CapsulePose: a variational CapsNet for real-time end-to-end 3D human pose estimation

Nicola Garau[a,*], Nicola Conci[a]

[a] *University of Trento, Via Sommarive, 9, 38123 Povo, Trento TN (IT)*

Abstract

Estimating 3D human poses from images is an ill-posed regression problem, which is usually tackled by viewpoint-invariant convolutional neural networks (CNNs). Recently, capsule networks (CapsNets) have been introduced as a viable alternative to CNNs, ensuring viewpoint-equivariance and drastically reducing both the dataset size and the network complexity, while retaining high output accuracy. We propose a real-time end-to-end human pose estimation (HPE) network which employs state-of-the-art matrix capsules [1] and a fast variational Bayesian capsule routing, without relying on pre-training, complex data augmentation or multiple datasets. We achieve comparable results to the HPE state-of-the-art, and the lowest error among methods using CapsNets, while at the same time achieving other desirable properties, namely greater generalization capabilities, stronger viewpoint equivariance and highly decreased data dependency, allowing for our network to be trained with only a fraction of the available datasets and without any data augmentation.

Keywords: capsule networks, 3D human pose estimation,
viewpoint-equivariance, deep learning, real-time

[*] Corresponding author

Email addresses: `nicola.garau@unitn.it` (Nicola Garau), `nicola.conci@unitn.it` (Nicola Conci)

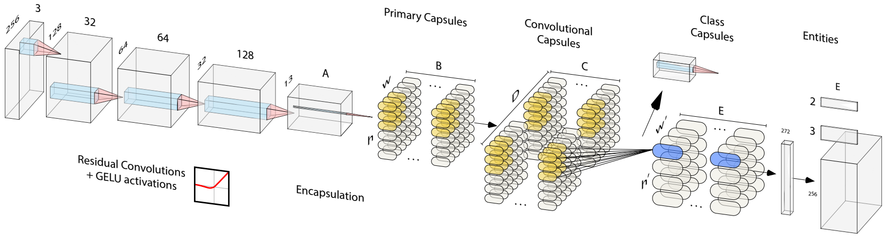


Figure 1: CapsulePose's network architecture. From left to right: Encoder with residual convolutions; Capsule layers with matrix capsules (primary, convolutional and class capsules) and fast variational Bayes routing; Decoder with multi-task loss and interpretable capsule latent space.

1. Introduction

The literature on human pose estimation (HPE) has received an increasing number of contributions with applications to many different domains, including motion capture and tracking [2], virtual and augmented reality [3], robotics [4], sports analysis [5], camera calibration [6, 7], activity recognition [8] and ambient-assisted living [9, 10], to name a few. Some crucial aspects that concern real-world applications are reliability, generalization, and real-time compliance of the employed architectures. However, it is known that, in order to maximize performance in many deep learning applications, including HPE, a huge amount of labelled data is required. Moreover, annotating 3D human pose data is not feasible without specific expensive equipment, such as 3D motion capture systems; since generalization greatly depends on the quantity and quality of data, HPE networks have grown in complexity, in proportion to the growth of the datasets. To further push the accuracy of the estimation, many methods take into consideration the usage of multiple datasets, which often rely on different joint labelling, non-standard data augmentation, biometric models and the usage of 2D ground truth during training or even testing. However, in real-world applications, 2D ground truth data is not always available, and estimating 2D data to use it as a pseudo-ground truth is time-consuming and leads to error propagation. For this reason, end-to-end methods are usually preferable and more reliable than two-stage methods relying on 2D estimations. Nonetheless, still today, end-to-end 3D human pose estimation architectures are very complex and hard to train, especially when using a single dataset, and they usually rely on additional data and non-standard data augmentation in order to reach state-of-the-art performance. Additionally, there are other factors to be considered when dealing with real-world applications, such as, for example, crowding and unconventional camera positions or occlusions. These factors make generalization on unseen data harder, thus requiring more complex networks and increasing the risk of overfitting.

A novel learning system has been presented in recent literature by Hinton et al. [1]. Capsule networks (CapsNets) address some of the issues of traditional convolutional neural networks (CNNs), such as poor information routing through *max-pooling* and the limits of *scalar activation values* for generalization, by replacing them with *routing-by-agreement algorithms* and *capsule activation values*, respectively. CapNets have already shown excellent results on simple datasets, such as MNIST, smallNORB and CIFAR-10, proving to have superior generalization capabilities across unseen viewpoints and better interpretability, thanks to the embedded inverse graphics capabilities, while at the same time requiring significantly fewer parameters compared to traditional CNNs. All these qualities elicited a growing interest in the research community. However, despite their major advantages, CapsNets are not a popular choice when dealing with high-resolution datasets, because of the longer training times of the capsule routing algorithms.

In the long run, what we would like to achieve is to train neural networks which can jointly satisfy the following conditions:

- Simple network architecture design
- Small number of trainable parameters
- High generalization to different datasets and viewpoints

- Easy upgradability in terms of architecture
- Fast, real-time inference performance

We believe that capsule-like architectures are a good candidate to respect these constraints and greatly improve human pose estimation, at least in the long run. We present the first neural network based on matrix capsules that tackles the problem of view-independent human pose estimation. Compared to other state-of-the-art solutions, our network has a simple architecture with a very small number of parameters. CapsulePose obtains good results on benchmarks while training only on one dataset, with no data augmentation or additional training-time tricks such as learning rate scheduling and warm-up. Additionally, we achieve very fast inference, which is a welcome requirement for real-time applications. With this paper, our aim is to propose a simple yet effective baseline for human pose estimation using capsules, encouraging the development of better and more accurate methods using matrix capsules.

The contribution of this paper is manifold: we propose a simple yet effective baseline network for end-to-end and real-time 3D human pose estimation. To our knowledge, it is the first architecture to employ state-of-the-art matrix capsules with 4×4 3D pose matrices. We achieve state-of-the-art performance on the Human3.6M dataset without employing additional data or non-standard data augmentation. We employ a Variational Bayes (VB) capsule routing paired with an optimized and modular codebase, to minimize both training and testing times. We consider two of the most important novelties of this work to be the embedding of viewpoint-equivariance as frames of reference for greater generalization capabilities and the unprecedented usage of matrix capsules and VB routing for HPE. During testing, our network is almost twice as fast as other capsule-based architectures[1].

The paper is organized as follows: in Section 2 we propose an overview of

[1] The code, dataset and pre-trained models will be made available for fair comparison and replicability upon acceptance

the state-of-the-art for both capsule networks and HPE; in Section 3 we dissect our proposed architecture down to its core, commenting on the design choices and their reasons; finally, in Section 4, we discuss our experiments, showing quantitative and qualitative results, commenting on the improved generalization capabilities of the proposed architecture.

2. Related work

In this section, we will explore the existing literature for both capsule networks and 3D human pose estimation.

2.1. Capsule networks

Capsule networks have been proposed in literature with the purpose of modelling a system capable of learning part-whole relationships between so-called *entities* across different viewpoints, similarly to how our visual cortex system operates, according to the recognition-by-components theory [11].This problem is also known as the viewpoint invariance problem, namely, how the network activations change with the change of the viewpoint, usually after a transformation (translation, scaling, rotation, shearing). CNNs' scalar activations are not suited to efficiently manage these kinds of viewpoint transformations, thus needing to often rely on max-pooling and aggressive data augmentation. However, by doing so, CNNs achieve viewpoint-invariance, meaning that a slight modification of the input image would lead to the same activation value. Hinton et al. in [12] extensively discuss the drawbacks and issues of CNNs and how to address them using capsules. A more desirable property would be to capture and retain the transformation applied to the input image, in order for the network activations to be aware of the different transformations applied to the input. Being able to model network activations that change in a structured way according to the input viewpoint transformations is also called viewpoint-equivariance (Eq. 2), which is what CapsNets propose to achieve. In viewpoint-invariance (Eq. 1), a viewpoint transformation T does not change the outcome of the network activations. On the other hand, in viewpoint-equivariance (Eq. 2), the network activations change according to the applied viewpoint transformation T. Unlike traditional CNNs, which usually retain viewpoint-invariance (Eq. 1), capsule networks can explicitly model and jointly preserve a viewpoint transformation T through the network activations, drastically reducing the number of trainable parameters, depending on the application.

$$f(Tx) = f(x) \tag{1}$$

$$f(Tx) = Tf(x) \tag{2}$$

This is achieved by introducing the concept of *capsules*: groups of neurons, which explicitly encode the intrinsic viewpoint-invariant relationship that exists between different parts of the same object. As of today, three official capsule network iterations have been presented [13, 1, 14]. The first one, by Sabour et al., introduces for the first time a routing algorithm for vector capsules (Fig. 2), called *routing-by-agreement* as a better max-pooling substitute, achieving very promising results on simple datasets such as MNIST, CIFAR10 and smallNORB.

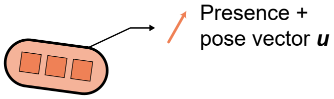


Figure 2: The first iteration of a capsule's structure (vector capsules), suitable for dynamic-like routing, as described in [13]. Classic CNNs scalar-output feature detectors are replaced by vector-output capsules. Each capsule contains a single vector which describes both the pose and the presence of an entity.

The second official iteration of capsule networks [1], by Hinton et al., further improves accuracy, reducing the number of test errors on the smallNORB dataset by 45%. This is achieved through a more complex capsule structure (Fig. 3) and an Expectation-Maximization routing (*EM-routing*) for capsules. Unfortunately, the EM-routing and the 4×4 pose matrix embedded in the capsule contribute to increasing the training time, when compared to both CNNs and [13].

The third official capsule network [14] by Kosiorek et al., constitutes a big

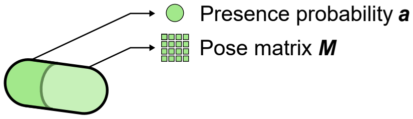


Figure 3: The second iteration of a capsule's structure (matrix capsules) is suitable for expectation-maximization-like routing, as described in [1]. It contains both a scalar value and a 4X4 matrix, respectively describing the presence probability and a more robust 3D pose compared to the first capsule iteration.

leap forward, introducing for the first time an unsupervised capsule-based autoencoder. Ribeiro et al. in [15] build up on the EM-routing version of capsules by proposing for the first time a VB capsule routing for fitting a mixture of transforming Gaussians. They present state-of-the-art results on smallNORB by using $\sim 50\%$ fewer capsules, thus unlocking additional possibilities both in terms of performance gain and network complexity reduction. Recent works in literature explore the possibility of further reducing capsule network complexity through quaternions [16] while increasing performance as well. However, all the mentioned works only consider small datasets for benchmarking.

2.2. Human pose estimation

Estimating the human pose from images can be seen as an ill-posed regression problem. Once semi-parametric [17] and model-based or, learning-based human pose estimation (HPE) methods have recently gained a lot of interest in the research community, particularly real-time 2D HPE approaches [18], and only recently 3D HPE and human mesh recovery (HMR) approaches.

3D HPE usually relies on additional cues, such as 2D predictions [19, 20, 21], multiple images [22], pre-trained autoencoders [23] and pose dictionaries [24]. Other recent works aim at end-to-end, learning-based 3D HPE [25, 26, 27] or at designing architectures that make better use of multi-view data [28].

Bogo et al. in [29] considered the possibility of improving the human pose modelling via skinned multi-person linear models (SMPL), and in literature end-to-end 3D HMR approaches can be found [30, 31, 32]. Among the most recent HPE developments, many good works have been focusing on video mesh recovery [33], fast on-device inference [34] and multi-camera approaches without 3D supervision [35]. To our knowledge, only a recent work from Ramírez [36] tackles the problem of using capsule networks to solve the ill-posed 3D HPE problem in an end-to-end fashion. They propose a Bayesian formulation of the original version of the capsule network (with dynamic routing, as described in [13]) and benchmark it for the ill-posed problem of 3D HPE from single images, obtaining very promising results and showing how CapsNets can be used even with complex, big datasets. In this paper, we adhere to certain choices adopted by [36], in order to allow fair comparisons. In particular, we both train on the same dataset (Human3.6M dataset) and with the same training-testing protocol (protocol #1). Moreover, we even keep our loss calculation as similar as possible. Nonetheless, the network we present is fairly different to what is described in [36], namely that we employ a different capsule paradigm (matrix capsules [1] instead of dynamic capsules [13]), and thus a different capsule routing algorithm, that can work with matrix capsules, a renewed encoding-decoding pipeline with GELU activations and in general a completely redesigned network architecture. In a recent work [37] similar to the one presented here, we show how to use matrix capsules to generalize to unseen viewpoints at testing time, namely the top viewpoint, both from depth and RGB input images. In this work we instead focus on building a simple yet robust baseline for human pose estimation using capsules, which offers good inference-time speed and that can be extended to multiple datasets and scenarios.

3. Proposed architecture

We present a novel capsule network architecture with Variational Bayes routing [15] for real-time end-to-end 3D human pose estimation from a single image. We aim at keeping a simple network structure, which can be summed up into three main blocks:

1. **Encoding**: residual convolutional encoder with GELU activations [38].

2. **Encapsulation**: matrix capsules (primary, convolutional, class capsules) with VB routing [15].
3. **Decoding**: custom DDGELU decoding with transposed convolutions and multi-task self-balanced loss.

For training and evaluation, we rely on the Human3.6M dataset [2], showing that capsule networks should be able to correctly learn 3D pose representations without the need for multiple datasets. As for the optimizer, we conducted some tests with both the Adam [39] and the improved AdamW [40] optimizer with decoupled weight decay regularization. We observed that the latter provides faster overall loss convergence and comparable performance when training with a learning rate of 1e−10, a weight decay value of 1e−2 and a batch size of 32. An overview of the proposed architecture can be seen in Fig. 1 and a more detailed layer-wise pipeline is shown in Algorithm 1.

3.1. Initialization

We start by normal-initializing all the loss register buffers s_{3D}, s_{2D}, s_H, s_W to 1 and every convolutional, capsule and dense weight w_c with values sampled from $\mathcal{U}(-\alpha, \alpha)$, where

$$\alpha = gain \times \sqrt{\frac{6}{n_i + n_{i+1}}} \tag{3}$$

according to the Xavier initialization [41]. This kind of initialization was demonstrated to achieve quicker convergence and higher accuracy on CIFAR10 [41]. We define the capsule pose matrix size $P = 4$ and assign capsule parameters $[A, B, C, D, E, F]$ the values $[64, 8, 16, 16, 17, 13]$. The number of input channels for the primary capsules is defined by A, while B, C, D are the numbers of output channels for primary, convolutional (first and second) capsules respectively; E defines the number of classes, which in our case is the number of joints. Finally, F defines the size of the feature space during encapsulation.

Algorithm 1: CapsulePose network overview: from RGB images to 2D, 3D human poses and $E = 17$ joints heatmaps

```
**CapsulePose** $(x)$
    **inputs** : A batch $x = x_0 \dots x_{BS}$, $\#BS$ = batch size of
                 bounding-box RGB images containing a person
    **outputs:** $\hat{y}_{3D} \in R^{BS \times E \times 3}$,
                 $\hat{y}_{2D} \in R^{BS \times E \times 2}$,
                 $\hat{y}_{H} \in R^{BS \times E \times 256 \times 256}$
    $s_{3D}, s_{2D}, s_H, s_W \leftarrow 1$;
    $w_c \leftarrow xavier_{uniform}() \quad \forall c \in ConvLayers$;
    **foreach** $i \in ConvLayers$ **do**
        $x \leftarrow Conv2d_i(x) + Residual_i(x)$;
        $x \leftarrow \mathrm{GELU}(x)$;
        $x \leftarrow InstanceNorm2d_i(x)$;
        $x \leftarrow Dropout_{0.3}(x)$;
    $a, x \leftarrow PrimaryCapsules2d(x)$;
    **foreach** $j \in ConvCapsuleLayers$ **do**
        $a, x \leftarrow ConvCapsules2d_j(a, x)$;
        $a, x \leftarrow VBRouting2d_j(a, x)$;
    $a, x, \hat{y}_W \leftarrow ClassCapsules(a, x)$;
    $a, x \leftarrow ClassRouting(a, x)$;
    $x \leftarrow Entities(x)$;
    $\hat{y}_{3D} \leftarrow tanh(\mathrm{DDGELU}_{0.3}(\mathrm{DDGELU}_{0.3}(x)))$;
    $\hat{y}_{2D} \leftarrow sigm(\mathrm{DDGELU}_{0.3}(\mathrm{DDGELU}_{0.3}(x)))$;
    $\hat{y}_{H} \leftarrow ReLU6(\mathrm{DDGELU}_{0.3}(\mathrm{DDGELU}_{0.3}(x)))$;
    $\hat{y}_{H} \leftarrow ConvTranspose2d(reshape(\hat{y}_H))$;
    **return** $[\hat{y}_{3D}, \hat{y}_{2D}, \hat{y}_H, \hat{y}_W]$;
```

3.2. Encoding

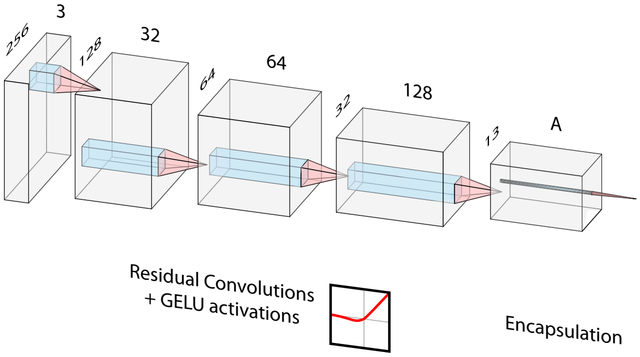


Figure 4: CNN encoder with residual convolutions and GELU [38] activations. The input image (256X256X3) is transformed by 4 convolutional layers into 32-channels, 64-channels, 128-channels and A-channels embeddings. After the convolutions, the dimensionality of the embedding is defined by A.

For the first block, we accept as input $BS \times 3 \times 256 \times 256$ (BS=batch size) previously-cropped RGB images and sequentially apply 4 convolutional steps to obtain a $BS \times A \times F \times F$ feature space, which will be eventually converted into matrix capsules. Each convolutional layer is composed of residual convolutions, to prevent vanishing gradients, as detailed in [42]. As for the activation function, we employ the Gaussian Error Linear Unit (GELU, Eq. 4) [38].

$$\begin{aligned} \mathrm{GELU}(x) &= xP(X \leq x) \\ &= x\phi(x) \\ &\approx 0.5x(1 + \tanh \sqrt{2/\pi}(x + 0.044715x^3)) \end{aligned} \tag{4}$$

GELU activation functions have been used in many recent Transformer networks and have been proven to perform better in many learning tasks, including computer vision.

As for normalization, we adopted a combination of instance normalization followed by a dropout layer with rate 0.3. We chose instance normalization over batch normalization because the latter combined with dropout may lead to anomalous behaviour during training. On the other hand, dropout can serve as additional normalization, as well as a means to capture the model uncertainty, as detailed in [43]. Each convolutional layer has a kernel of size 9×9 and stride 2, except the last one which has stride 3 and additional padding.

3.3. Encapsulation

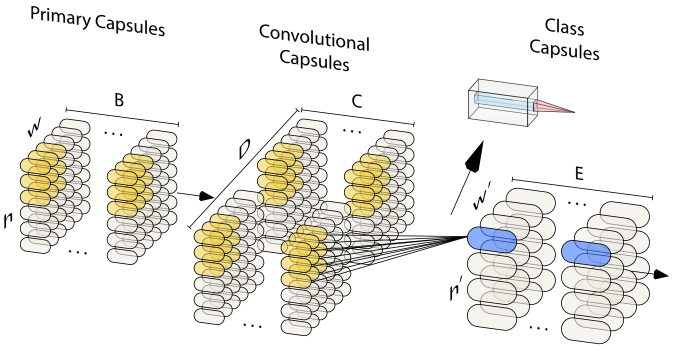


Figure 5: Capsules layers. From left to right: primary, convolutional and class capsules. Primary capsules transform the A outputs of convolutions into B primary capsules. The B primary capsules are then forwarded to capsule convolution layers, producing first C and then D capsules. Finally, E primary capsules are produced by the Class Capsules layer.

The $BS \times A \times F \times F$ output coming from the convolutional layers is now 'encapsulated' into B primary capsules, resulting in a shape of $BS \times B \times P \times P \times F \times F$. Each primary capsule is a matrix capsule (Fig. 3), and thus is composed of an activation value a_i and a pose matrix M_i. To our knowledge, this is the first work in literature that uses matrix capsules [1] to tackle the human pose estimation problem. A brief overview of the capsule layers is shown in Fig. 5.

Primary capsules are followed by 3 convolutional capsules (*ConvCaps*) layers, of which the last one is a class capsules (*ClassCaps*) layer. The first ConvCaps layer outputs capsules of shape $BS{\times}C{\times}P{\times}P{\times}6{\times}6$, while the second has output $BS{\times}D{\times}P{\times}P{\times}4{\times}4$. They both employ a $3{\times}3$ convolutional kernel, with strides 2 and 1 respectively. Finally, the ClassCaps layer gives as output E capsules of $BS \times E \times P \times P$ by sharing weights matrices W across spatial dimensions and using a kernel of size 1 with stride 1. For both convolutional and class capsule voting, we use the VB matrix capsule routing procedure detailed in [15]. Additionally, during capsule convolutions, we learn an inverse graphics matrix $\hat{y}_W$, similarly to what Ramìrez et al. introduce in chapter 3 of [36], but working with matrix capsules and the VB routing. Given each lower-level capsule i and the corresponding higher-level capsule j, we define M_i as the lower level pose matrix and $W_{ij} \in R^{4\times4}$ a trainable viewpoint-invariant transformation matrix such that:

$$V_{j|i} = M_i W_{ij} \tag{5}$$

where $V_{j|i}$ is the vote coming from lower capsules i for higher capsules j, as we show in Eq. 5.

The output of the ClassCaps layer will be flattened into a $BS \times 272$ latent space vector representation, which contains compressed 3D, 2D and joint heatmaps data for the entire batch.

3.4. Decoding

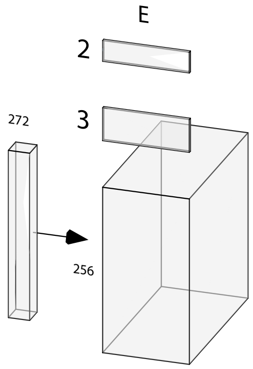


Figure 6: Decoding phase: from a shared *e Entities vector* to 2D, 3D joints and 256 × 256 joint feature maps using a multi-task loss and three separate linear decoders.

The final part of the proposed architecture deals with decoding the $BS \times 272$ latent space coming from capsule layers, to obtain 2D, 3D and heatmaps joints representations. During the joints reconstruction, we employ three separate dense sub-networks, without sharing any layer between the three. This is crucial because the 2D information is not used to 'lift' the 3D joints, and vice-versa. For every dense sub-network, we introduce a DDGELU (Dense Dropout GELU), which we define as a sequential combination of a dense linear layer, followed by a GELU activation and finally dropout, as shown in Eq. 6.

$$\text{DDGELU}_{0.3}(x) = Dropout_{0.3}(\text{GELU}(Linear(x))) \tag{6}$$

During training, we noticed both an increment in convergence speed and a decrease in the train-test loss gap, which should lead to better generalization of the model. As shown in Algorithm 1, we iterate 2 DDGELU layers for each desired output, and choose three different activation functions, depending on the output:

- *sigm* activation for the 2D joints

- *tanh* activation for the 3D joints, which extended range between -1 and 1 better suits the considered 3D domain
- *ReLU6* activation for the joint heatmaps, which provides a good convergence during training for the reconstruction task

Finally, we reshape the heatmaps vector into $BS \times E \times 64 \times 64$ feature maps, which are given as input to the final transposed convolution layer; this will produce full-size $BS \times E \times 256 \times 256$ feature maps. During training, transposed convolutions have shown better results and faster convergence when compared to simple bilinear or bicubic interpolation. A summary of the three final outputs is shown in Fig. 6.

3.4.1. Loss

Multi-tasks network training has shown multiple advantages in learning efficiency and prediction accuracy over the years. For this reason, we encourage the network to jointly learn multiple tasks (3D, 2D, and heatmaps) by employing a self-balancing loss that takes into account the contribution of each task (Eq. 7). To clarify, we stress the fact that during joints reconstruction we employ three separate reconstruction sub-networks so that the network is forced to learn a multi-task enabled latent space.

$$\begin{aligned}\mathcal{L}(x) &= \sum_{\tau \in \mathcal{T}} \left(s_\tau + e^{-s_\tau} \mathcal{L}_\tau\right) \\ &= \left(s_{3D} + e^{-s_{3D}} \mathcal{L}_{3D}\right) + \left(s_{2D} + e^{-s_{2D}} \mathcal{L}_{2D}\right) \\ &+ \left(s_H + e^{-s_H} \mathcal{L}_H\right) + \left(s_W + e^{-s_W} \mathcal{L}_W\right) \\ \mathcal{T} &= \{3\mathcal{D}, 2\mathcal{D}, \mathcal{R}, \mathcal{W}\}\end{aligned} \tag{7}$$

The proposed loss L is able to self-balance through trainable register buffers s_{3D}, s_{2D}, s_H, s_W which dynamically change their value to weight the contributions for each task, at the same time mitigating overfitting for single contributions. To demonstrate the positive effects of multi-task learning in the considered training scenario, we experimented with different sub-networks combinations for the decoder, as well as with only one sub-network. The overall convergence at training time was slower and worse in every test involving less than 3 tasks, while the best results were obtained when enforcing at least 3 tasks (2D joints estimation, 3D joints estimation, joint heatmaps reconstruction). The main explanation is that by forcing the network to perform multiple tasks, it is encouraged to organize its latent space in a more efficient way, leading to a more coherent and less cluttered feature representation. We stress the fact that every sub-network does not share any parameter with each other, to further promote this effect. Multi-task learning also enforces a better representation of data points in the latent space (Fig. 7). Same class joints tend to cluster together better when enforcing the multi-task constraints, while at the same time reducing the number of outliers.

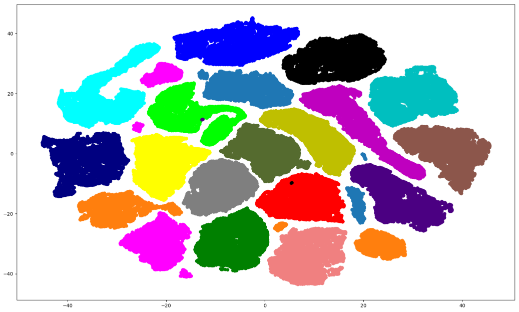


Figure 7: Organization of the latent space after t-SNE: the colour of each sample point corresponds to a joint class.

4. Results

In the next sections we briefly describe the full experimental setup, some implementation details, as well as quantitative and qualitative results.

4.1. Setup description

For training and evaluation, we follow the default protocol #1 from Human3.6M [2]. In more details, we employ all the data from subjects 9 and 11 for evaluation, while only training on data from subjects $1, 5, 6, 7, 8$. The employed metric is the Mean Per Joint Position Error (MPJPE) in millimeter between the ground-truth and the prediction. This metric is applied across all joints and cameras. We also consider two scenarios: one in which the poses are aligned using the Procrustes transformation and another one in which we do not align the poses, as shown in Table 1.

4.2. Implementation details

Our architecture is fully end-to-end, requiring as input just one image and no additional information such as 2D joints ground truth, multiple sequential frames, or non-standard data augmentation. Compared to the majority of methods present in literature, we don't rely on additional datasets for training, at the same time showing high generalization capabilities even after training on a subset of the available data. The metrics we use for comparison are the Mean Per Joint Position Error (MPJPE) in millimetres for each of the 15 activities in the Human3.6M dataset and the average by activity MPJPE, for each camera in the dataset. As for the implementation, the network we present is written using Pytorch Lightning, focusing on high modularity, allowing for real-time joint 3D and 2D predictions, achieving over 229 FPS ($0.00436 s/frame$) on an Nvidia GeForce 1080Ti (desktop) and over 52 FPS ($0.01913 s/frame$) on and Nvidia GeForce 1050 Mobile (laptop), almost twice as fast as what is reported in [36]. All the results were conducted on the same exact hardware and in the same conditions. In Fig. 8 we show the activity-wise and mean frames per second of our architecture compared to the other capsule-based networks [36] on a high-end, desktop-grade GPU. In this scenario, our architecture allows for a 2.33× speed-up. Even in more resource-constrained scenarios (laptop-grade GPU, Fig. 9) we manage to gain an additional $\sim 15 FPS$ on average. According to our experiments, the biggest improvements in terms of speed mostly come down to a combination of simplified network structure, the usage of the improved capsule paradigm and faster routing. In the following sections, we show some quantitative and qualitative results as well, both from the Human3.6 dataset and in-the-wild.

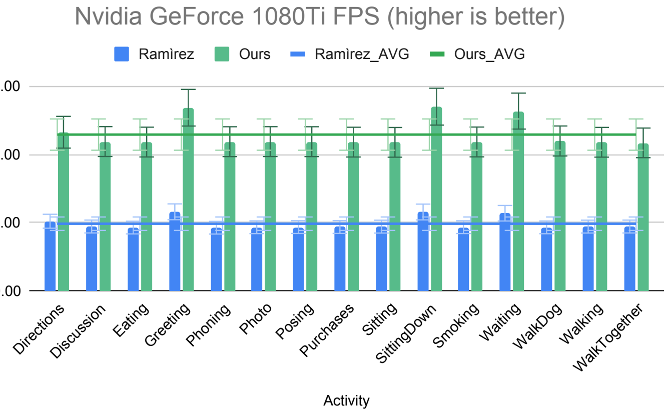


Figure 8: Activity-wise and average inference speed comparison on the same hardware (Nvidia GeForce 1080Ti).

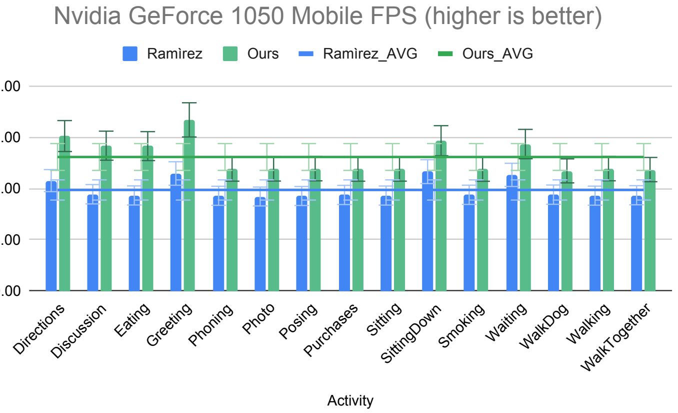


Figure 9: Activity-wise and average inference speed comparison on the same hardware (Nvidia GeForce 1050 Mobile).

4.3. Quantitative results

In Table 1 we show our results compared to the state-of-the-art methods, both the ones using Procrustes transformation (right) and the ones reporting results without Procrustes (left). We achieve the lowest average MPJPE in both the categories and on most of the activities, without using additional information or non-standard data augmentation. Works using additional data, such as 2D-to-3D lifting, ground truth 2D joints, multiple datasets or temporal information are marked in Table 1 with a * symbol. We achieve similar or better results even with those methods, without relying on additional information, dataset or data augmentation, as shown in Table 2. Even considering other similar works that employ additional information, we obtain the lowest average MPJPE scores (yellow row). Compared to the only other work in literature using CapsNet [36], our model achieves better MPJPE in almost every activity.

For the sake of completeness, we selected the top recent works in literature (2019-2020) with the lowest average MPJPE on the Human3.6M dataset, working on monocular data (Table 2). However, as Table 2 shows, most of the works are aided by 2D ground truth information, meaning that they cannot be properly considered end-to-end. Additionally, many of them even exploit temporal frame sequences to refine joint predictions, thus non-working with single images. Others use additional datasets and hand-crafted data augmentation of biometric models during training. We stress the fact that a big advantage of employing capsule networks is the increased generalization capabilities, which highly reduce the need for additional training data, and at the same time boost network efficiency. Nonetheless, even considering the more recent results that use additional information or datasets, our results remain comparable.

Activity	No Procrustes								Procrustes			
	Zhou *	Tekin *	**Tome, I** *	Ramìrez, I	Tome, II *	Ramìrez, II	Ramìrez, III	Ours, I	Sanzari *	Bogo *	Ramìrez, IV	Ours, II
	[22]	[21]	[19]	[36]	[19]	[36]	[36]		[24]	[29]	[36]	
Directions	87.36	85.03	68.55	79.42	*64.98*	73.15	73.33	**70.16**	*48.82*	62	57.55	**55.02**
Discussion	109.31	108.79	78.27	83.73	*73.47*	84.95	83.45	**76.67**	*56.31*	60.2	61.32	**58.06**
Eating	87.05	84.38	77.22	84.01	*76.82*	85.87	85.33	**78.41**	95.98	67.8	66.48	**60.91**
Greeting	103.16	98.94	89.05	83.15	86.43	80.12	79.08	**76.87**	84.78	76.5	64.49	**61.69**
Phoning	116.18	119.39	91.63	**86.42**	*86.28*	91.44	89.99	87.99	96.47	92.1	68	**66.49**
Photo	143.32	95.65	110.05	112.38	110.67	**109.42**	109.95	109.49	105.58	*77*	83.16	**80.02**
Posing	106.88	98.49	74.92	81.34	*68.93*	76.40	76.08	**72.23**	66.3	73	56.05	**54.94**
Purchases	99.78	93.77	83.71	77.65	74.79	76.72	73.61	**73.12**	107.41	75.3	54.85	**52.89**
Sitting	124.52	*73.76*	115.94	105.10	110.19	105.54	**104.12**	108.84	116.89	100.3	**77.65**	80.11
SittingDown	199.23	170.40	185.72	135.55	173.91	**130.15**	136.27	149.53	129.63	137.3	**97.32**	99.84
Smoking	107.42	85.08	88.25	88.25	*84.95*	88.07	87.59	**87.29**	97.84	83.4	**67.31**	67.86
Waiting	118.09	116.91	88.73	79.24	85.78	80.25	79.19	**75.14**	65.94	77.3	59.63	**57.71**
WalkDog	114.23	113.72	92.37	87.45	*86.26*	88.75	**87.13**	87.70	130.46	79.7	**64.76**	65.28
Walking	79.39	62.08	76.48	67.56	71.36	**66.10**	66.31	**65.38**	92.58	86.8	**49.96**	51.19
WalkTogether	97.70	94.83	77.95	80.45	*73.14*	76.84	76.88	**75.76**	102.21	81.7	**60.47**	61.04
Avg, by activity	112.91	100.08	93.26	88.78	88.53	87.58	87.22	**86.17**	93.15	82.03	65.93	**64.98**
Std, Dev,	27.78	24.21	27.63	16.28	26.21	**15.86**	17.15	20.97	23.97	17.9	**11.74**	12.55

Table 1: Activity-wise MPJPE scores for comparable works (with and without Procrustes transformation), including the top-3 in CVPR'17 Human 3.6 challenge and the top-3 IJCVm Jan'18. Columns marked with * make use of additional information or datasets, among the ones depicted in Table 2. Results in **bold** show the best MPJPE score among methods not relying on multiple datasets or additional information at training time. *Underlined* results show the best MPJPE score among all the methods, including the ones employing additional training time information.

	Year	L.	T.	M.D.	D.A.
Cheng [44]	2020		X	X	X
Pham [45]	2019	X	X	X	
Zhao [46]	2019	X		X	
Chen [47]	2020		X		X
Lin [48]	2019	X	X		
Sharma [49]	2019	X		X	X
Tripathi [50]	2020	X	X		X
Wandt [51]	2019	X			X
Arnab [52]	2019	X	X	X	
Mehta [53]	2019	X	X	X	
Ours	2020				

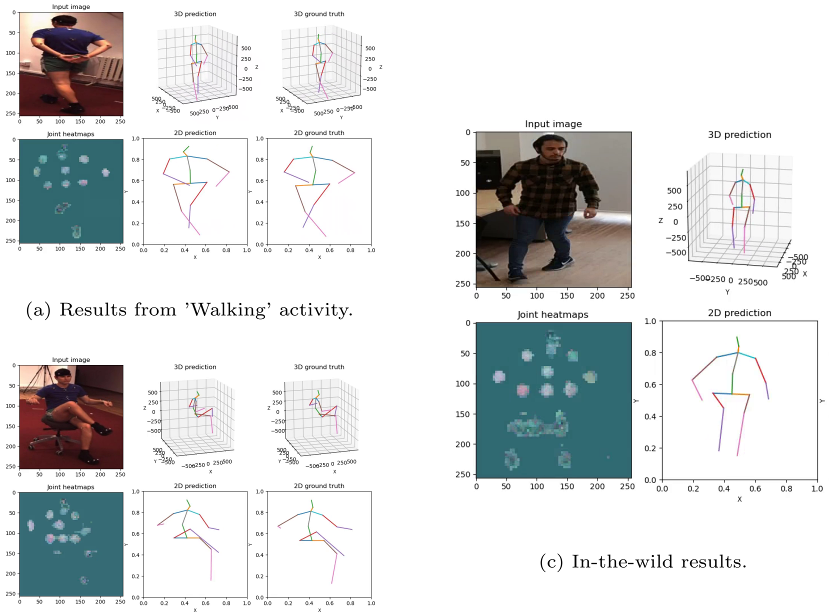
Table 2: Comparison of the most relevant competing methods from 2019-2020 (top Average MPJPE on Human3.6M). **L.**: using 2D joints ground truth and/or lifting from 2D joints, **T.**: using temporal information, **M.D.**: using multiple training datasets, **D.A.**: using non-standard data augmentation techniques or biometric models. In the table we did not include works with lower Average MPJPE than ours.

4.4. Qualitative results

In Figs. 10a, 10b we show some qualitative results for the *Walking* and *Sitting Down* activities from test examples of the Human3.6 dataset. Starting from the upper left: input RGB image, predicted 3D pose, ground truth 3D pose, a combination of the 17 ”attention” heatmaps, predicted 2D pose and ground truth 2D pose. In Fig. 10c we show some in-the-wild results (no ground truth is present in this case).

4.5. Generalization capabilities and the effects of data augmentation

One of the main issues that we address in this paper is the promotion of generalization and viewpoint-equivariance capabilities in deep networks. A huge drawback of using deep networks is their intrinsic data dependency, meaning that the task of learning a dataset almost always leads to some degree of overfitting of the network to the underlying data. For this reason, even the best performing HPE methods in Table 2, completely fail when dealing with previously unseen novel viewpoints. As we show in Fig. 11 our network is capable to cope with novel and extreme viewpoints, such as the top-view viewpoint, thus proving the advanced generalization capabilities of our network. Even when



(a) Results from 'Walking' activity.

(b) Results from 'Sitting Down' activity.

(c) In-the-wild results.

Figure 10: Qualitative results on the Human3.6M dataset (a, b) and in-the-wild (c)

.

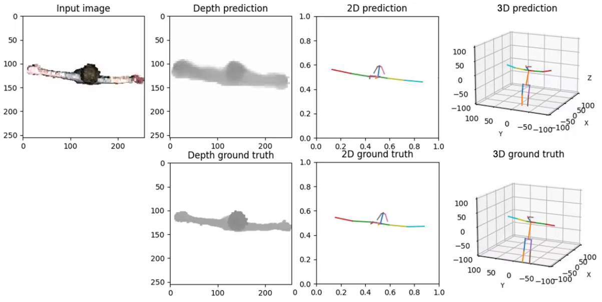


Figure 11: CapsulePose working on rasterized top-view images of the Panoptic Dataset [54] point clouds. Assigned tasks: depth estimation, 2D and 3D top-view pose estimation.

training on data coming from a single viewpoint (e.g. front-view) and testing with data from a completely different viewpoint (e.g. top-view, Fig. 11), our network shows very good generalization capabilities, which are achieved through the hierarchical representation of joints as capsule entities in the latent space. Moreover, in our tests we experienced little to no benefit from using classic data augmentation during training, effectively showing how implicitly learning frame of references and viewpoints has a broader impact on vanilla networks during training time, allowing to greatly reduce the training dataset size, simultaneously simplifying the network complexity and boosting its generalization capabilities.

5. Conclusions

We presented the first human pose estimation architecture based on matrix capsules [1]. The method is real-time and operates in an end-to-end fashion with single images. The simple, modular architecture, paired with a fast Variational Bayes routing and modern frameworks, contributes to achieving very fast performance, running almost twice as fast as other CapsNet-based methods, while at the same time performing better on Human3.6M benchmarks. We show how the presented architecture is competitive with respect to state-of-the-art networks, even by not relying on any additional information or data augmentation at training time, making it a simple yet effective baseline network.

As for future work, we are interested in employing unsupervised capsule autoencoders [14] for human pose estimation and developing a similar model for 3D human mesh recovery.

References

[1] G. E. Hinton, S. Sabour, N. Frosst, Matrix capsules with EM routing, in: International Conference on Learning Representations, 2018.
URL `https://openreview.net/forum?id=HJWLfGWRb`

[2] C. Ionescu, D. Papava, V. Olaru, C. Sminchisescu, Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (7) (2014) 1325–1339.

[3] D. Tome, T. Alldieck, P. Peluse, G. Pons-Moll, L. Agapito, H. Badino, F. De la Torre, Selfpose: 3d egocentric pose estimation from a headset mounted camera, arXiv preprint arXiv:2011.01519.

[4] C. Zimmermann, T. Welschehold, C. Dornhege, W. Burgard, T. Brox, 3d human pose estimation in rgbd images for robotic task learning, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 1986–1992.

[5] L. Bridgeman, M. Volino, J.-Y. Guillemaut, A. Hilton, Multi-person 3d pose estimation and tracking in sports, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, 2019, pp. 0–0.

[6] N. Garau, F. G. De Natale, N. Conci, Fast automatic camera network calibration through human mesh recovery, Journal of Real-Time Image Processing 17 (6) (2020) 1757–1768.

[7] N. Garau, N. Conci, Unsupervised continuous camera network pose estimation through human mesh recovery, in: Proceedings of the 13th International Conference on Distributed Smart Cameras, 2019, pp. 1–6.

[8] M. B. Holte, C. Tran, M. M. Trivedi, T. B. Moeslund, Human pose estimation and activity recognition from multi-view videos: Comparative explorations of recent developments, IEEE Journal of selected topics in signal processing 6 (5) (2012) 538–552.

[9] M. Sebastiani, N. Garau, F. De Natale, N. Conci, Joint trajectory and fatigue analysis in wheelchair users, in: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019, pp. 0–0.

[10] N. Garau, D. Fruet, A. Luchetti, F. De Natale, N. Conci, A multimodal framework for the evaluation of patients' weaknesses, supporting the design of customised aal solutions, Expert Systems with Applications 202 (2022) 117172.

[11] I. Biederman, Recognition-by-components: a theory of human image understanding., Psychological review 94 (2) (1987) 115.

[12] G. E. Hinton, A. Krizhevsky, S. D. Wang, Transforming auto-encoders, in: International conference on artificial neural networks, Springer, 2011, pp. 44–51.

[13] S. Sabour, N. Frosst, G. E. Hinton, Dynamic routing between capsules, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 3859–3869.

[14] A. R. Kosiorek, S. Sabour, Y. W. Teh, G. Hinton, Stacked capsule autoencoders, 2019.
URL `https://arxiv.org/pdf/1906.06818.pdf`

[15] F. Ribeiro, G. Leontidis, S. Kollias, Capsule routing via variational bayes, Proceedings of the AAAI Conference on Artificial Intelligence 34 (2020) 3749–3756. doi:10.1609/aaai.v34i04.5785.

[16] B. Özcan, F. Kınlı, F. Kıraç, Quaternion capsule networks, arXiv preprint arXiv:2007.04389.

[17] Y. Tian, Y. Jia, Y. Shi, Y. Liu, H. Ji, L. Sigal, Inferring 3d body pose using variational semi-parametric regression, in: 2011 18th IEEE International Conference on Image Processing, IEEE, 2011, pp. 29–32.

[18] Z. Cao, T. Simon, S. Wei, Y. Sheikh, Realtime multi-person 2d pose estimation using part affinity fields, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1302–1310.

[19] D. Tome, C. Russell, L. Agapito, Lifting from the deep: Convolutional 3d pose estimation from a single image, 2017. doi:10.1109/CVPR.2017.603.

[20] K. Wang, L. Lin, C. Jiang, C. Qian, P. Wei, 3d human pose machines with self-supervised learning, IEEE Transactions on Pattern Analysis & Machine Intelligence 42 (05) (2020) 1069–1082. doi:10.1109/TPAMI.2019.2892452.

[21] B. Tekin, P. Márquez-Neila, M. Salzmann, P. Fua, Learning to fuse 2d and 3d image cues for monocular body pose estimation, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3941–3950.

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

[23] I. Katircioglu, B. Tekin, M. Salzmann, V. Lepetit, P. Fua, Learning Latent Representations of 3D Human Pose with Deep Neural Networks, International Journal of Computer Vision 126 (12) (2018) 1326–1341.

doi:10.1007/s11263-018-1066-6.
URL https://hal.archives-ouvertes.fr/hal-02509358

[24] M. Sanzari, V. Ntouskos, F. Pirri, Bayesian image based 3d pose estimation, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016, pp. 566–582.

[25] G. Rogez, P. Weinzaepfel, C. Schmid, LCR-Net++: Multi-person 2D and 3D Pose Detection in Natural Images, IEEE Transactions on Pattern Analysis and Machine Intelligence.

[26] Y. Tian, W. Hu, H. Jiang, J. Wu, Densely connected attentional pyramid residual network for human pose estimation, Neurocomputing 347 (2019) 13 – 23. doi:https://doi.org/10.1016/j.neucom.2019.01.104.
URL http://www.sciencedirect.com/science/article/pii/S0925231219301973

[27] J. Liu, H. Ding, A. Shahroudy, L. Duan, X. Jiang, G. Wang, A. C. Kot, Feature boosting network for 3d pose estimation, IEEE Transactions on Pattern Analysis and Machine Intelligence 42 (2) (2020) 494–501.

[28] C. Yan, B. Gong, Y. Wei, Y. Gao, Deep multi-view enhancement hashing for image retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (4) (2020) 1445–1451.

[29] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, M. J. Black, Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image, in: Computer Vision – ECCV 2016, Lecture Notes in Computer Science, Springer International Publishing, 2016.

[30] A. Kanazawa, M. J. Black, D. W. Jacobs, J. Malik, End-to-end recovery of human shape and pose, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 7122–7131.

[31] N. Kolotouros, G. Pavlakos, M. J. Black, K. Daniilidis, Learning to reconstruct 3d human pose and shape via model-fitting in the loop, in: Proceed-

ings of the IEEE International Conference on Computer Vision, 2019, pp. 2252–2261.

[32] M. Keller, S. Zuffi, M. J. Black, S. Pujades, Osso: Obtaining skeletal shape from outside, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 20492–20501.

[33] M. Kocabas, N. Athanasiou, M. J. Black, Vibe: Video inference for human body pose and shape estimation, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 5253–5263.

[34] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, M. Grundmann, Blazepose: On-device real-time body pose tracking, arXiv preprint arXiv:2006.10204.

[35] B. Usman, A. Tagliasacchi, K. Saenko, A. Sud, Metapose: Fast 3d pose from multiple views without 3d supervision, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 6759–6770.

[36] I. Ramírez, A. Cuesta-Infante, E. Schiavi, J. J. Pantrigo, Bayesian capsule networks for 3d human pose estimation from single 2d images, Neurocomputing 379 (2020) 64 – 73. `doi:https://doi.org/10.1016/j.neucom.2019.09.101`.
URL `http://www.sciencedirect.com/science/article/pii/S092523121931522X`

[37] N. Garau, N. Bisagno, P. Bródka, N. Conci, Deca: Deep viewpoint-equivariant human pose estimation using capsule autoencoders, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 11677–11686.

[38] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), arXiv preprint arXiv:1606.08415.

[39] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[40] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, arXiv preprint arXiv:1711.05101.

[41] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, Vol. 9 of Proceedings of Machine Learning Research, JMLR Workshop and Conference Proceedings, Chia Laguna Resort, Sardinia, Italy, 2010, pp. 249–256.
URL `http://proceedings.mlr.press/v9/glorot10a.html`

[42] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[43] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, JMLR.org, 2016, p. 1050–1059.

[44] Y. Cheng, B. Yang, B. Wang, R. T. Tan, 3d human pose estimation using spatio-temporal networks with explicit occlusion training, arXiv preprint arXiv:2004.11822.

[45] H. H. Pham, H. Salmane, L. Khoudour, A. Crouzil, S. A. Velastin, P. Zegers, A unified deep framework for joint 3d pose estimation and action recognition from a single rgb camera, Sensors 20 (7) (2020) 1825.

[46] L. Zhao, X. Peng, Y. Tian, M. Kapadia, D. N. Metaxas, Semantic graph convolutional networks for 3d human pose regression, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3425–3435.

[47] T. Chen, C. Fang, X. Shen, Y. Zhu, Z. Chen, J. Luo, Anatomy-aware 3d human pose estimation in videos, arXiv preprint arXiv:2002.10322.

[48] J. Lin, G. H. Lee, Trajectory space factorization for deep video-based 3d human pose estimation, arXiv preprint arXiv:1908.08289.

[49] S. Sharma, P. T. Varigonda, P. Bindal, A. Sharma, A. Jain, Monocular 3d human pose estimation by generation and ordinal ranking, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 2325–2334.

[50] S. Tripathi, S. Ranade, A. Tyagi, A. Agrawal, Posenet3d: Unsupervised 3d human shape and pose estimation, arXiv preprint arXiv:2003.03473.

[51] B. Wandt, B. Rosenhahn, Repnet: Weakly supervised training of an adversarial reprojection network for 3d human pose estimation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2019, pp. 7782–7791.

[52] A. Arnab, C. Doersch, A. Zisserman, Exploiting temporal context for 3d human pose estimation in the wild, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3395–3404.

[53] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, M. Elgharib, P. Fua, H.-P. Seidel, H. Rhodin, G. Pons-Moll, C. Theobalt, Xnect: Real-time multi-person 3d motion capture with a single rgb camera, ACM Transactions on Graphics (TOG) 39 (4) (2020) 82–1.

[54] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, Y. Sheikh, Panoptic studio: A massively multiview system for social motion capture, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3334–3342.