribution is to show that combining traditional CNNs for supervised learning with autoencoders 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 framework in more detail, introduce our LSTM-based architectures and finally demonstrate that our approach achieves competitive performance with the state-of-the-art methods on standard 3D human pose estimation benchmarks.

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 either directly to pose vectors [58] or to heatmaps encoding 2D joint locations [24,42,57]. This has been exploited very effectively to infer 3D poses by fitting a 3D model to the 2D predictions [4,13,64,66]. This approach currently yields some of the best results, but involves a separate, typically expensive model-fitting stage, outside of the Deep Learning framework.

In parallel, there has been a trend towards performing direct 3D pose estimation [23,32], formulated as a regression problem. In other words, the algorithms output continuous 3D joint locations, because discretizing the 3D space is more challenging than the 2D one.

Our work fits in that line research, which involves dealing with the ambiguities inherent to inferring a 3D pose from a 2D input. To resolve them, recent algorithms have sought to encode the dependencies between the different joints within Deep Learning approaches, thus effectively achieving structured prediction. In particular, [21] uses autoencoders to learn a shared representation for 2D silhouettes and 3D poses. This approach, however, relies on accurate foreground masks and exploits handcrafted features, which mitigates the benefits of Deep Learning. In the context of hand pose estimation, [39] introduces a bottleneck, low dimensional layer that aims at accounting for joint dependencies. This layer, however, is obtained directly via PCA, which limits the range of dependencies it can model.

The work of [33] constitutes an 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.

To achieve a similar result effectively, we drew our inspiration from earlier KDE-based approaches [22,23], which map both image and 3D pose to high-dimensional Hilbert spaces and learn a mapping between these spaces. In this paper, we show how to do this in a Deep Learning context by combining CNNs and autoencoders. Not only does this allow us to leverage the power of learned features, which have proven more effective than hand-designed ones such as HOG [1] and 3D-HOG [62], but it yields a direct and efficient regression between the two spaces. Furthermore, it also allows us to learn the mapping from high-dimensional space to pose space, thus avoiding the need of KDE-based methods to solve an optimization problem at test time.

Using autoencoders for unsupervised feature learning has proven effective in several recognition tasks [31,28,60]. In particular, denoising autoencoders [59] that aim at recon-

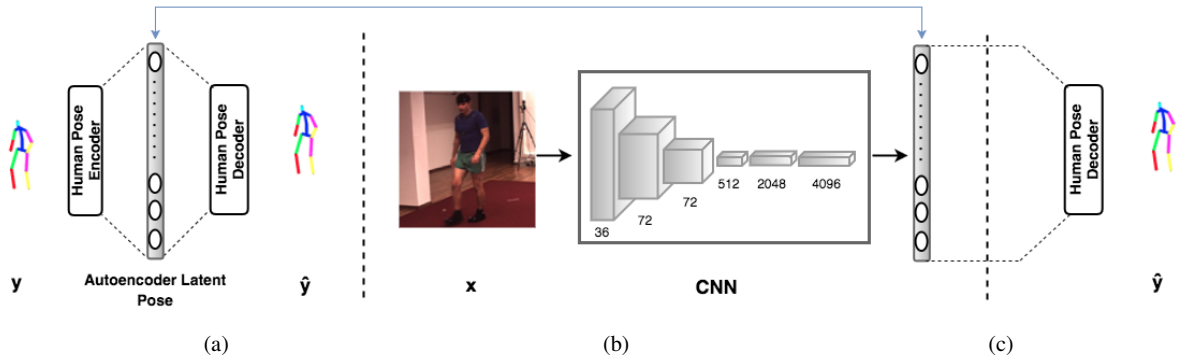


Fig. 1 Our architecture for the structured prediction of the 3D human pose. **(a)** An autoencoder whose hidden layers have a larger dimension than both its input and output layers is pretrained. In practice we use either this one or more sophisticated versions that are described in more detail in Section 3.1 **(b)** A CNN maps either a monocular image or a 2D joint location heatmap to the latent representation learned by the autoencoder. **(c)** The latent representation is mapped back to the original pose space using the decoder.

structing the perfect data from a corrupted version of it have demonstrated good generalization ability. Similarly, contractive autoencoders have been shown to produce intermediate representations that are robust to small variations of the input data [47]. All these methods, however, rely on autoencoders to learn features for recognition tasks. By contrast, here, we exploit them to model the output structure for regression purposes.

In this paper, we further investigate the use of Recurrent Neural Networks (RNNs), and in particular LSTMs, to model temporal information. RNNs have recently been used in many Natural Language Processing [30,53] and Computer Vision [35,43] tasks, and, at the intersection of these fields, for image captioning and video description [10,26]. More closely related to our work, in [14,25], RNNs have been employed to model human dynamics. Nevertheless, these methods do not tackle human pose estimation, but motion capture generation, video pose labeling and forecasting for [14], and human-object interaction prediction for [25]. To the best of our knowledge [34] is the only method that exploits RNNs for 3D human pose estimation from images. However, this approach operates on single images and makes use of RNNs to iteratively refine the pose predictions of [33]. By contrast we leverage the power of RNNs at modeling long term temporal dependencies across image sequences.

3 Method

In this work, we aim at directly regressing from an input image or heatmap x to a 3D human pose. As in [3,23,32], 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 purposes.

As discussed above, a straightforward approach to creating a regressor is to train a conventional CNN such as the one used in [32]. However, this fails to encode dependencies between joint locations. In [33], 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, as evidenced by our experiments, more accurate way by learning a mapping between the output of a CNN and a latent representation obtained using an overcomplete autoencoder, as illustrated in Fig. 2. The autoencoder 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 [9,22, 23], 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.

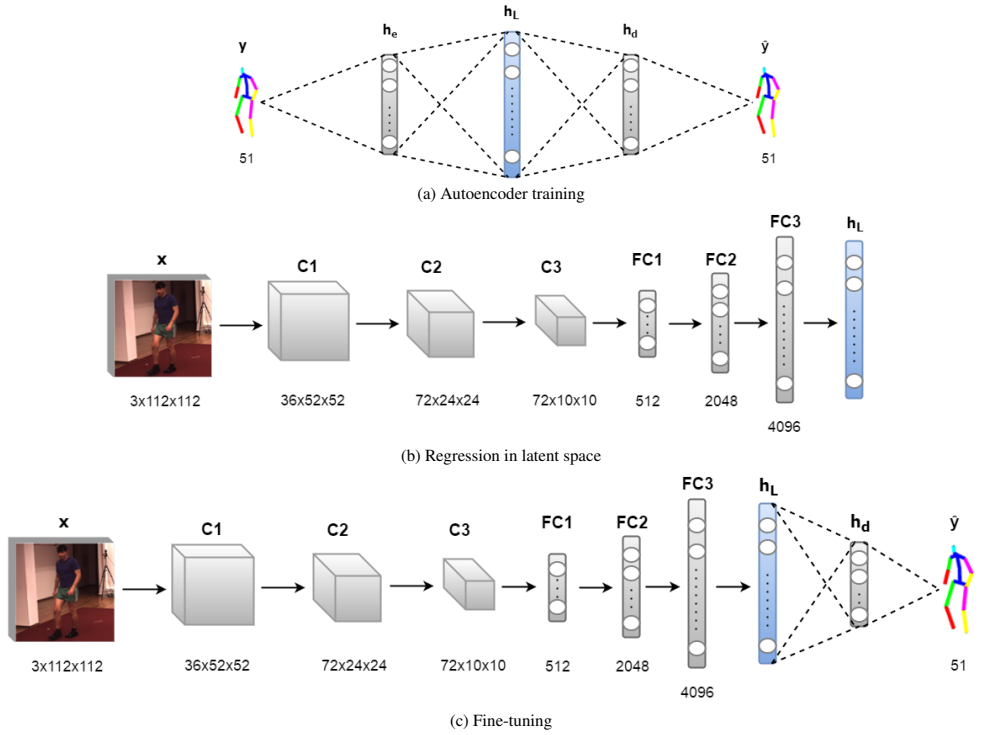
In the remainder of this section, we describe the different stages of our single-frame approach. We then extend this framework to modeling temporal consistency in Section 4.

3.1 Structured Latent Representations via Autoencoders

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 autoencoder that can have one or more hidden layers.

Following standard practice [60], 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 autoencoder to take as input a noisy $\tilde{y}_i$ and return a denoised y_i. The behavior of the autoencoder is controlled by the set $\theta_{ae} = (W_{enc,j}, b_{enc,j}, W_{dec,j}, b_{dec,j})_{j=1}^{L}$ of weights and biases for L encoding and decoding layers.

We take the middle layer to be our latent pose representation and denote it by $h_L = g(\tilde{y}, \theta_{ae})$, where $g(\cdot)$ represents



(a) Autoencoder training

(b) Regression in latent space

(c) Fine-tuning

Fig. 2 Our approach. **(a)** We train a stacked denoising autoencoder that learns the structural information and enforces implicit constraints about human body in its latent middle layer h_L. **(b)** Our CNN architecture maps the raw image or the 2D joint location heatmap predicted from the input image to the latent representation h_L learned by the autoencoder. **(c)** We stack the decoding layers of the autoencoder 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.

the encoding function. For example, with a single layer, the latent representation can be expressed as

$$h_L = g(\tilde{y}, W_{enc}, b_{enc}) = r(W_{enc}\tilde{y} + b_{enc}) \ , \tag{1}$$

where $r(\cdot)$ is the activation function. In practice, we use ReLU as the activation function of the encoding layers. This favors a sparse hidden representation [16], which has been shown to be effective at modeling a wide range of human poses [2,46]. For the decoding part of the autoencoder, we use a linear activation function to be able to predict 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 true, noise-free pose, 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, $J(\tilde{y}) = \frac{\partial g}{\partial \tilde{y}}(\tilde{y})$. Training can thus be expressed as finding

$$\theta_{ae}^* = \underset{\theta_{ae}}{\operatorname{argmin}} \sum_{i=1}^{N} ||y_i - f(\tilde{y}_i, \theta_{ae})||_2^2 + \lambda \|J(\tilde{y}_i)\|_F^2 \ , \tag{2}$$

where $f(\cdot)$ represents the complete autoencoder function, and λ is the regularization weight. Unlike when using KDE, we do not need to solve a complex problem to go from the latent pose representation to the pose itself. This mapping, which corresponds to the decoding part of our autoencoder, is learned directly from data.

3.2 Regression in Latent Space

Once the autoencoder is trained, we aim to learn a mapping from the input image or heatmap to the latent representation of the human pose. To this end, and as shown in Fig. 2(b), we use 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 or heatmap x, we consider the square loss between the representation predicted by the CNN, $f_{cnn}(x, \theta_{cnn})$, and the one that was previously learned by the autoencoder, h_L. Given our N training samples, learning amounts to finding

$$\theta_{cnn}^* = \underset{\theta_{cnn}}{\operatorname{argmin}} \sum_{i=1}^{N} ||f_{cnn}(x_i, \theta_{cnn}) - h_{L,i}||_2^2 \ . \tag{3}$$

In practice, we either rely on a standard CNN architecture shown in Fig. 2(b), similar to the one of [32,58] or a

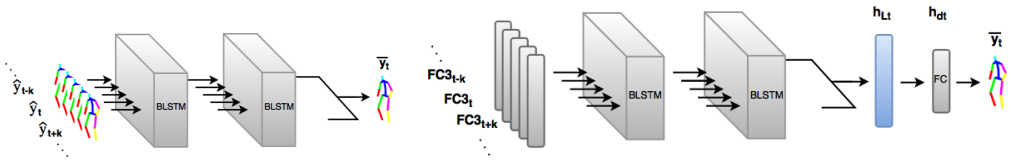


Fig. 3 (B)LSTMs to enforce temporal consistency. (a) The (B)LSTM-Pose approach involves refining 3D human pose predictions by feeding those obtained as described in Fig. 2(c) into a (B)LSTM network, which yields the final 3D poses. (b) The (B)LSTM-Feature approach maps the features obtained from the last fully-connected layer of a CNN trained to directly regress 3D pose from monocular images to the latent representation h_L of Fig. 2(a) via a (B)LSTM network. The final pose is recovered by the decoder part of the autoencoder.

very deep network architecture, e.g. ResNet-50 [18]. In our implementation, the input volume is a three channel image of size 128×128 or a 16 channel heatmap of size 128×128. The last fully-connected layer of the base network is mapped linearly to the latent pose embedding. Except for this last linear layer, each layer uses a ReLU activation function. When we use images as input, we initialize the convolutional layers of our CNN from those of a network trained for the detection of body joints in 2D as in [32,37].

In the case of 3D pose prediction from 2D joint location heatmaps, we rely on the stacked hourglass network design [38], which assigns high confidence values to most likely joint positions in the image. In practice, we have observed a huge performance improvement in overall 3D pose estimation accuracy when using reliable 2D joint location heatmaps produced by stacked hourglass networks compared to directly using RGB images as input to our standard CNN architecture in Fig. 2(b).

3.3 Fine-Tuning the Whole Network

Finally, as shown in Fig. 2(c), we append the decoding layers of the autoencoder to the CNN discussed above, which maps 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 model parameters, including θ_{cnn} and the decoding weights and biases $(W_{dec,j}, b_{dec,j})_{j=1}^{L}$, and f_{ft} is the mapping function.

At test time, a new input image or heatmap is then simply passed forward through this fine-tuned network, which predicts the 3D pose via the learned latent representation.

4 Modeling Temporal Consistency

We have so far focused on predicting 3D poses from single images or heatmaps. However, it is well known that accounting for temporal consistency increases robustness. In this section, we show that our approach naturally allows us to use Long Short-Term Memory Units (LSTMs) to this end. Below, we first briefly review LSTMs and then introduce two different ways to exploit them to encode temporal information in our framework.

4.1 LSTMs

Recurrent Neural Networks (RNNs) have become increasingly popular to model temporal dynamics. In their simplest form, they map a sequence of inputs to a sequence of hidden states, each connected to its temporal neighbors, which are in turn mapped to a sequence of outputs. In theory, simple memory units and backpropagation through time (BPTT) allow RNNs to capture the temporal correlations between distant data points. However, in practice, longer sequences often cause the gradients to either vanish or explode, thus making optimization impossible. LSTMs [20] were introduced as a solution to this problem. Although they have four times as many parameters as traditional RNNs, they can be trained efficiently thanks to their sharing of parameters across time slices. An LSTM unit is defined by the recurrence equations

$$\begin{aligned} i_t &= \sigma_i(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ f_t &= \sigma_f(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ o_t &= \sigma_o(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \sigma_c(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ h_t &= o_t \odot \sigma_h(c_t) \ , \end{aligned} \tag{5}$$

where x_t, c_t and h_t are the input, hidden/cell state and output at time t, respectively, and i_t, f_t and o_t represent gate vectors to forget/select information. $\sigma.(\cdot)$ are sigmoids and $\odot$ denotes the Hadamard or element-wise product.

In practice, we use either LSTMs or Bidirectional LSTMs (BLSTMs). A BLSTM comprises two LSTMs with information traveling in opposite temporal directions [17]. They have been shown to boost performance when the quantity

to be predicted depends on contextual information coming from both forward and backward in time [17]. This is typically the case for human pose estimation, where the estimate at time t is correlated to those at time $t-1$ and $t+1$.

4.2 Recurrent Pose Estimation

We tested two different ways to incorporate (B)LSTMs into our framework.

4.2.1 Constraining the Final Poses

The first is to refine the pose estimates by imposing temporal consistency on the output of the network introduced in the previous section, as shown in Fig. 3(a).

More specifically, let $S_t = [\hat{y}_{t-\frac{T}{2}+1}, \ldots, \hat{y}_t, \ldots, \hat{y}_{t+\frac{T}{2}}]$ be the input sequence of T predicted poses centered at time t. The network prediction can be expressed as

$$\bar{y}_t = f_p(S_t, \theta_p) \ , \tag{6}$$

where θ_p includes all the parameters of the network. During training, these parameters are taken to be

$$\theta_p^* = \underset{\theta_p}{\operatorname{argmin}} \sum_{t=T/2}^{N-T/2} ||f_p(S_t, \theta_p) - y_t||_2^2 \ . \tag{7}$$

We refer to this method as *(B)LSTM-Pose*.

4.2.2 Constraining the Features

An alternative would be to enforce temporal consistency not on the poses, but earlier in the network on the features extracted from a direct CNN regressor. To this end, we made use of the features of the penultimate layer of our base network. This, for example, corresponds to FC3 features for the network shown in Fig. 2(b). These features act as input to the model depicted in Fig. 3(b), which stacks two BLSTM layers and maps the features to the latent representation learned by the autoencoder of Section 3.1. This is followed by the decoder to finally predict 3D poses.

Let $F_t = [\mathrm{FC}_{t-T/2+1}, \ldots, \mathrm{FC}_t, \ldots, \mathrm{FC}_{t+T/2}]$ be the sequence of such features. Then, training this network can be achieved by solving the problem

$$\theta_f^* = \underset{\theta_f}{\operatorname{argmin}} \sum_{t=T/2}^{N-T/2} ||f_f(F_t, \theta_f) - y_t||_2^2 \ , \tag{8}$$

where $f_f(F_t, \theta_f)$ represents the complete network mapping, with parameters θ_f. We refer to this method as *(B)LSTM-Feature*.

5 Results

In this section, we first describe the datasets 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.

5.1 Datasets

We evaluate our method on the Human3.6m [23], HumanEva [51], KTH Multiview Football II [5] and Leeds Sports Pose (LSP) [27] datasets.

Human3.6m 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 [32], we collect the input images by extracting a square region around the subject using the bounding box present in the dataset and the output pose is a vector of 17 3D joint coordinates.

HumanEva-I comprises synchronized images and motion capture data and is a standard benchmark for 3D human pose estimation. The output pose is a vector of 15 3D joint coordinates.

KTH Multiview Football II is a recent benchmark to evaluate the performance of pose estimation algorithms in unconstrained outdoor settings. The camera follows a soccer player moving around the field. The videos are captured from 3 different camera viewpoints and the output pose is a vector of 14 3D joint coordinates.

LSP is a standard benchmark for 2D human pose estimation and does not contain any ground-truth 3D pose data. The images are captured in unconstrained outdoor settings. 2D pose is represented in terms of a vector of 14 joint coordinates. We report qualitative 3D pose estimation results on this dataset.

5.2 Implementation Details

We trained our autoencoder using a greedy layer-wise training scheme followed by fine-tuning as in [19, 60]. We set the regularization weight of Eq. 2 to $\lambda = 0.1$. We experimented with single-layer autoencoders, as well as with 2-layer ones. The size of the layers were set to 2000 and 300-300 for the 1-layer and 2-layer cases, respectively. We corrupted the input pose with zero-mean Gaussian noise with standard deviation of 40 for 1-layer and 40-20 for 2-layer autoencoders. In all

Method	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting	Sitting Down
Ionescu et al. [23]	132.71	183.55	132.37	164.39	162.12	150.61	171.31	151.57	243.03
Li & Chan [32]	-	148.79	104.01	127.17	-	-	-	-	-
Li et al. [33]	-	134.13	97.37	122.33	-	-	-	-	-
Li et al. [34]	-	133.51	97.60	120.41	-	-	-	-	-
Zhou et al. [66]	-	-	-	-	-	-	-	-	-
Rogez & Schmid [48]	-	-	-	-	-	-	-	-	-
Tekin et al. [54]	-	129.06	91.43	121.68	-	-	-	-	-
Park et al. [40]	100.34	116.19	89.96	116.49	115.34	117.57	106.94	137.21	190.82
Zhou et al. [65]	91.83	102.41	96.95	98.75	113.35	90.04	93.84	132.16	158.97
Tome et al. [56]	64.98	73.47	76.82	86.43	86.28	68.93	74.79	110.19	173.91
Pavlakos et al. [41]	67.38	71.95	66.70	69.07	71.95	65.03	68.30	83.66	**96.51**
OURS (ShallowNet-Autoencoder)	94.98	129.06	91.43	121.68	133.54	115.13	133.76	140.78	214.52
OURS (ShallowNet-Hm-Autoencoder)	69.64	93.79	69.02	96.47	103.42	83.36	85.22	116.62	147.57
OURS (ResNet-Autoencoder)	**57.84**	**64.62**	**59.41**	**62.83**	**71.52**	**57.50**	**60.38**	**80.22**	104.14

Method:	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Avg. (6 Actions)	Avg. (All)
Ionescu et al. [23]	162.14	205.94	170.69	96.60	177.13	127.88	159.99	162.14
Li & Chan [32]	-	189.08	-	77.60	146.59	-	132.20	-
Li et al. [33]	-	166.15	-	68.51	132.51	-	120.17	-
Li et al. [34]	-	163.33	-	73.66	135.15	-	121.55	-
Zhou et al. [66]	-	-	-	-	-	-	-	120.99
Rogez & Schmid [48]	-	-	-	-	-	-	-	121.20
Tekin et al. [54]	-	162.17	-	65.75	130.53	-	116.77	-
Park et al. [40]	105.78	149.55	125.12	62.64	131.90	96.18	111.12	117.34
Zhou et al. [65]	106.91	125.22	94.41	79.02	126.04	98.96	104.73	107.26
Tome et al. [56]	84.95	110.67	85.78	71.36	86.26	73.14	84.17	88.39
Pavlakos et al. [41]	71.74	**76.97**	65.83	59.11	74.89	63.24	69.78	71.90
OURS (ShallowNet-Autoencoder)	121.26	162.17	138.2	65.75	130.53	113.34	116.77	127.07
OURS (ShallowNet-Hm-Autoencoder)	87.17	120.50	95.31	55.87	85.69	64.66	86.89	91.62
OURS (ResNet-Autoencoder)	**66.31**	80.50	**61.20**	**52.55**	**69.97**	**60.08**	**61.20**	**67.27**

Table 1 Comparison of our approach with state-of-the-art algorithms on *Human3.6m*. We report 3D joint position errors in mm, computed as the average Euclidean distance between the ground-truth and predicted joint positions. '-' indicates that the results were not reported for the respective action class in the original paper. Note that our method achieves the best overall accuracy.

cases, we used the ADAM optimization procedure [29] with a learning rate of 0.001 and a batch size of 128.

The number and individual sizes of the layers of our base architecture 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 autoencoders, we used ADAM [29] 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 augmented the data by randomly cropping 112×112 patches from the 128×128 image. When using 2D heatmaps as input, the 64×64 outputs of stacked hourglass network of [38] were upscaled to 128×128 before processing.

To demonstrate that our approach scales to very deep architectures, we also use ResNet-50 [18] as baseline CNN architecture. More specifically, we use it up to level 5, with the first three levels initialized on a 2D pose estimation task as in [37] and then kept constant throughout the 3D pose prediction process. We then use two additional convolutional layers of size 512 and 128 and a linear layer to regress the 3D pose from the convolutional features of level 4.

To train *Ours-LSTM-Feature* and *Ours-BLSTM-Feature*, we relied on the features extracted from the penultimate layer of a CNN trained to directly predict 3D pose, referred to later as *CNN-Direct*. We did not backpropagate the loss of our LSTM-based models through this network, but rather kept its weights fixed. By contrast, *Ours-LSTM-Pose* and *Ours-BLSTM-Pose* take as input the 3D pose predictions obtained using the network in Fig. 2(c). In all cases, we cascaded two (B)LSTM layers of size 512 , whose output sequence was merged into a single fully-connected layer of size 51. The activation function was $tanh$ for the recurrent layers and linear for the fully-connected layer at the end. In all architectures, we used a temporal window of length $T = 5$ with a stride of 5 covering 0.5 seconds for 50 fps Human3.6m videos. The first $T/2 - 1$ and the last $T/2$ frames were excluded from the evaluation. We optimized the recurrent networks using the ADAM optimization procedure [29] with a learning rate of 0.001 and a batch size of 128.

5.3 Evaluation Protocol

On Human3.6m, for the comparison to be fair, we used the same data partition protocol as in earlier work [32,33] to obtain the training and test splits. The data from 5 subjects (S1,S5,S6,S7,S8) was used for training and the data from 2 different subjects (S9,S11) was used for testing. We trained a single model for all actions. We evaluate the accuracy of

Model	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting
Bogo et al.[4]	62.0	60.2	67.8	76.5	92.1	73.0	75.3	100.3
Sanzari et al.[50]	48.82	56.31	95.98	84.78	96.47	66.30	107.41	116.89
OURS (ResNet-Autoencoder)	**43.89**	**48.54**	**46.57**	**49.95**	**53.94**	**43.77**	**43.94**	**60.20**

Model	Sitting Down	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Average
Bogo et al.[4]	137.3	83.4	77.0	77.3	86.8	79.7	81.7	82.3
Sanzari et al.[50]	129.63	97.84	105.58	65.94	92.58	130.46	102.21	93.15
OURS (ResNet-Autoencoder)	**73.64**	**51.15**	**59.29**	**46.30**	**39.81**	**52.25**	**47.18**	**50.69**

Table 2 Average Euclidean distance in mm between the ground-truth 3D joint locations and those predicted by competing methods [12,4,50] and ours after Procrustes transformation.

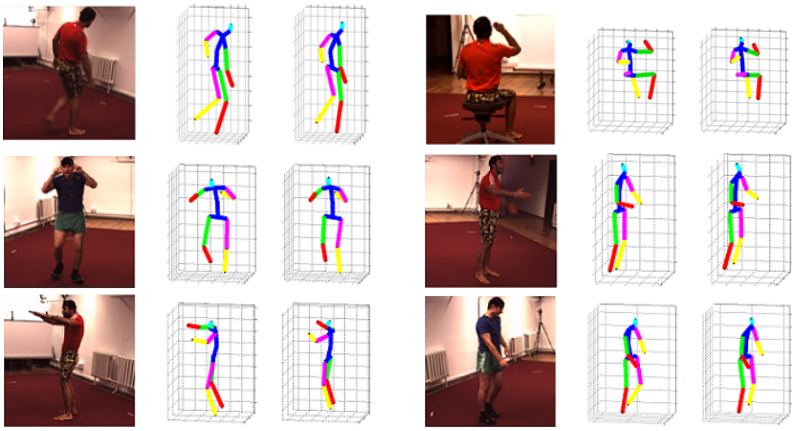


Fig. 4 3D poses for the *Walking, Eating, Taking Photo, Greeting, Discussion* and *Walking Dog* actions of the Human3.6m database. In each case, the first skeleton depicts the ground-truth pose and the second one the pose we recover. Best viewed in color.

3D human pose estimation in terms of average Euclidean distance between the predicted and ground-truth 3D joint positions as in [32,33]. To compare against [4,50], we further evaluate the pose estimation accuracy after Procrustes transformation. The accuracy numbers are reported in milimeters for all actions. Training and testing were carried out monocularly in all camera views for each separate action.

On HumanEva-I, we trained our model on the Walking sequences of subjects S1, S2 and S3 as in [52,66] and evaluate on the validation sequences of all subjects. We pretrained our network on the Walking sequences of Human3.6m and used only the first camera view for further training and validation.

On KTH Multiview Football II, we trained our model on the first half of the sequence containing Player 2 and test on the second half, as in [5]. We report accuracy using the percentage of correctly estimated parts (PCP) score with a threshold of 0.5 for a fair comparison. Since the training set is quite small, we pretrained our CNN model on the synthetic dataset introduced in [7], which contains images of sports players with their corresponding 3D poses.

On LSP, in order to generalize to the unconstrained outdoor settings, we trained our regressor on the recently released synthetic dataset of [7] and tested on the actual data from the LSP dataset.

5.4 Results

We first discuss our results on predicting 3D pose from a single image, and then turn to the case where we use multiple consecutive frames as input.

5.4.1 Human Pose from a Single Image

Fig. 4 depicts selected pose estimation results on Human3.6m. In Table 1, we report our single-image autoencoder-based results on this dataset along with those of the following state-of-the-art single image-based methods: KDE regression from HOG features to 3D poses [23], jointly training a 2D body part detector and a 3D pose regressor [32,40], the maximum-margin structured learning framework of [33,34], the deep structured prediction approach of [54], pose regression with kinematic constraints [65], pose estimation with mocap guided data augmentation [48], volumetric pose prediction approach of [41] and lifting 2D heatmap predictions to 3D human pose [56]. *ShallowNet-Autoencoder* refers to our autoencoder-based regression approach using the base architecture depicted in Fig. 2, and *ResNet-Autoencoder* to the one using the ResNet-50 architecture. For the shallow network architecture, we also evaluate the pose estimation accuracy using the 2D joint location heatmaps of [38] as input. This is referred to as *ShallowNet-Hm-Autoencoder*.

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

(a)

Model	Joint error
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-Autoencoder[40]	165.11
OURS-Autoencoder[2000]	**158.51**

(b)

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

Model	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting
ResNet	56.77	64.73	60.94	63.49	74.98	57.65	61.08	81.29
ResNet-Autoencoder w/o ExtraMocap	57.84	64.62	59.41	62.83	**71.52**	57.50	60.38	**80.22**
ResNet-Autoencoder w/ ExtraMoCap	**55.87**	**63.65**	**59.08**	**62.64**	72.08	**56.15**	**58.88**	80.53

Model	Sitting Down	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Average
ResNet	102.45	66.65	80.96	60.87	53.26	70.27	60.95	68.29
ResNet-Autoencoder w/o ExtraMoCap	104.14	66.31	80.50	61.20	52.55	69.97	60.08	67.27
ResNet-Autoencoder w/ ExtraMoCap	**102.30**	**65.68**	**78.25**	**59.05**	**51.81**	**68.44**	**58.19**	**66.17**

Table 4 Average Euclidean distance in mm between the ground-truth and predicted 3D joint locations a direct ResNet regressor, ResNet-Autoencoder trained with only motion capture data from Human3.6m and ResNet-Autoencoder trained with motion capture data from Human3.6m and MPI-INF-3DHP.

The shallow network architecture provides satisfactory pose estimation accuracy with a fast computational runtime of 6 ms/frame, which corresponds to 166 fps real-time performance, whereas *ResNet-Autoencoder* comes at the cost of a three times slower runtime. Our autoen-coder-based regression approach using ResNet-50 as base network outperforms all the baselines.

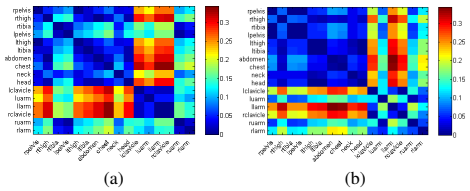
In [4], the reconstruction error was evaluated by first aligning the estimated skeleton to the ground-truth one by Procrustes transformation, and we confirmed through personal communication that the same protocol was used in [50]. To compare our results to those of these state-of-the-art methods, we therefore also report in Table 2 the joint error after Procrustes transformation. Altogether, by leveraging the power of deep neural networks and accounting for the dependencies between body parts, *ResNet-Autoencoder* significantly outperforms the state-of-the-art.

We further evaluated our approach on the official test set of Human3.6m for six different actions [1]. We obtained a pose reconstruction error of 64.38, 63.86, 63.85, 70.45, 86.41 and 93.36 mm for the *Directions*, *Discussion*, *Eating*, *Phoning*, *Sitting* and *Sitting Down* actions, respectively. Our method currently outperforms the first ranking method [45] on the average of these 6 actions in the leaderboard. Note also that the first ranking method [45] relies on the knowledge of body part segmentations whereas we do not use this additional piece of ground-truth information.

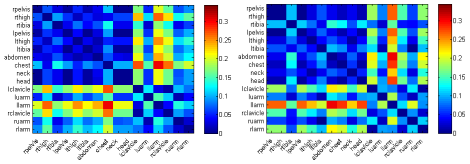
To validate our design choices, we report in Table 3, the pose estimation accuracies obtained with various autoencoder configurations using the shallow network depicted in Fig. 2. The results reported in Tables 1 and 2 were obtained using a two layer autoencoder. However, as discussed in Section 3.1 our formalism applies to autoencoders of any depth. Therefore, in Table 3(a), we also report results obtained using a single layer one obtained 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 autoencoder at all. We will refer to it as *CNN-Direct*. We found that both kinds of autoencoders 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 autoencoding. 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 autoencoder latent representation by a PCA-based one. In Table 3(b), we show that our approach significantly outperforms these two baselines on the *Taking Photo* action. This suggests that our over-complete autoencoder 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 autoencoder with 40 dimensions outperforms it.

[1] We experimented on only 6 actions due to time limitations of the submission server.



(a) (b)



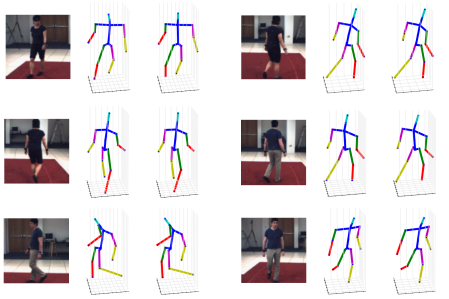
(c) (d)

Model	Lower Body	Upper Body	Full Body
KDE [23]	1.02	7.18	16.43
CNN	0.57	6.86	14.97
OURS-Autoencoder no FT	0.62	5.30	11.99
OURS-Autoencoder with FT	0.77	5.43	11.90

(e)

Fig. 5 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 [22] for **(a)** KDE [23], **(b)** CNN-Direct as in Table 3, 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.

Fig. 6 t-SNE embedding [36] for the latent representation of the poses from the *Sitting Down* category in Human3.6m.



(a) Image (b) Prediction (c) GT (d) Image (e) Prediction (f) GT

Fig. 7 Pose estimation results on HumanEva-I. **(a,d)** Input images. **(b,e)** Recovered pose. **(c,f)** Ground truth. Best viewed in color.

To learn a more powerful latent pose space, we exploit additional motion capture data from the MPI-INF-3DHP dataset [37] for training the autoencoder. In Table 4, we report results with and without this additional data. We achieve better pose estimation accuracy when we train on a wider range of poses. As Human3.6m already includes a large variety of poses and the marker placements between the two datasets do not exactly match each other, we only observe a slight improvement. However, our results suggest that training an autoencoder on a larger pose space without any dataset bias would result in an even more representative latent pose space and, eventually, a higher pose estimation accuracy. We further compare our autoencoder-based regression approach to a direct regression baseline. The relative contribution of the autoencoder on very deep neural networks is smaller than that on a shallower network. However, we still increase the accuracy by applying our autoencoder training on top of the ResNet architecture.

Following [22], we show in Fig. 5 the differences between the ground-truth limb ratios and the limb ratios obtained from predictions based on KDE, *CNN-Direct* and our autoencoder-based approach. These results demonstrate that our predictions better preserve these limb ratios, and thus better model the dependencies between joints.

In Fig. 6, we visualize the latent space learned by the autoencoder after embedding it in 2D using the t-SNE algorithm [36]. It can be seen that the upper left corner spans the downward-facing body poses, the diagonal includes mostly the upright body poses and the lower right corner clusters the forward-facing body poses sitting on the ground. Note that our latent representation covers the entire low-dimensional space, thus making it well-suited to discriminate between poses with small variations.

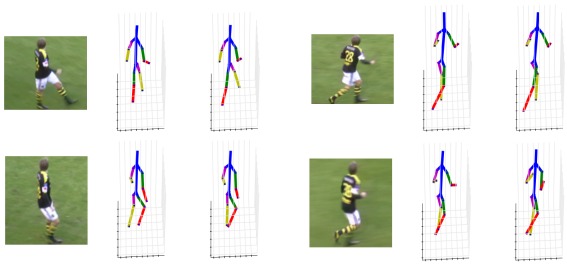
We further report single-image 3D pose estimation accuracy on the HumanEva-I dataset in Table 5 and show qualitative pose estimation results in Fig. 7. We follow the protocol adopted in the state-of-the-art approaches to 3D inference from 2D body part detections [52] and to 3D model-fitting [4, 66]. Following these methods, we measure 3D pose error after aligning the prediction to the ground-truth by a rigid transformation. Note that [66] uses video instead of a single frame for prediction. Our method outperforms the state-of-the-art on this standard benchmark.

Method	S1	S2	S3	Average
Simo-Serra et al. [52]	65.1	48.6	73.5	62.4
Bogo et al. [4]	73.3	59.0	99.4	77.2
Zhou et al. [66]	34.2	30.9	**49.1**	38.07
OURS-Autoencoder	**29.32**	**17.94**	59.51	**35.59**

Table 5 Quantitative results of our approach on Walking sequences of the HumanEva-I dataset [51]. S1, S2 and S3 correspond to Subject 1, 2, and 3, respectively. The accuracy is reported in terms of average Euclidean distance (in mm) between the predicted and ground-truth 3D joint positions.

Method:	Pelvis	Torso	Upper Arms	Lower arms	Upper Legs	Lower Legs	All parts
[5]	**97**	87	14	6	63	41	43
OURS-Autoencoder	66	**100**	**66.5**	**16.5**	**83**	**66.5**	**63.1**

Table 6 On KTH Multiview Football II we compare our method that uses a single image to that of [5]. We rely on the percentage of correctly estimated parts (PCP) score to evaluate performance as in [5]. Higher PCP score corresponds to better 3D pose estimation accuracy.



(a) Image (b) Prediction (c) GT (d) Image (e) Prediction (f) GT

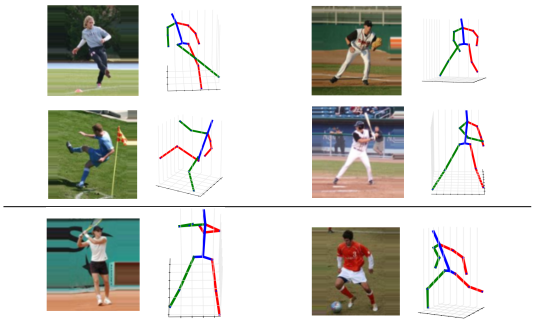
Fig. 8 Pose estimation results on KTH Multiview Football II. **(a, d)** Input images. **(b, e)** Recovered pose. **(c, f)** Ground truth. Best viewed in color.

On the KTH Multiview Football II dataset, we compare our autoencoder-based approach against [5], which is the only monocular single-image 3D pose estimation method publishing results on this dataset so far. As can be seen in Table 6, we outperform the PCP accuracy of this baseline significantly on all body parts except for the pelvis. Fig. 8 depicts example pose estimation results on this dataset.

In Fig. 9, we provide additional qualitative results on the LSP dataset, which features challenging poses. Our autoencoder-based regression approach nevertheless delivers accurate 3D predictions.

5.4.2 Human Pose from Video

In Table 7, we demonstrate the effectiveness of imposing temporal consistency using LSTMs on Human3.6m, as described in Section 4. We compare our results with and without LSTMs against those of [12,55,66], which also rely on video sequences. On average, our LSTM-based approaches applied to the 3D pose predictions of *ResNet-Autoencoder* bring an improvement over single-image results, with the one of Section 4.2.2 that enforces temporal consistency at pose level being significantly better than the other. Using



(a) Image (b) Prediction (c) Image (d) Prediction

Fig. 9 Pose estimation results on LSP. **(a,c)** Input images. **(b,d)** Recovered pose. We trained our network on the recently released synthetic dataset of [7] and tested it on the LSP dataset. The quality of the 3D pose predictions demonstrates the generalization of our method. In the last row, we show failure cases in the 3D pose prediction of lower legs due to foreshortening (left) and orientation ambiguities (right)

.

standard LSTMs instead of BLSTMs degrades the accuracy but eliminates the latency involved in working on image-batches, which can be a worthwhile trade-off if real-time performance is required.

As shown in Table 8, our LSTM units improves the pose estimation accuracy on average by approximately 3% and our ResNet-based results are significantly more accurate than the other methods, with an average pose estimation accuracy of 65.37 mm vs 124.97 mm for [55], 113.01 mm for [66] and 126.47 mm for [12]. Fig. 10 depicts example pose estimation results of our BLSTM approach compared to our autoencoder-based approach based on a single image.

(a) $t-k$ (b) t (c) $t+k$ (d) $t-k$ (e) t (f) $t+k$

Fig. 10 Pose estimation results with LSTMs on Human3.6m. **(a,d)** $t-k^{th}$ frame. **(b,e)** t^{th} frame. **(c,g)** $t+k^{th}$ frame. k denotes the stride between consecutive frames. Top row: Input image, Second row: Our pose estimate from the single image, Third row: Our BLSTM pose estimate, Last row: Ground truth. Our BLSTM network can correct for the errors made by the autoencoder by accounting for the temporal consistency. Best viewed in color.

Model	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting
OURS (ResNet-Autoencoder)	57.84	64.62	59.41	62.83	71.52	57.50	60.38	80.22
OURS-LSTM-Pose	55.63	64.55	57.56	62.20	70.71	56.52	57.37	78.93
OURS-BLSTM-Pose	54.93	63.26	57.26	62.30	70.28	56.66	57.08	78.98
OURS-LSTM-Feature	71.34	68.88	67.12	75.87	79.36	66.19	61.49	83.28
OURS-BLSTM-Feature	70.01	68.74	64.64	75.90	78.99	64.21	60.50	83.10

Model	Sitting Down	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Average
OURS (ResNet-Autoencoder)	104.14	66.31	80.50	61.20	52.55	69.97	60.08	67.27
OURS-LSTM-Pose	98.47	64.43	77.18	62.32	50.12	67.50	66.77	66.02
OURS-BLSTM-Pose	97.13	64.29	77.40	61.94	49.76	67.11	62.26	65.37
OURS-LSTM-Feature	97.66	71.51	83.93	78.67	63.69	73.23	69.03	74.08
OURS-BLSTM-Feature	96.44	70.29	83.51	77.83	62.02	71.11	69.55	73.52

Table 7 Average Euclidean distance in mm between the ground-truth 3D joint locations and the predictions obtained by our ResNet-Autoencoder approach evaluated using different LSTM architectures on video data.

Model	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting
Du et al. [12]	85.07	112.68	104.90	122.05	139.08	105.93	166.16	117.49
Tekin et al. [55]	102.41	147.72	88.83	125.28	118.02	112.3	129.17	138.89
Zhou et al. [66]	87.36	109.31	87.05	103.16	116.18	106.88	99.78	124.52
OURS (ResNet-Autoencoder)	57.84	64.62	59.41	62.83	71.52	57.50	60.38	80.22
OURS-BLSTM-Pose	**54.93**	**63.26**	**57.26**	**62.30**	**70.28**	**56.66**	**57.08**	**78.98**

Model	Sitting Down	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Average
Du et al. [12]	226.04	120.02	135.91	117.65	99.26	137.36	106.54	126.47
Tekin et al. [55]	224.90	118.42	182.73	138.75	55.07	126.29	65.76	124.97
Zhou et al. [66]	199.23	107.42	143.32	118.09	79.39	114.23	97.70	113.01
OURS (ResNet-Autoencoder)	104.14	66.31	80.50	**61.20**	52.55	69.97	**60.08**	67.27
OURS-BLSTM-Pose	**97.13**	**64.29**	**77.40**	61.94	**49.76**	**67.11**	62.26	**65.37**

Table 8 Average Euclidean distance in mm between the ground-truth 3D joint locations and the predictions obtained by our ResNet-Autoencoder approach with and without BLSTM regularization on output poses, compared to [12,55,66] .

Model	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting
HOG + KDE [23]	132.71	183.55	132.37	164.39	162.12	150.61	171.31	151.57
Conv3 Feat. + KDE	99.13	160.84	112.10	137.32	137.97	118.16	137.13	153.79
FC3 Feat. + KDE	99.06	160.39	104.53	132.01	132.35	118.13	144.36	149.80
CNN-Direct	106.23	161.54	108.42	136.15	136.21	123.37	148.68	157.15
OURS-Autoencoder	**94.98**	**129.06**	**91.43**	**121.68**	**133.54**	**115.13**	**133.76**	**140.78**

Model	Sitting Down	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Average
HOG + KDE [23]	243.03	162.14	205.94	170.69	96.60	177.13	127.88	162.14
Conv3 Feat. + KDE	190.48	137.06	181.77	151.15	93.97	149.81	120.46	138.74
FC3 Feat. + KDE	206.35	133.91	169.31	150.76	86.44	144.83	113.20	136.36
CNN-Direct	217.88	136.59	169.42	157.71	88.75	149.58	115.02	140.85
OURS-Autoencoder	**214.52**	**121.26**	**162.17**	**138.2**	**65.75**	**130.53**	**113.34**	**127.07**

Table 9 Average Euclidean distance in mm between the ground-truth 3D joint locations and those predicted by competing methods [23] and ours.

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 10 Average Euclidean distance in mm between the ground-truth 3D joint locations and the ones predicted by our approach trained using autoencoders 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.

We further compare our *OURS-BLSTM-Pose* model with a network where the BLSTM was replaced by two fully-connected layers, thus giving it a similar capacity as the BLSTM one, but not explicitly modeling temporal consistency. This model gives an average pose estimation accuracy on all Human3.6m actions of 77.96 mm, whereas our BLSTM-based model achieves 65.37 mm. Our method significantly outperforms this baseline, thus showing that the better performance of our LSTM-based networks does not just come from their larger number of parameters, but truly from their ability to model temporal information.

5.5 Comparison Between KDE and Autoencoders

In Table 9, we compare two structured 3D human pose estimation methods: Our autoencoder-based deep network approach and kernel dependency estimation (KDE) [22,23]. In the earlier works of [22] and [23], KDE is applied to hand-crafted HOG features, whereas in our approach we rely on deep features. In order to compare the structured regression performance of KDE to our autoencoder-based approach, we also applied KDE to the deep features extracted from a CNN. We extract either the features from the last convolutional layer (Conv3) or the last fully-connected layer (FC3) of the network depicted in Fig. 2(b). As can be seen in Ta-

ble 9, we consistently outperform all the baselines, which demonstrates the power of autoencoding.

5.6 Parameter Choices

In Table 10, we compare the results of different autoencoder configurations in terms of number of layers and channels per layer on the *Greeting* action. Similarly to what we did in Table 3(b), the bracketed numbers denote the dimension of the autoencoder'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.

6 Conclusion

We have introduced a novel Deep Learning regression architecture for structured prediction of 3D human pose from a monocular image or a 2D joint location heatmap. We have shown that our approach to combining autoencoders with CNNs accounts for the dependencies between the human body parts efficiently and significantly improves accuracy. We have also shown that accounting for the temporal information with LSTMs further increases the accuracy of our pose estimates. Since our framework is generic, in future work, we intend to apply it to other structured prediction problems, such as deformable surface reconstruction.

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. L. Bo and C. Sminchisescu. Twin Gaussian Processes for Structured Prediction. *IJCV*, 2010.
4. F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep It SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. In *ECCV*, 2016.
5. M. Burenius, J. Sullivan, and S. Carlsson. 3D Pictorial Structures for Multiple View Articulated Pose Estimation. In *CVPR*, 2013.
6. J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human Pose Estimation with Iterative Error Feedback. In *CVPR*, 2016.
7. W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-or, and B. Chen. Synthesizing Training Images for Boosting Human 3D Pose Estimation. In *3DV*, 2016.
8. X. Chen and A. L. Yuille. Articulated Pose Estimation by a Graphical Model with Image Dependent Pairwise Relations. In *NIPS*, 2014.
9. C. Cortes, M. Mohri, and J. Weston. A General Regression Technique for Learning Transductions. In *ICML*, 2005.
10. J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. In *CVPR*, 2015.
11. M. Du and R. Chellappa. Face Association Across Unconstrained Video Frames Using Conditional Random Fields. In *ECCV*, 2012.
12. Y. Du, Y. Wong, Y. Liu, F. Han, Y. Gui, Z. Wang, M. Kankanhalli, and W. Geng. Marker-Less 3D Human Motion Capture with Monocular Image Sequence and Height-Maps. In *ECCV*, 2016.
13. A. Elhayek, E. Aguiar, A. Jain, J. Tompson, L. Pishchulin, M. Andriluka, C. Bregler, B. Schiele, and C. Theobalt. Efficient Convnet-Based Marker-Less Motion Capture in General Scenes with a Low Number of Cameras. In *CVPR*, 2015.
14. K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent Network Models for Human Dynamics. In *ICCV*, 2015.
15. G. Gkioxari, A. Toshev, and N. Jaitly. Chained Predictions Using Convolutional Neural Networks. In *ECCV*, 2016.
16. X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *AISTATS*, 2011.
17. A. Graves, S. Fernandez, and J. Schmidhuber. Bidirectional LSTM networks for improved phoneme classification and recognition. In *ICANN*, 2005.
18. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.
19. G. Hinton and R. Salakutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 2006.
20. S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.
21. C. Hong, J. Yu, J. Wan, D. Tao, and M. Wang. Multimodal Deep Autoencoder for Human Pose Recovery. *TIP*, 2014.
22. C. Ionescu, F. Li, and C. Sminchisescu. Latent Structured Models for Human Pose Estimation. In *ICCV*, 2011.
23. 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.
24. A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler. Learning Human Pose Estimation Features with Convolutional Networks. In *ICLR*, 2014.
25. A. Jain, A. Zamir, S. Savarese, and A. Saxena. Structural-Rnn: Deep Learning on Spatio-Temporal Graphs. In *CVPR*, 2016.
26. J. Johnson, A. Karpathy, and L. Fei-fei. Densecap: Fully Convolutional Localization Networks for Dense Captioning. In *CVPR*, 2016.
27. S. Johnson and M. Everingham. Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation. In *BMVC*, 2010.
28. D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2014.
29. D.P. Kingma and J. Ba. Adam: A Method for Stochastic Optimisation. In *ICLR*, 2015.
30. S. Kombrink, T. Mikolov, M. Karafiat, and L. Burget. Recurrent Neural Network Based Language Modeling in Meeting Recognition. In *INTERSPEECH*, 2011.
31. K. Konda, R. Memisevic, and D. Krueger. Zero-Bias Autoencoders and the Benefits of Co-Adapting Features. In *ICLR*, 2015.
32. S. Li and A.B. Chan. 3D Human Pose Estimation from Monocular Images with Deep Convolutional Neural Network. In *ACCV*, 2014.
33. S. Li, W. Zhang, and A. B. Chan. Maximum-Margin Structured Learning with Deep Networks for 3D Human Pose Estimation. In *ICCV*, 2015.
34. S. Li, W. Zhang, and A. B. Chan. Maximum-Margin Structured Learning with Deep Networks for 3D Human Pose Estimation. In *IJCV*, 2016.
35. M. Liang and X. Hu. Recurrent Convolutional Neural Network for Object Recognition. In *CVPR*, 2015.
36. L.J.P.v.d. Maaten and G.E. Hinton. Visualizing High Dimensional Data Using t-SNE. *JMLR*, 2008.
37. D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt. Monocular 3D Human Pose Estimation In The Wild Using Improved CNN Supervision. In *International Conference on 3D Vision*, 2017.
38. A. Newell, K. Yang, and J. Deng. Stacked Hourglass Networks for Human Pose Estimation. In *ECCV*, 2016.

39. M. Oberweger, P. Wohlhart, and V. Lepetit. Hands Deep in Deep Learning for Hand Pose Estimation. *arXiv Preprint*, abs/1502.06807, 2015.
40. S. Park, J. Hwang, and N. Kwak. 3D Human Pose Estimation Using Convolutional Neural Networks with 2D Pose Information. In *ECCV Workshops*, 2016.
41. Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *CVPR*, 2017.
42. T. Pfister, J. Charles, and A. Zisserman. Flowing Convnets for Human Pose Estimation in Videos. In *ICCV*, 2015.
43. P.O. Pinheiro and R. Collobert. Recurrent Neural Networks for Scenel Labelling. In *ICML*, 2014.
44. 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.
45. Alin-Ionut Popa, Mihai Zanfir, and Cristian Sminchisescu. Deep multitask architecture for integrated 2d and 3d human sensing. In *CVPR*, 2017.
46. V. Ramakrishna, T. Kanade, and Y. Sheikh. Reconstructing 3D Human Pose from 2D Image Landmarks. In *ECCV*, 2012.
47. S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *ICML*, 2011.
48. G. Rogez and C. Schmid. Mocap Guided Data Augmentation for 3D Pose Estimation in the Wild. In *NIPS*, 2016.
49. M. Salzmann and R. Urtasun. Implicitly Constrained Gaussian Process Regression for Monocular Non-Rigid Pose Estimation. In *NIPS*, December 2010.
50. M. Sanzari, V. Ntouskos, and F. Pirri. Bayesian Image Based 3D Pose Estimation. In *ECCV*, 2016.
51. L. Sigal and M.J. Black. Humaneva: Synchronized Video and Motion Capture Dataset for Evaluation of Articulated Human Motion. Technical report, Department of Computer Science, Brown University, 2006.
52. E. Simo-Serra, A. Quattoni, C. Torras, and F. Moreno-Noguer. A Joint Model for 2D and 3D Pose Estimation from a Single Image. In *CVPR*, 2013.
53. I. Sutskever, G. E. Hinton, and G. W. Taylor. Generating Text with Recurrent Neural Networks. In *ICML*, 2011.
54. B. Tekin, I. Katircioglu, M. Salzmann, V. Lepetit, and P. Fua. Structured Prediction of 3D Human Pose with Deep Neural Networks. In *BMVC*, 2016.
55. B. Tekin, A. Rozantsev, V. Lepetit, and P. Fua. Direct Prediction of 3D Body Poses from Motion Compensated Sequences. In *CVPR*, pages 991–1000, 2016.
56. D. Tome, C. Russell, and L. Agapito. Lifting From the Deep: Convolutional 3D Pose Estimation From a Single Image. In *arXiv preprint, arXiv:1701.00295*, 2017.
57. 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.
58. A. Toshev and C. Szegedy. Deeppose: Human Pose Estimation via Deep Neural Networks. In *CVPR*, 2014.
59. P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In *ICML*, 2008.
60. 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.
61. S. E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional Pose Machines. In *CVPR*, 2016.
62. D. Weinland, M. Ozuysal, and P. Fua. Making Action Recognition Robust to Occlusions and Viewpoint Changes. In *ECCV*, pages 635–648, 2010.
63. Y. Yang and D. Ramanan. Articulated Pose Estimation with Flexible Mixtures-Of-Parts. In *CVPR*, 2011.
64. H. Yasin, U. Iqbal, B. Kruger, A. Weber, and J. Gall. A Dual-Source Approach for 3D Pose Estimation from a Single Image. In *CVPR*, 2016.
65. X. Zhou, X. Sun, W. Zhang, S. Liang, and Y. Wei. Deep Kinematic Pose Regression. In *ECCV Workshops*, 2016.
66. X. Zhou, M. Zhu, S. Leonardos, K. Derpanis, and K. Daniilidis. Sparseness Meets Deepness: 3D Human Pose Estimation from Monocular Video. In *CVPR*, 2016.