3D human pose estimation in video with temporal convolutions and semi-supervised training

Dario Pavllo*
ETH Zürich

Christoph Feichtenhofer
Facebook AI Research

David Grangier*
Google Brain

Michael Auli
Facebook AI Research

Abstract

In this work, we demonstrate that 3D poses in video can be effectively estimated with a fully convolutional model based on dilated temporal convolutions over 2D keypoints. We also introduce back-projection, a simple and effective semi-supervised training method that leverages unlabeled video data. We start with predicted 2D keypoints for unlabeled video, then estimate 3D poses and finally back-project to the input 2D keypoints. In the supervised setting, our fully-convolutional model outperforms the previous best result from the literature by 6 mm mean per-joint position error on Human3.6M, corresponding to an error reduction of 11%, and the model also shows significant improvements on HumanEva-I. Moreover, experiments with back-projection show that it comfortably outperforms previous state-of-the-art results in semi-supervised settings where labeled data is scarce. Code and models are available at https://github.com/facebookresearch/VideoPose3D

1. Introduction

Our work focuses on 3D human pose estimation in video. We build on the approach of state-of-the-art methods which formulate the problem as 2D keypoint detection followed by 3D pose estimation [41, 52, 34, 50, 10, 40, 56, 33]. While splitting up the problem arguably reduces the difficulty of the task, it is inherently ambiguous as multiple 3D poses can map to the same 2D keypoints. Previous work tackled this ambiguity by modeling temporal information with recurrent neural networks [16, 27]. On the other hand, convolutional networks have been very successful in modeling temporal information in tasks that were traditionally tackled with RNNs, such as neural machine translation [11], language modeling [7], speech generation [55], and speech recognition [6]. Convolutional models enable parallel processing of multiple frames which is not possible with recurrent networks.

*Work done while at Facebook AI Research.

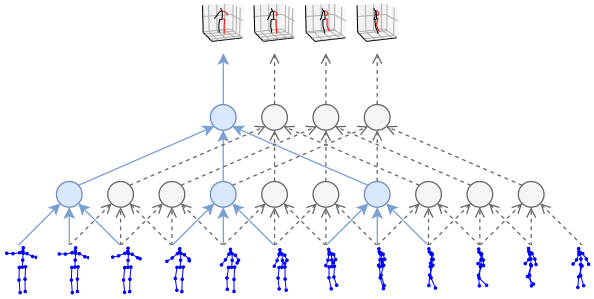


Figure 1: Our temporal convolutional model takes 2D keypoint sequences (bottom) as input and generates 3D pose estimates as output (top). We employ dilated temporal convolutions to capture long-term information.

In this paper, we present a fully convolutional architecture that performs temporal convolutions over 2D keypoints for accurate 3D pose prediction in video (see Figure 1). Our approach is compatible with any 2D keypoint detector and can effectively handle large contexts via dilated convolutions. Compared to approaches relying on RNNs [16, 27], it provides higher accuracy, simplicity, as well as efficiency, both in terms of computational complexity as well as the number of parameters (§3).

Equipped with a highly accurate and efficient architecture, we turn to settings where labeled training data is scarce and introduce a new scheme to leverage unlabeled video data for *semi-supervised training*. Low resource settings are particularly challenging for neural network models which require large amounts of labeled training data and collecting labels for 3D human pose estimation requires an expensive motion capture setup as well as lengthy recording sessions. Our method is inspired by cycle consistency in unsupervised machine translation, where round-trip translation into an intermediate language and back into the original language should be close to the identity function [46, 26, 9]. Specifically, we predict 2D keypoints for an unlabeled video with an off the shelf 2D keypoint detector, predict 3D poses, and then map these back to 2D space (§4).

In summary, this paper provides two main contributions. First, we present a simple and efficient approach for 3D human pose estimation in video based on dilated temporal convolutions on 2D keypoint trajectories. We show that our model is more efficient than RNN-based models at the same level of accuracy, both in terms of computational complexity and the number of model parameters.

Second, we introduce a semi-supervised approach which exploits unlabeled video, and is effective when labeled data is scarce. Compared to previous semi-supervised approaches, we only require camera intrinsic parameters rather than ground-truth 2D annotations or multi-view imagery with extrinsic camera parameters.

In comparison to the state of the art our approach outperforms the previously best performing methods in both supervised and semi-supervised settings. Our supervised model performs better than other models even if these exploit extra labeled data for training.

2. Related work

Before the success of deep learning, most approaches to 3D pose estimation were based on feature engineering and assumptions about skeletons and joint mobility [48, 42, 20, 18]. The first neural methods with convolutional neural networks (CNN) focused on end-to-end reconstruction [28, 53, 51, 41] by directly estimating 3D poses from RGB images without intermediate supervision.

Two-step pose estimation. A new family of 3D pose estimators builds on top of 2D pose estimators by first predicting 2D joint positions in image space (*keypoints*) which are subsequently lifted to 3D [21, 34, 41, 52, 4, 16]. These approaches outperform the end-to-end counterparts, since they benefit from intermediate supervision. We follow this approach. Recent work shows that predicting 3D poses is relatively straightforward given ground-truth 2D keypoints, and that the difficulty lies in predicting accurate 2D poses [34]. Early approaches [21, 4] simply perform a k-nearest neighbour search for a predicted set of 2D keypoints over a large set of 2D keypoints for which the 3D pose is available and then simply output the corresponding 3D pose. Some approaches leverage both image features and 2D ground-truth poses [39, 41, 52]. Alternatively, the 3D pose can be predicted from a given set of 2D keypoints by simply predicting their depth [58]. Some works enforce priors about bone lengths and projection consistency with the 2D ground truth [2].

Video pose estimation. Most previous work operates in a single-frame setting but recently there have been efforts in exploiting temporal information from video to produce more robust predictions and to be less sensitive to noise. [53] infer 3D poses from the HoG features (histograms of oriented gradients) of spatio-temporal volumes. LSTMs have been used to refine 3D poses predicted from single images [30, 24]. The most successful approaches, however, learn from *2D keypoint trajectories*. Our work falls under this category.

Recently, LSTM sequence-to-sequence learning models have been proposed, which encode a sequence of 2D poses from a video into a fixed-size vector that is then decoded into a sequence of 3D poses [16]. However, both the input and output sequences have the same length and a deterministic transformation of 2D poses is a much more natural choice. Our experiments with *seq2seq* models showed that output poses tend to drift over lengthy sequences. [16] tackles this problem by re-initializing the encoder every 5 frames, at the expense of temporal consistency. There has also been work on RNN approaches which consider priors on body part connectivity [27].

Semi-supervised training. There has been work on multitask networks [3] for joint 2D and 3D pose estimation [36, 33] as well as action recognition [33]. Some works transfer the features learned for 2D pose estimation to the 3D task [35]. Unlabeled multi-view recordings have been used for pre-training representations for 3D pose estimation [45], but these recordings are not readily available in unsupervised settings. Generative adversarial networks (GAN) can discriminate realistic poses from unrealistic ones in a second dataset where only 2D annotations are available [56], thus providing a useful form of regularization. [54] use GANs to learn from unpaired 2D/3D datasets and include a 2D projection consistency term. Similarly, [8] discriminate generated 3D poses after randomly projecting them to 2D. [40] propose a weakly-supervised approach based on ordinal depth annotations which leverages a 2D pose dataset augmented with depth comparisons, e.g. "the left leg is behind the right leg".

3D shape recovery. While this paper and the discussed related work focus on reconstructing accurate 3D poses, a parallel line of research aims at recovering full 3D shapes of people from images [1, 23]. These approaches are typically based on parameterized 3D meshes and give less importance to pose accuracy.

Our work. Compared to [41, 40], we do not use heatmaps and instead describe poses with detected keypoint coordinates. This allows the use of efficient 1D convolution over coordinate time series, instead of 2D convolutions over individual heatmaps (or 3D convolutions over heatmap sequences). Our approach also makes computational complexity independent of keypoint spatial resolution. Our models can reach high accuracy with fewer parameters and allow for faster training and inference. Compared to the single-frame baseline proposed by [34] and the LSTM model by [16], we exploit temporal information by performing 1D convolutions over the time dimension, and we propose several optimizations that result in lower reconstruction error. Unlike [16], we learn a deterministic mapping

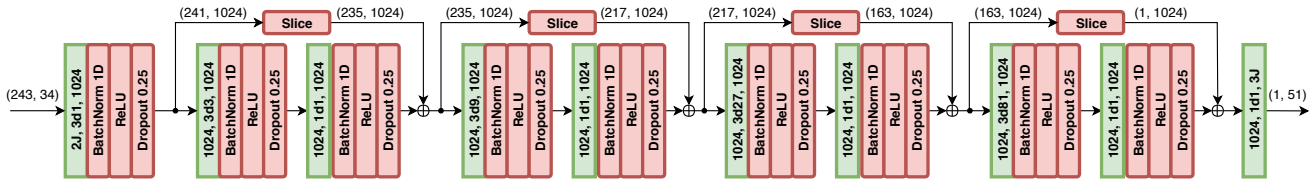


Figure 2: An instantiation of our fully-convolutional 3D pose estimation architecture. The input consists of 2D keypoints for a recpetive field of 243 frames ($B = 4$ blocks) with $J = 17$ joints. Convolutional layers are in green where `2J, 3d1, 1024` denotes $2 \cdot J$ input channels, kernels of size 3 with dilation 1, and 1024 output channels. We also show tensor sizes in parentheses for a sample 1-frame prediction, where `(243, 34)` denotes 243 frames and 34 channels. Due to valid convolutions, we slice the residuals (left and right, symmetrically) to match the shape of subsequent tensors.

instead of a *seq2seq* model. Finally, contrary to most of the two-step models mentioned in this section (which use the popular *stacked hourglass network* [38] for 2D keypoint detection), we show that Mask R-CNN [12] and cascaded pyramid network (CPN) [5] detections are more robust for 3D human pose estimation.

3. Temporal dilated convolutional model

Our model is a fully convolutional architecture with residual connections that takes a sequence of 2D poses as input and transforms them through temporal convolutions. Convolutional models enable parallelization over both the batch and the time dimension while RNNs cannot be parallelized over time. In convolutional models, the path of the gradient between output and input has a fixed length regardless of the sequence length, which mitigates vanishing and exploding gradients which affect RNNs. A convolutional architecture also offers precise control over the temporal receptive field, which we found beneficial to model temporal dependencies for the task of 3D pose estimation. Moreover, we employ *dilated convolutions* [15] to model long-term dependencies while at the same time maintaining efficiency. Architectures with dilated convolutions have been successful for audio generation [55], semantic segmentation [57] and machine translation [22].

The input layer takes the concatenated (x, y) coordinates of the J joints for each frame and applies a temporal convolution with kernel size W and C output channels. This is followed by B ResNet-style blocks which are surrounded by a skip-connection [13]. Each block first performs a 1D convolution with kernel size W and dilation factor $D = W^B$, followed by a convolution with kernel size 1. Convolutions (except the very last layer) are followed by batch normalization [17], rectified linear units [37], and dropout [49]. Each block increases the receptive field exponentially by a factor of W, while the number of parameters increases only linearly. The filter hyperparameters, W and D, are set so that the receptive field for any output frame forms a tree that covers all input frames (see §1). Finally, the last layer outputs a prediction of the 3D poses for all frames in the input sequence using both past and future data to exploit temporal information. To evaluate real-time scenarios, we also experiment with *causal convolutions*, *i.e.* convolutions that only have access to past frames. Appendix A.1 illustrates dilated convolutions and causal convolutions.

Convolutional image models typically apply zero-padding to obtain as many outputs as inputs. Early experiments however showed better results when performing only unpadded convolutions while padding the input sequence with replica of the boundary frames to the left and the right (see Appendix A.5, Figure 9a for an illustration).

Figure 2 shows an instantiation of our architecture for a receptive field size of *243 frames* with $B = 4$ blocks. For convolutional layers, we set $W = 3$ with $C = 1024$ output channels and we use a dropout rate $p = 0.25$.

4. Semi-supervised approach

We introduce a semi-supervised training method to improve accuracy in settings where the availability of labeled 3D ground-truth pose data is limited. We leverage *unlabeled* video in combination with an off the shelf 2D keypoint detector to extend the supervised loss function with a *back-projection* loss term. We solve an auto-encoding problem on unlabeled data: the encoder (pose estimator) performs 3D pose estimation from 2D joint coordinates and the decoder (projection layer) projects the 3D pose back to 2D joint coordinates. Training penalizes when the 2D joint coordinates from the decoder are far from the original input.

Figure 3 represents our method which combines our supervised component with our unsupervised component which acts as a regularizer. The two objectives are optimized jointly, with the labeled data occupying the first half of a batch, and the unlabeled data occupying the second half. For the labeled data we use the ground truth 3D poses as target and train a supervised loss. The unlabeled data is used to implement an autoencoder loss where the predicted 3D poses are projected back to 2D and then checked for consistency with the input.

Trajectory model. Due to the perspective projection, the 2D pose on the screen depends both on the trajectory (i.e.

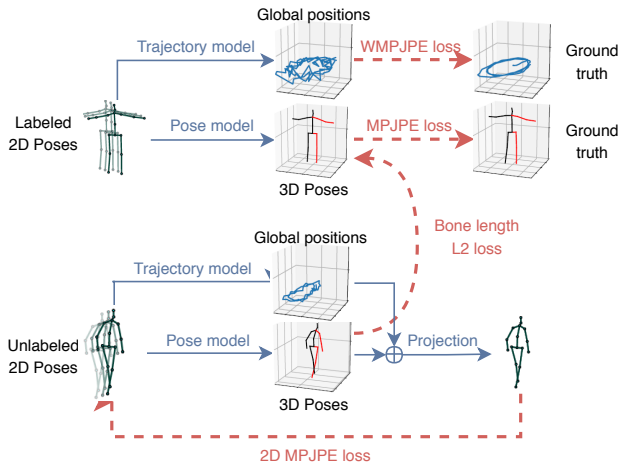


Figure 3: Semi-supervised training with a 3D pose model that takes a sequence of possibly predicted 2D poses as input. We regress the 3D trajectory of the person and add a soft-constraint to match the mean bone lengths of the unlabeled predictions to the labeled ones. Everything is trained jointly. WMPJPE stands for "Weighted MPJPE".

the global position of the human root joint) and the 3D pose (the position of all joints with respect to the root joint). Without the global position, the subject would always be reprojected at the center of the screen with a fixed scale. We therefore also regress the 3D trajectory of the person, so that the back-projection to 2D can be performed correctly. To this end, we optimize a second network which regresses the global trajectory *in camera space*. The latter is added to the pose before projecting it back to 2D. The two networks have the same architecture but do not share any weights as we observed that they affect each other negatively when trained in a multi-task fashion. As it becomes increasingly difficult to regress a precise trajectory if the subject is further away from the camera, we optimize a weighted mean per-joint position error (WMPJPE) loss function for the trajectory:

$$E = \frac{1}{\mathbf{y}_z} \| f(\mathbf{x}) - \mathbf{y} \| \tag{1}$$

that is, we weight each sample using the inverse of the ground-truth depth ($\mathbf{y}_z$) in camera space. Regressing a precise trajectory for far subjects is also unnecessary for our purposes, since the corresponding 2D keypoints tend to concentrate around a small area.

Bone length L2 loss. We would like to incentivize the prediction of plausible 3D poses instead of just copying the input. To do so, we found it effective to add a soft constraint to approximately match the mean bone lengths of the subjects in the unlabeled batch to the subjects of the labeled batch ("Bone length L2 loss" in Figure 3). This term plays an important role in self-supervision, as we show in §6.2.

Discussion. Our method only requires the camera *intrinsic* parameters, which are often available for commercial cameras.[1] The approach is not tied to any specific network architecture and can be applied to any 3D pose detector which takes 2D keypoints as inputs. In our experiments we use the architecture described in §3 to map 2D poses to 3D. To project 3D poses to 2D, we use a simple projection layer which considers linear parameters (focal length, principal point) as well as non-linear lens distortion coefficients (tangential and radial). We found the lens distortions of the cameras used in Human3.6M have negligible impact on the pose estimation metric, but we include these terms nonetheless because they always provide a more accurate modeling of the real camera projection.

5. Experimental setup

5.1. Datasets and Evaluation

We evaluate on two motion capture datasets, Human3.6M [20, 19] and HumanEva-I [47]. Human3.6M contains 3.6 million video frames for 11 subjects, of which seven are annotated with 3D poses. Each subject performs 15 actions that are recorded using four synchronized cameras at 50 Hz. Following previous work [41, 52, 34, 50, 10, 40, 56, 33], we adopt a 17-joint skeleton, train on five subjects (S1, S5, S6, S7, S8), and test on two subjects (S9 and S11). We train a single model for all actions.

HumanEva-I is a much smaller dataset, with three subjects recorded from three camera views at 60 Hz. Following [34, 16], we evaluate on three actions (Walk, Jog, Box) by training a different model for each action (*single action* – SA). We also report results when training one model for all actions (*multi action* – MA), as in [41, 27]. We adopt a 15-joint skeleton and use the provided train/test split.

In our experiments, we consider three evaluation protocols: **Protocol 1** is the mean per-joint position error (MPJPE) in millimeters which is the mean Euclidean distance between predicted joint positions and ground-truth joint positions and follows [29, 53, 59, 34, 41]. **Protocol 2** reports the error after alignment with the ground truth in translation, rotation, and scale (P-MPJPE) [34, 50, 10, 40, 56, 16]. **Protocol 3** aligns predicted poses with the ground-truth only in scale (N-MPJPE) following [45] for semi-supervised experiments.

5.2. Implementation details for 2D pose estimation

Most previous work [34, 58, 52] extracts the subject from ground-truth bounding boxes and then applies the stacked hourglass detector to predict the 2D keypoint locations within the ground-truth bounding box [38]. Our approach (§3 and §4) does not depend on any particular 2D

[1] Even low-end devices typically embed this information in the EXIF metadata of images or videos.

keypoint detector. We therefore investigate several 2D detectors that do not rely on ground-truth boxes which enables the use of our setup in the wild. In addition to the *stacked hourglass detector*, we investigate *Mask R-CNN* [12] with a ResNet-101-FPN [31] backbone, using its reference implementation in *Detectron*, as well as *cascaded pyramid network* (CPN) [5] which represents an extension of FPN. The CPN implementation requires bounding boxes to be provided externally (we use Mask R-CNN boxes for this case).

For both Mask R-CNN and CPN, we start with pretrained models on COCO [32] and fine-tune the detectors on 2D projections of Human3.6M, since the keypoints in COCO differ from Human3.6M [20]. In our ablations, we also experiment with directly applying our 3D pose estimator to pretrained 2D COCO keypoints for estimating the 3D joints of Human3.6M.

For Mask R-CNN, we adopt a ResNet-101 backbone trained with the "stretched 1x" schedule [12].[2] When fine-tuning the model on Human3.6M, we reinitialize the last layer of the keypoint network, as well as the deconv layers that regress the heatmaps to learn a new set of keypoints. We train on 4 GPUs with a step-wise decaying learning rate: 1e-3 for 60k iterations, then 1e-4 for 10k iterations, and 1e-5 for 10k iterations. At inference, we apply a softmax over the the heatmaps and extract the expected value of the resulting 2D distribution (*soft-argmax*). This results in smoother and more precise predictions than *hard-argmax* [33].

For CPN, we use a ResNet-50 backbone with a 384×288 resolution. To fine-tune, we re-initialize the final layers of both *GlobalNet* and *RefineNet* (convolution weights and batch normalization statistics). Next, we train on one GPU with batches of 32 images and with a step-wise decaying learning rate: 5e-5 (1/10th of the initial value) for 6k iterations, then 5e-6 for 4k iterations, and finally 5e-7 for 2k iterations. We keep batch normalization enabled while fine-tuning. We train with ground-truth bounding boxes and test using the bounding boxes predicted by the fine-tuned Mask R-CNN model.

5.3. Implementation details for 3D pose estimation

For consistency with other work [34, 29, 53, 59, 34, 41], we train and evaluate on 3D poses in *camera space* by rotating and translating the ground-truth poses according to the camera transformation, and not using the global trajectory (except for the semi-supervised setting, §4).

As optimizer we use Amsgrad [43] and train for 80 epochs. For Human3.6M, we adopt an exponentially decaying learning rate schedule, starting from $\eta = 0.001$ with a shrink factor $\alpha = 0.95$ applied each epoch.

All temporal models, *i.e.* models with receptive fields larger than one, are sensitive to the correlation of samples in pose sequences (*cf.* §3). This results in biased statistics for batch normalization which assumes independent samples [17]. In preliminary experiments, we found that predicting a large number of adjacent frames during training yields results that are worse than a model exploiting no temporal information (which has well-randomized samples in the batch). We reduce correlation in the training samples by choosing training clips from different video segments. The clip set size is set to the width of the receptive field of our architecture so that the model predicts a single 3D pose per training clip. This is important for generalization and we analyze it in detail in Appendix A.5.

We can greatly optimize this single frame setting by replacing dilated convolutions with strided convolutions where the stride is set to be the dilation factor (see Appendix A.6). This avoids computing states that are never used and we apply this optimization only during training. At inference, we can process entire sequences and reuse intermediate states of other 3D frames for faster inference. This is possible because our model does not use any form of pooling over the time dimension. To avoid losing frames to valid convolutions, we pad by replication, but only at the input boundaries of a sequence (Appendix A.5, Figure 9a shows an illustration).

We observed that the default hyperparameters of batch normalization lead to large fluctuations of the test error ($\pm$ 1 mm) as well as to fluctuations in the running estimates for inference. To achieve more stable running statistics, we use a schedule for the batch-normalization momentum β: we start from $\beta = 0.1$, and decay it exponentially so that it reaches $\beta = 0.001$ in the last epoch.

Finally, we perform horizontal flip augmentation at train and test time. We show the effect of this in Appendix A.4.

For HumanEva, we use $N = 128$, $\alpha = 0.996$, and train for 1000 epochs using a receptive field of 27 frames. Some frames in HumanEva are corrupted by sensor dropout and we split the corrupted videos into valid contiguous chunks and treat them as independent videos.

6. Results

6.1. Temporal dilated convolutional model

Table 1 shows results for our convolutional model with $B = 4$ blocks and a receptive field of 243 input frames for both evaluation protocols (§5). The model has lower average error than all other approaches under both protocols, and does not rely on additional data such as many other approaches (+). Under protocol 1 (Table 1a), our model outperforms the previous best result [27] by 6 mm on average, corresponding to an 11% error reduction. Notably, [27] uses ground-truth boxes whereas our model does not.

The model clearly takes advantage of temporal infor-

[2] https://github.com/facebookresearch/Detectron/blob/master/configs/12_2017_baselines/e2e_keypoint_rcnn_R-101-FPN_s1x.yaml

	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	Avg
Pavlakos *et al*. [41] CVPR'17 (∗)	67.4	71.9	66.7	69.1	72.0	77.0	65.0	68.3	83.7	96.5	71.7	65.8	74.9	59.1	63.2	71.9
Tekin *et al*. [52] ICCV'17	54.2	61.4	60.2	61.2	79.4	78.3	63.1	81.6	70.1	107.3	69.3	70.3	74.3	51.8	63.2	69.7
Martinez *et al*. [34] ICCV'17 (∗)	51.8	56.2	58.1	59.0	69.5	78.4	55.2	58.1	74.0	94.6	62.3	59.1	65.1	49.5	52.4	62.9
Sun *et al*. [50] ICCV'17 (+)	52.8	54.8	54.2	54.3	61.8	67.2	53.1	53.6	71.7	86.7	61.5	53.4	61.6	47.1	53.4	59.1
Fang *et al*. [10] AAAI'18	50.1	54.3	57.0	57.1	66.6	73.3	53.4	55.7	72.8	88.6	60.3	57.7	62.7	47.5	50.6	60.4
Pavlakos *et al*. [40] CVPR'18 (+)	48.5	54.4	54.4	52.0	59.4	65.3	49.9	52.9	65.8	71.1	56.6	52.9	60.9	44.7	47.8	56.2
Yang *et al*. [56] CVPR'18 (+)	51.5	58.9	50.4	57.0	62.1	65.4	49.8	52.7	69.2	85.2	57.4	58.4	43.6	60.1	47.7	58.6
Luvizon *et al*. [33] CVPR'18 (∗)(+)	49.2	51.6	47.6	50.5	51.8	60.3	48.5	51.7	61.5	70.9	53.7	48.9	57.9	44.4	48.9	53.2
Hossain & Little [16] ECCV'18 (†)(∗)	48.4	50.7	57.2	55.2	63.1	72.6	53.0	51.7	66.1	80.9	59.0	57.3	62.4	46.6	49.6	58.3
Lee *et al*. [27] ECCV'18 (†)(∗)	**40.2**	49.2	47.8	52.6	50.1	75.0	50.2	**43.0**	**55.8**	73.9	54.1	55.6	58.2	43.3	43.3	52.8
Ours, single-frame	47.1	50.6	49.0	51.8	53.6	61.4	49.4	47.4	59.3	67.4	52.4	49.5	55.3	39.5	42.7	51.8
Ours, 243 frames, causal conv. (†)	45.9	48.5	44.3	47.8	51.9	57.8	46.2	45.6	59.9	68.5	50.6	46.4	51.0	34.5	35.4	49.0
Ours, 243 frames, full conv. (†)	45.2	**46.7**	43.3	**45.6**	**48.1**	**55.1**	44.6	44.3	57.3	**65.8**	**47.1**	**44.0**	**49.0**	32.8	**33.9**	**46.8**
Ours, 243 frames, full conv. (†)(∗)	45.1	47.4	**42.0**	46.0	49.1	56.7	**44.5**	44.4	57.2	66.1	47.5	44.8	49.2	**32.6**	34.0	47.1

(a) Protocol 1: reconstruction error (MPJPE).

	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	Avg
Martinez *et al*. [34] ICCV'17 (∗)	39.5	43.2	46.4	47.0	51.0	56.0	41.4	40.6	56.5	69.4	49.2	45.0	49.5	38.0	43.1	47.7
Sun *et al*. [50] ICCV'17 (+)	42.1	44.3	45.0	45.4	51.5	53.0	43.2	41.3	59.3	73.3	51.0	44.0	48.0	38.3	44.8	48.3
Fang *et al*. [10] AAAI'18	38.2	41.7	43.7	44.9	48.5	55.3	40.2	38.2	54.5	64.4	47.2	44.3	47.3	36.7	41.7	45.7
Pavlakos *et al*. [40] CVPR'18 (+)	34.7	39.8	41.8	38.6	42.5	47.5	38.0	36.6	50.7	56.8	42.6	39.6	43.9	32.1	36.5	41.8
Yang *et al*. [56] CVPR'18 (+)	**26.9**	**30.9**	36.3	39.9	43.9	47.4	**28.8**	**29.4**	**36.9**	58.4	41.5	**30.5**	**29.5**	42.5	32.2	37.7
Hossain & Little [16] ECCV'18 (†)(∗)	35.7	39.3	44.6	43.0	47.2	54.0	38.3	37.5	51.6	61.3	46.5	41.4	47.3	34.2	39.4	44.1
Ours, single-frame	36.0	38.7	38.0	41.7	40.1	45.9	37.1	35.4	46.8	53.4	41.4	36.9	43.1	30.3	34.8	40.0
Ours, 243 frames, causal conv. (†)	35.1	37.7	36.1	38.8	38.5	44.7	35.4	34.7	46.7	53.9	39.6	35.4	39.4	27.3	28.6	38.1
Ours, 243 frames, full conv. (†)	34.1	36.1	34.4	**37.2**	**36.4**	**42.2**	34.4	33.6	45.0	**52.5**	**37.4**	33.8	37.8	**25.6**	**27.3**	**36.5**
Ours, 243 frames, full conv. (†)(∗)	34.2	36.8	**33.9**	37.5	37.1	43.2	34.4	33.5	45.3	52.7	37.7	34.1	38.0	25.8	27.7	36.8

(b) Protocol 2: reconstruction error after rigid alignment with the ground truth (P-MPJPE), where available.

Table 1: Reconstruction error on Human3.6M. **Legend:** (†) uses temporal information. (∗) ground-truth bounding boxes. (+) extra data – [50, 40, 56, 33] use 2D annotations from the MPII dataset, [40] uses additional data from the Leeds Sports Pose (LSP) dataset as well as ordinal annotations. [50, 33] evaluate every 64th frame. [16] provided us with corrected results over the originally published results [3]. Lower is better, best in bold, second best underlined.

mation as the error is about 5 mm higher on average for protocol 1 compared to a single-frame baseline where we set the width of all convolution kernels to $W = 1$. The gap is larger for highly dynamic actions, such as "Walk" (6.7 mm) and "Walk Together" (8.8 mm). The performance for a model with causal convolutions is about half way between the single frame baseline and our model; causal convolutions enable online processing by predicting the 3D pose for the rightmost input frame. Interestingly, ground-truth bounding boxes result in similar performance to predicted bounding boxes with Mask R-CNN, which suggests that predictions are almost-perfect in our single-subject scenario. Figure 4 shows examples of predicted poses including the predicted 2D keypoints and we included a video illustration in the supplementary material (Appendix A.7) as well as at https://dariopavllo.github.io/VideoPose3D.

Next, we evaluate the impact of the 2D keypoint detector on the final result. Table 3 reports accuracy of our model with ground-truth 2D poses, hourglass-network predictions from [34] (both pre-trained on MPII and fine-tuned on Human3.6M), Detectron and CPN (both pre-trained on COCO and fine-tuned on Human3.6M). Both Mask R-CNN and CPN give better performance than the stacked hourglass network. The improvement is likely to be due to the higher heatmap resolution, stronger feature combination (*feature pyramid network* [31, 44] for Mask R-CNN and *RefineNet* for CPN), and the more diverse dataset on which they are pretrained, *i.e*. COCO [32]. When trained on 2D ground-truth poses, our model improves the lower bound of [34] by 8.3 mm, and the LSTM-based approach of Lee *et al*. [27] by 1.2 mm for protocol 1. Therefore, our improvements are not merely due to a better 2D detector.

Absolute position errors do not measure the smoothness of predictions over time, which is important for video. To evaluate this, we measure joint velocity errors (MPJVE), corresponding to the MPJPE of the first derivative of the 3D pose sequences. Table 2 shows that our temporal model

[3] All subsequent results for [16] in this paper were computed by us using their public implementation.

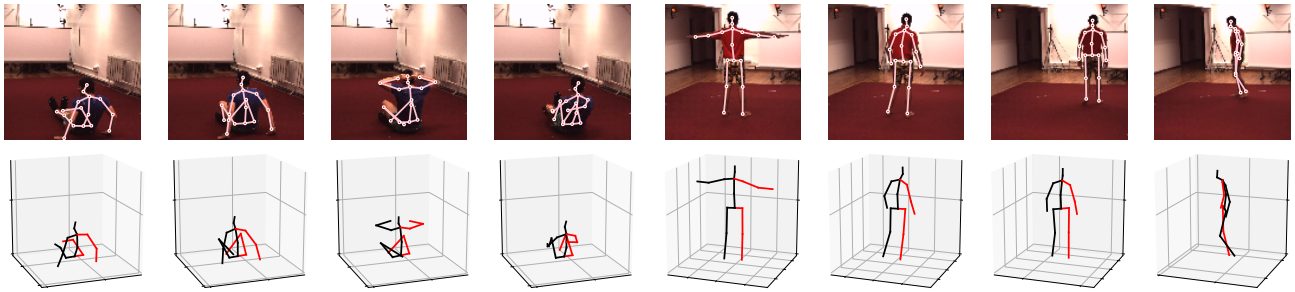


Figure 4: Qualitative results for two videos. **Top:** video frames with 2D pose overlay. **Bottom:** 3D reconstruction.

	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	Avg
Single-frame	12.8	12.6	10.3	14.2	10.2	11.3	11.8	11.3	8.2	10.2	10.3	11.3	13.1	13.4	12.9	11.6
Temporal	3.0	3.1	2.2	3.4	2.3	2.7	2.7	3.1	2.1	2.9	2.3	2.4	3.7	3.1	2.8	2.8

Table 2: Velocity error over the 3D poses generated by a convolutional model that considers time and a single-frame baseline.

Method	P1	P2	Method	P1	P2
Martinez *et al.* [34] (GT)	45.5	37.1	Ours (GT)	37.2	27.2
Martinez *et al.* [34] (SH PT)	67.5	52.5	Ours (SH PT from [34])	58.6	45.0
Martinez *et al.* [34] (SH FT)	62.9	47.7	Ours (SH FT from [34])	53.4	40.1
Hossain & Little [16] (GT)	41.6	31.7	Ours (D PT)	54.8	42.0
Lee *et al.* [27] (GT)	38.4	–	Ours (D FT)	51.6	40.3
Ours (CPN PT)	52.1	40.1	Ours (CPN FT)	46.8	36.5

Table 3: Effect of the 2D detector on the final result, under Protocol 1 (P1) and Protocol 2 (P2) **Legend:** ground-truth (GT), stacked hourglass (SH), Detectron (D), cascaded pyramid network (CPN), pre-trained (PT), fine-tuned (FT).

	Walk			Jog			Box		
	S1	S2	S3	S1	S2	S3	S1	S2	S3
Pavlakos *et al.* [41] (MA)	22.3	19.5	29.7	28.9	21.9	23.8	–	–	–
Martinez *et al.* [34] (SA)	19.7	17.4	46.8	26.9	18.2	18.6	–	–	–
Pavlakos *et al.* [40] (+) (MA)	18.8	12.7	**29.2**	23.5	15.4	14.5	–	–	–
Lee *et al.* [27] (MA)	18.6	19.9	30.5	25.7	16.8	17.7	42.8	48.1	53.4
Ours (SA)	14.5	10.5	47.3	21.9	13.4	13.9	24.3	34.9	32.1
Ours (MA)	**13.9**	**10.2**	46.6	**20.9**	**13.1**	**13.8**	**23.8**	**33.7**	**32.0**

Table 4: Error on HumanEva-I under Protocol 2 for single-action (SA) and multi-action (MA) models. Best in bold, second best underlined. (+) uses extra data. The high error on "Walk" of S3 is due to corrupted mocap data.

reduces the MPJVE of the single-frame baseline by 76% on average resulting in vastly smoother poses.

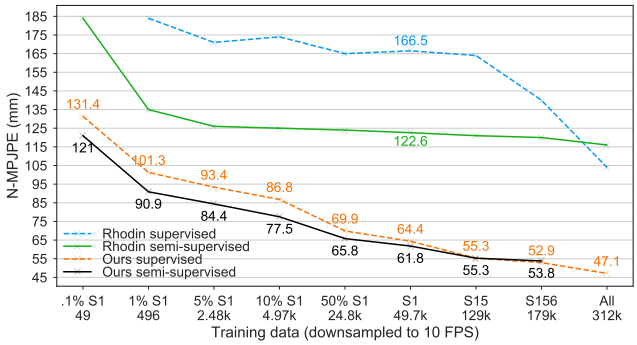
Table 4 shows results on HumanEva-I and that our model generalizes to smaller datasets; results are based on pre-trained Mask R-CNN 2D detections. Our models outperform the previous state-of-the-art.

Finally, Table 5 compares the convolutional model to the LSTM model of [16] in terms of complexity. We report the number of model parameters and an estimate of the floating-point operations (FLOPs) to predict one frame at inference time (details in Appendix A.2). For the latter, we only consider matrix multiplications and report the amortized cost over a hypothetical sequence of infinite length (to disregard padding). MPJPE results are based on models trained on ground-truth 2D poses without test-time augmentation. Our model achieves a significantly lower error even when the number of computations are halved. Our largest model with receptive field of 243 frames has roughly the same complexity as [16], but at 3.8 mm lower error. The table also highlights the effectiveness of dilated convolutions which increase complexity only logarithmically with respect to the receptive field.

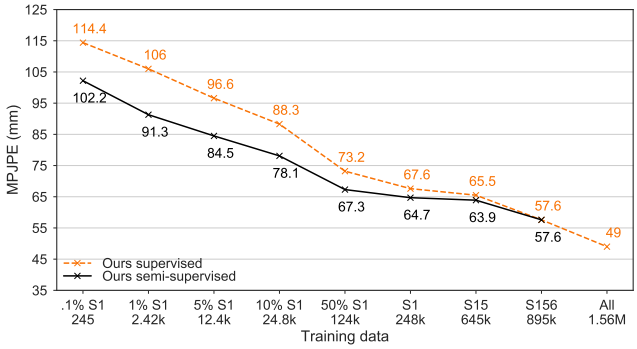
Model	Parameters	≈ FLOPs	MPJPE
Hossain & Little [16]	16.96M	33.88M	41.6
Ours 27f w/o dilation	29.53M	59.03M	41.1
Ours 27f	8.56M	17.09M	40.6
Ours 81f	12.75M	25.48M	38.7
Ours 243f	16.95M	33.87M	37.8

Table 5: Computational complexity of various models under *Protocol 1* trained on ground-truth 2D poses. Results are without test-time augmentation.

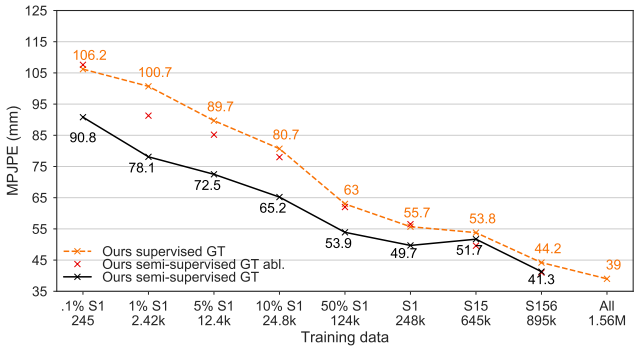
Since our model is convolutional, it can be parallelized both over the number of sequences as well as over the temporal dimension. This contrasts to RNNs, which can only be parallelized over different sequences and are thus much less efficient for small batch sizes. For inference, we measured about 150k FPS on a single NVIDIA GP100 GPU over a single long sequence, i.e., batch size one, assuming that 2D poses were already available. Speed is largely independent of the batch size due to parallel temporal processing.



(a) Downsampled to 10 FPS under Protocol 3.



(b) Full framerate under Protocol 1.



(c) Full framerate under Protocol 1 with ground-truth 2D poses.

Figure 5: **Top:** comparison with [45] on *Protocol 3*, using a downsampled version of the dataset for consistency. **Middle:** our method under *Protocol 1* (full frame rate). **Bottom:** our method under *Protocol 1* when trained on ground-truth 2D poses (full frame rate). The small crosses ("abl." series) denote the ablation of the bone length term.

6.2. Semi-supervised approach

We adopt the setup of [45] who consider various subsets of the Human3.6M training set as labeled data and the remaining samples are used as unlabeled data. Their setup also generally downsamples all data to 10 FPS (from 50 FPS). Labeled subsets are created by first reducing the number of subjects and then by downsampling Subject 1.

Since the dataset is downsampled, we use a receptive field of 9 frames, equivalent to 45 frames upsampled. For the very small subsets, 1% and 5% of S1, we use 3 frames, and we use a single-frame model for 0.1% of S1 where only 49 frames are available. We fine-tuned CPN on the labeled data only and warm up training by iterating only over labeled data for a few epochs (1 epoch for $\geq$ S1, 20 epochs for smaller subsets).

Figure 5a shows that our semi-supervised approach becomes more effective as the amount of labeled data decreases. For settings with less than 5K labeled frames, our approach achieves improvements of about 9-10.4 mm N-MPJPE over our supervised baseline. Our supervised baseline is much stronger than [45] and outperforms all of their results by a large margin. Although [45] uses a single-frame model in all experiments, our findings still hold on 0.1% of S1 (where we also use a single-frame model).

Figure 5b shows results for our method under the more common Protocol 1 for the non-downsampled version of the dataset (50 FPS). This setup is more appropriate for our approach since it allows us to exploit full temporal information in videos. Here we use a receptive field of 27 frames, except in 1% of S1, where we use 9 frames, and 0.1% of S1, where we use one frame. Our semi-supervised approach gains up to 14.7 mm MPJPE over the supervised baseline.

Figure 5c switches the CPN 2D keypoints for ground-truth 2D poses to measure if we could perform better with a better 2D keypoint detector. In this case, improvements can be up to 22.6 mm MPJPE (1% of S1) which confirms that better 2D detections could improve performance. The same graph shows that the bone length term is crucial for predicting valid poses, since it forces the model to respect kinematic constraints (line "Ours semi-supervised GT abl."). Removing this term drastically decreases the effectiveness of semi-supervised training: for 1% of S1 the error increases from 78.1 mm to 91.3 mm which compares to 100.7 mm for the supervised baseline.

7. Conclusion

We have introduced a simple fully convolutional model for 3D human pose estimation in video. Our architecture exploits temporal information with dilated convolutions over 2D keypoint trajectories. A second contribution of this work is back-projection, a semi-supervised training method to improve performance when labeled data is scarce. The method works with unlabeled video and only requires intrinsic camera parameters, making it practical in scenarios where motion capture is challenging (*e.g.* outdoor sports).

Our fully convolutional architecture improves the previous best result on the popular Human3.6M dataset by 6mm average joint error which corresponds to a relative reduction of 11% and also shows improvements on HumanEva-I. Back-projection can improve 3D pose estimation accuracy by about 10mm N-MPJPE (15mm MPJPE) over a strong baseline when 5K or fewer annotated frames are available.

References

[1] 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 *European Conference on Computer Vision (ECCV)*, pages 561–578. Springer, 2016. 2

[2] E. Brau and H. Jiang. 3d human pose estimation via deep learning from 2d annotations. In *International Conference on 3D Vision (3DV)*, pages 582–591. IEEE, 2016. 2

[3] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997. 2

[4] C.-H. Chen and D. Ramanan. 3D human pose estimation = 2D pose estimation + matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5759–5767, 2017. 2

[5] Y. Chen, Z. Wang, Y. Peng, and Z. Zhang. Cascaded pyramid network for multi-person pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3, 5

[6] R. Collobert, C. Puhrsch, and G. Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*, 2016. 1

[7] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *International Conference on Machine Learning (ICML)*, 2017. 1

[8] D. Drover, R. M. V, C.-H. Chen, A. Agrawal, A. Tyagi, and C. P. Huynh. Can 3d pose be learned from 2d projections alone? In *European Conference on Computer Vision Workshops (ECCVW)*, pages 78–94. Springer, 2018. 2

[9] S. Edunov, M. Ott, M. Auli, and D. Grangier. Understanding back-translation at scale. In *Proc. of EMNLP*, 2018. 1

[10] H. Fang, Y. Xu, W. Wang, X. Liu, and S.-C. Zhu. Learning pose grammar to encode human body configuration for 3d pose estimation. In *AAAI*, 2018. 1, 4, 6

[11] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*, 2017. 1

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *International Conference on Computer Vision (ICCV)*, pages 2980–2988. IEEE, 2017. 3, 5

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 3

[14] E. Hoffer, R. Banner, I. Golan, and D. Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. *arXiv preprint arXiv:1803.01814*, 2018. 12

[15] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. *Wavelets, Time-Frequency Methods and Phase Space*, -1:286, 01 1989. 3

[16] M. R. I. Hossain and J. J. Little. Exploiting temporal information for 3d pose estimation. In *European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 4, 6, 7, 11

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015. 3, 5

[18] C. Ionescu, J. Carreira, and C. Sminchisescu. Iterated second-order label sensitive pooling for 3d human pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1661–1668, 2014. 2

[19] C. Ionescu, F. Li, and C. Sminchisescu. Latent structured models for human pose estimation. In *International Conference on Computer Vision (ICCV)*, 2011. 4

[20] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3D human sensing in natural environments. *Transaction on Pattern Analysis and Machine Intelligence (TPAMI)*, 2014. 2, 4, 5

[21] H. Jiang. 3d human pose reconstruction using millions of exemplars. In *International Conference on Pattern Recognition (ICPR)*, pages 1674–1677. IEEE, 2010. 2

[22] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu. Neural machine translation in linear time. *arXiv*, abs/1610.10099, 2016. 3

[23] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7122–7131, 2018. 2

[24] I. Katircioglu, B. Tekin, M. Salzmann, V. Lepetit, and P. Fua. Learning latent representations of 3d human pose with deep neural networks. *International Journal of Computer Vision (IJCV)*, pages 1–16, 2018. 2

[25] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations (ICLR)*, 2017. 12

[26] G. Lample, A. Conneau, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR)*, 2018. 1

[27] K. Lee, I. Lee, and S. Lee. Propagating LSTM: 3d pose estimation based on joint interdependency. In *European Conference on Computer Vision (ECCV)*, pages 119–135, 2018. 1, 2, 4, 5, 6, 7

[28] S. Li and A. B. Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *Asian Conference on Computer Vision (ACCV)*, pages 332–347. Springer, 2014. 2

[29] S. Li, W. Zhang, and A. B. Chan. Maximum-margin structured learning with deep networks for 3d human pose estimation. In *International Conference on Computer Vision (ICCV)*, pages 2848–2856, 2015. 4, 5

[30] M. Lin, L. Lin, X. Liang, K. Wang, and H. Cheng. Recurrent 3d pose sequence machines. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 810–819, 2017. 2

[31] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. 5, 6

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision (ECCV)*, pages 740–755. Springer, 2014. 5, 6

[33] D. C. Luvizon, D. Picard, and H. Tabia. 2d/3d pose estimation and action recognition using multitask deep learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2018. 1, 2, 4, 5, 6

[34] J. Martinez, R. Hossain, J. Romero, and J. J. Little. A simple yet effective baseline for 3d human pose estimation. In *International Conference on Computer Vision (ICCV)*, pages 2659–2668, 2017. 1, 2, 4, 5, 6, 7, 12

[35] 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 (3DV)*, pages 506–516. IEEE, 2017. 2

[36] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):44, 2017. 2

[37] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814, 2010. 3

[38] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016. 3, 4

[39] S. Park, J. Hwang, and N. Kwak. 3d human pose estimation using convolutional neural networks with 2d pose information. In *European Conference on Computer Vision (ECCV)*, pages 156–169. Springer, 2016. 2

[40] G. Pavlakos, X. Zhou, and K. Daniilidis. Ordinal depth supervision for 3d human pose estimation. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 4, 6, 7

[41] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1263–1272. IEEE, 2017. 1, 2, 4, 5, 6, 7

[42] V. Ramakrishna, T. Kanade, and Y. Sheikh. Reconstructing 3d human pose from 2d image landmarks. In *European Conference on Computer Vision (ECCV)*, pages 573–586. Springer, 2012. 2

[43] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations (ICLR)*, 2018. 5

[44] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015. 6

[45] H. Rhodin, M. Salzmann, and P. Fua. Unsupervised geometry-aware representation for 3D human pose estimation. In *European Conference on Computer Vision (ECCV)*, 2018. 2, 4, 8

[46] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proc. of ACL*, 2016. 1

[47] L. Sigal, A. O. Balan, and M. J. Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision (IJCV)*, 87(1-2):4, 2010. 4

[48] C. Sminchisescu. 3d human motion analysis in monocular video: techniques and challenges. In *Human Motion*, pages 185–211. Springer, 2008. 2

[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 3

[50] X. Sun, J. Shang, S. Liang, and Y. Wei. Compositional human pose regression. In *International Conference on Computer Vision (ICCV)*, pages 2621–2630, 2017. 1, 4, 6

[51] B. Tekin, I. Katircioglu, M. Salzmann, V. Lepetit, and P. Fua. Structured prediction of 3d human pose with deep neural networks. In *British Machine Vision Conference (BMVC)*, 2016. 2

[52] B. Tekin, P. Marquez Neila, M. Salzmann, and P. Fua. Learning to fuse 2d and 3d image cues for monocular body pose estimation. In *International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 4, 6

[53] B. Tekin, A. Rozantsev, V. Lepetit, and P. Fua. Direct prediction of 3d body poses from motion compensated sequences. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 991–1000, 2016. 2, 4, 5

[54] H.-Y. F. Tung, A. W. Harley, W. Seto, and K. Fragkiadaki. Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. In *International Conference on Computer Vision (ICCV)*, pages 4364–4372. IEEE, 2017. 2

[55] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. 1, 3

[56] W. Yang, W. Ouyang, X. Wang, J. Ren, H. Li, and X. Wang. 3d human pose estimation in the wild by adversarial learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2018. 1, 2, 4, 6

[57] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016. 3

[58] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei. Towards 3d human pose estimation in the wild: a weakly-supervised approach. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 4

[59] X. Zhou, M. Zhu, S. Leonardos, K. G. Derpanis, and K. Daniilidis. Sparseness meets deepness: 3d human pose estimation from monocular video. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4966–4975, 2016. 4, 5

A. Supplementary material

A.1. Dilated convolutions and information flow

Dilated convolutions are a particular form of convolution with a sparse structure, whose kernel points are spaced uniformly and filled with zeros in between. For instance, the discrete filter $h = [\ 1\ \boxed{2}\ 3\ ]$ (where $\boxed{2}$ is the center) becomes $[\ 1\ 0\ \boxed{2}\ 0\ 3\ ]$ with dilation factor $D = 2$, and $[\ 1\ 0\ 0\ \boxed{2}\ 0\ 0\ 3\ ]$ with $D = 3$. This particular structure enables optimized implementations that skip computations over zero points.

Consider a discrete convolution of two signals f (of length N) and h (zero-centered, of length $2M + 1$), which can be computed as

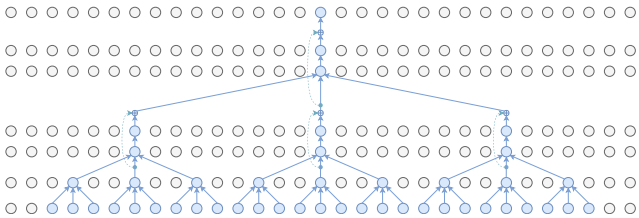
$$(f * h)[n] = \sum_{m=-M}^{M} f[n - m]\, h[m] \tag{2}$$

Instead of spacing the kernel explicitly and applying a regular convolution, a dilated convolution can be computed as

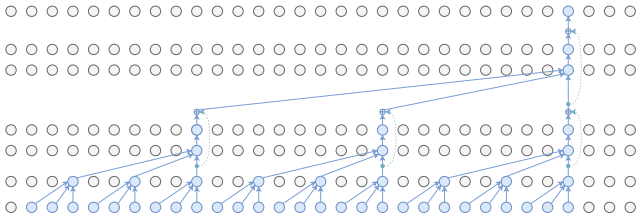
$$(f * h_D)[n] = \sum_{m=-M}^{M} f[D(n - m)]\, h[m] \tag{3}$$

yielding roughly the same computational cost as regular convolutions for the same number of non-zero entries, while increasing the receptive field.

For illustration purposes, in Figure 6 we depict the information flow in our models. We also highlight the difference between symmetric convolutions and causal convolutions.



(a) Symmetric convolutions.



(b) Causal convolutions.

Figure 6: Information flow in our models, from input (bottom layer) to output (top layer). Dashed lines represent skip-connections.

A.2. Computational cost analysis

In this section we show how we computed the computational complexity of our model and that of [16]. The common practice is to consider only matrix multiplications, as other operations (*e.g.* biases, batch normalization, activations) have negligible impact on the final complexity. For [16], we evaluated its reference implementation in TensorFlow. We computed the amortized cost to predict one frame using the TensorFlow profiler, and only counted operations corresponding to matrix multiplications. According to TensorFlow's approximation, multiplying a $N \times M$ matrix by a $M \times K$ matrix has a cost of $2\,NMK$ FLOPs (floating-point operations), which is equivalent to NMK multiply-add operations.

For our model, we adopted the same convention. We provide a sample cost analysis for a model with a receptive field of 27 frames, which consists of 2 residual blocks. Since the matrix multiplications in our model are only due to convolutions, the analysis is straightforward and can be computed by hand.

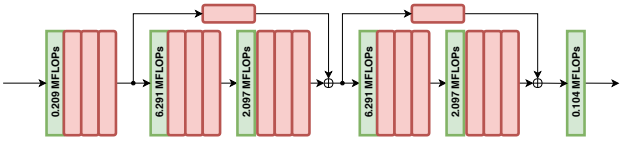


Figure 7: Architecture of a model with a receptive field of 27 frames, with the corresponding amortized cost to predict one frame in convolutional layers.

As can be seen in Figure 7, the model consists of 6 convolutional layers. Disregarding padding (*i.e.* for sequences of length $N \gg 0$), performing a 1D convolution with C_{in} input channels, C_{out} output channels, and width W has a cost of $2\,N\,W\,C_{in}\,C_{out}$ FLOPs, *i.e.* $2\,W\,C_{in}\,C_{out}$ FLOPs per frame. In our 27-frame model, the results can be summarized as follows:

1. $W = 3$, channels $17 \cdot 2 \rightarrow 1024$, cost 0.209 MFLOPs.
2. $W = 3$, channels $1024 \rightarrow 1024$, cost 6.291 MFLOPs.
3. $W = 1$, channels $1024 \rightarrow 1024$, cost 2.097 MFLOPs.
4. $W = 3$, channels $1024 \rightarrow 1024$, cost 6.291 MFLOPs.
5. $W = 1$, channels $1024 \rightarrow 1024$, cost 2.097 MFLOPs.
6. $W = 1$, channels $1024 \rightarrow 17 \cdot 3$, cost 0.104 MFLOPs.

Total: 17.089 MFLOPs per frame.

A.3. Ablation of receptive field and channel size

In Figure 8b we report the test error for different receptive fields, namely 1, 9, 27, 82, and 243 frames. To this end,

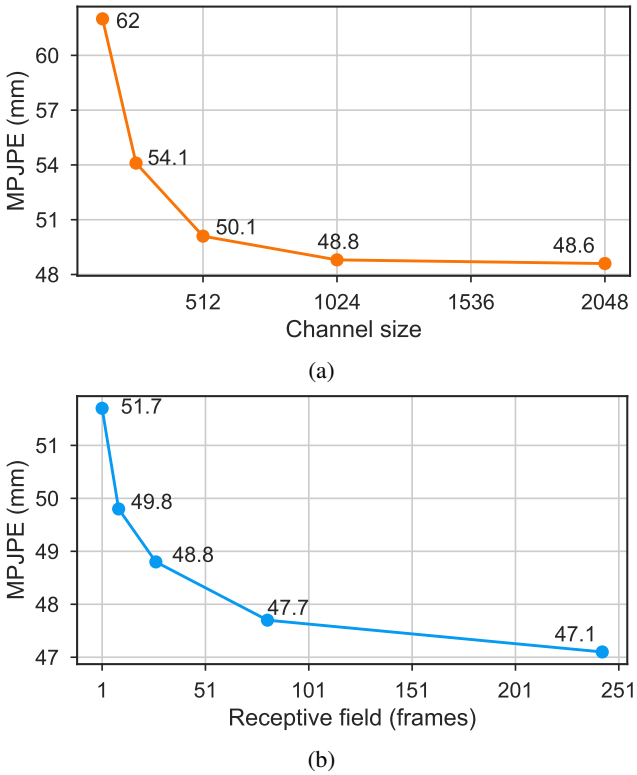


Figure 8: **Top:** Error as a function of the channel size, with a fixed receptive field of 27 frames. **Bottom:** Error as a function of the receptive field, with a fixed channel size of 1024. Fine-tuned CPN detections for both experiments.

we stack a varying number of residual blocks, each of which multiplies the receptive field by 3. In the single-frame scenario, we use 2 blocks and set the convolution widths of all layers to 1, obtaining a model functionally equivalent to [34]. As can be seen, the model does not seem to overfit as the receptive field increases. On the other hand, the error tends to saturate quickly, suggesting that the task of 3D human pose estimation does not require modeling long-term dependencies. Therefore, we generally adopt a receptive field of 243 frames. Similarly, in Figure 8a we vary the channel size between 128 and 2048, with the same findings: the model is not prone to overfitting, but the error saturates past a certain point. Since the computational complexity increases quadratically with respect to the channel size, we adopt $C = 1024$.

A.4. Data augmentation and convolution type

When we remove test-time augmentation, the error increases to 47.7 mm (from 46.8 mm) in our top-performing model. If we also remove train-time augmentation, the error reaches 49.2 mm (another +1.5 mm).

Next, we replace dilated convolutions with regular dense convolutions. In a model with a receptive field of 27 frames and fine-tuned CPN detections, the error increases from 48.8 mm to 50.4 mm (+1.6 mm), while also increasing the number of parameters and computations by a factor of ≈ 3.5. This highlights that dilated convolutions are crucial for efficiency, and that they counteract overfitting.

A.5. Batching strategy

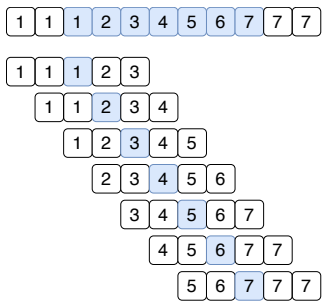
We argue that the reconstruction error is strongly dependent on how the model is trained, and we suggested to generate minibatches in a way that only one output frame at a time is predicted. To show why this is important, we introduce a new hyperparameter – the *chunk size* C (or *step size*), which specifies how many frames are predicted at once per sample. Predicting only one frame, i.e. $C = 1$, requires a full receptive field F as input. Predicting two frames ($C = 2$) requires $F + 1$ frames, and so on. It is evident that predicting multiple frames is computationally more efficient, as the results of intermediate convolutions can be shared among frames – and in fact, we do this at inference. On the other hand, we show that during training this is detrimental to generalization.

Figure 9b illustrates the reconstruction error (as well as the relative speedup in training time) when training a 27-frame model with different step sizes, namely 1, 2, 4, 8, 16, and 32 frames. Since predicting multiple frames is equivalent to increasing the batch size – thus hurting generalization [25] – we make the results comparable by adjusting the batch size so that the model always predicts 1024 frames. Therefore, the 1-frame experiment adopts a batch size of 1024 sequences, which becomes 512 for 2 frames, 256 for 4 frames, and so on. This methodology also ensures the models will be trained with the same number of weight updates.

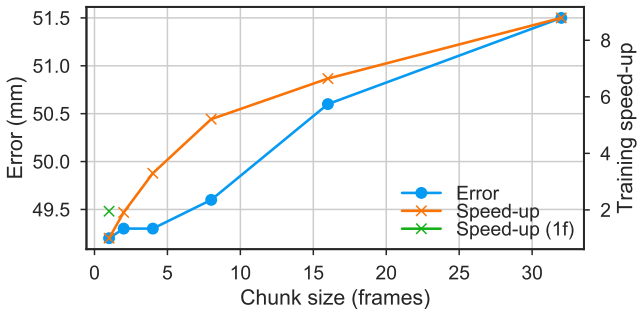
The results show that the error decreases in conjunction with the step size, at the expense of training speed. The impaired performance of the models trained with high step size is caused by correlated batch statistics [14]. Our implementation optimized for single-frame outputs achieves a speed-up factor of ≈ 2, but this gain is even higher on models with larger receptive fields (*e.g.* ≈ 4 with 81 frames), and enabled us to train the model with 243 frames.

A.6. Optimized training implementation

Figure 10 shows why our implementation tailored for single-frame predictions is important. A regular implementation computes intermediate states layer by layer. This is very efficient for long sequences, as the states computed in layer n can be reused by layer $n + 1$ without recomputing them. However, for short sequences, this approach becomes inefficient because states near boundaries are not used. In the extreme case of single-frame predictions (which we use for training), many intermediate computations are wasted,



(a)



(b)

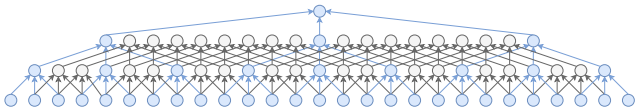
Figure 9: **Top:** batch creation process *for training*. This example shows a video of 7 frames which is used to train a model with a receptive field of 5 frames. We generate a training example for each of the 7 frames, such that only the center frame is predicted. The video is padded by replicating boundary frames. **Bottom:** reconstruction error and training speed-up with different step sizes. The speed-up is relative to $C = 1$. The `1f` variant shows the speed up corresponding to the implementation optimized for single-frame predictions.

as can be seen in Figure 10a. In this case, we replace dilated convolutions with strided convolutions, making sure to obtain a model which is functionally equivalent to the original one (*e.g.* by also adapting skip-connections). This strategy ensures that no intermediate states will be discarded.

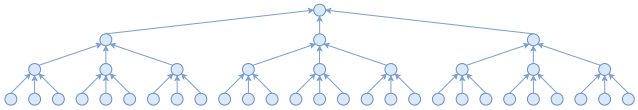
As mentioned, at inference we use the regular layer-by-layer implementation since it is more efficient for multi-frame predictions.

A.7. Demo videos

The supplementary material contains several videos highlighting the smoothness of the predictions of our temporal convolutional model compared to the single frame baseline. Specifically, we show side by side the original video sequence, poses predicted by the single-frame baseline, poses from the temporal convolutional model as well as the ground-truth poses. Some demo videos can also be found at https://dariopavllo.github.io/VideoPose3D.



(a) Layer-by-layer implementation.



(b) Implementation optimized for single-frame predictions.

Figure 10: Comparison between two implementations for a single-frame prediction, receptive field of 27 frames. In the layer-by-layer implementation many intermediate states are wasted, whereas the optimized implementation computes only required states. As the length of the sequence increases, the layer-by-layer implementation becomes more efficient.