A Comprehensive Study of Weight Sharing in Graph Networks for 3D Human Pose Estimation

Kenkun Liu[1⋆], Rongqi Ding[2⋆], Zhiming Zou[1], Le Wang[3], and Wei Tang[1⋆⋆]

[1] University of Illinois at Chicago, Chicago, IL, USA
{kliu44,zzou6,tangw}@uic.edu
[2] Northwestern University, Evanston, IL, USA
rongqiding2020@u.northwestern.edu
[3] Xi'an Jiaotong University, Xi'an, Shaanxi, P.R. China
lewang@xjtu.edu.cn

Abstract. Graph convolutional networks (GCNs) have been applied to 3D human pose estimation (HPE) from 2D body joint detections and have shown encouraging performance. One limitation of the vanilla graph convolution is that it models the relationships between neighboring nodes via a shared weight matrix. This is suboptimal for articulated body modeling as the relations between different body joints are different. The objective of this paper is to have a comprehensive and systematic study of weight sharing in GCNs for 3D HPE. We first show there are two different ways to interpret a GCN depending on whether feature transformation occurs before or after feature aggregation. These two interpretations lead to five different weight sharing methods, and three more variants can be derived by decoupling the self-connections with other edges. We conduct extensive ablation study on these weight sharing methods under controlled settings and obtain new conclusions that will benefit the community.

1 Introduction

The task of 3D human pose estimation (HPE) means to predict the locations of human body joints in the camera coordinate system from a single RGB image. It has attracted a lot of attention in recent years [8, 5, 26, 22, 19, 30, 43, 37] due to its broad applications in human-computer interaction, action recognition and robotics. 3D HPE is an ill-posed problem since multiple 3D poses may explain the same 2D projection in the image space. Fortunately, this ambiguity could be largely resolved as the human body is a highly structured object [21, 40].

Previous research on 3D HPE can be divided into two streams. The first one is to regress the 3D human pose directly from the input image [25, 35]. Early work [1, 41] rely on handcrafted features but they are prone to fail in case of depth ambiguity, rare viewpoints and occlusion. Recent approaches [25, 44, 31, 32, 39] exploit convolutional neural networks (CNNs) to learn powerful visual

⋆ The first two authors contributed equally to this work.
⋆⋆ Corresponding author.

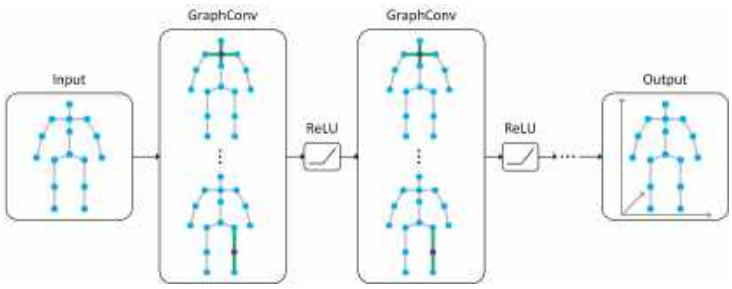

Fig. 1. Illustration of a graph convolutional network (GCN) for 3D human pose estimation. The input is the 2D body joint locations predicted by an off-the-shelf 2D pose detector. A GCN repeatedly transforms and aggregates features of neighboring body joints to learn an increasingly powerful representation. The output 3D pose is predicted by the last layer

representations from large-scale image data [3, 15] and significantly improve the regression accuracy. The second stream follows a two-stage pipeline, i.e., 2D human pose detection [23, 6, 33, 34] followed by 2D-to-3D pose lifting [21, 40, 19]. For example, Martinez et al. [21] use a fully connected network to regress the 3D body joint locations from the output of an off-the-shelf 2D pose detector. This simple baseline is effective and outperforms state-of-the-art one-stage methods.

Recently graph convolutional networks (GCNs) [17, 42] have been applied to solve the 2D-to-3D pose lifting problem [5, 8, 40]. They generalize CNNs by performing convolutions on graph data. As illustrated in Fig. 1, the articulated body skeleton naturally forms a graph wherein body joints and bones respectively correspond to the nodes and edges. A GCN then repeatedly transforms and aggregates features of neighboring body joints and learns their relational patterns which are critical to resolve the depth ambiguity. Compared with a fully connected network [21], a GCN not only learns a compact representation defined on graph nodes but also explicitly captures their structural relationships.

One limitation of the vanilla GCN, which was originlly proposed for graph or node classification, is that it shares a feature transformation for each node within a graph convolutional layer. While weight sharing leads to a more compact model and promotes its generalization ability, it is suboptimal for articulated body modeling. On one hand, the relations between different body joints are different. For example, the ankles of a standing, walking or sitting person are always below their knees while the relational pattern between wrists and elbows are more complex. On the other hand, it is actually the feature transformation that captures the relations between each node and their neighboring nodes. Thus, this kind of weight sharing can prevent the GCN from learning diverse models specific to different relations and therefore adversely affect 3D HPE.

The objective of this paper is to have a comprehensive and systematic study of weight sharing in GCNs for 3D HPE. Specifically, we consider five different weight sharing strategies: `full-sharing`, `pre-aggregation`, `post-aggregation`,

`convolution-style` and `no-sharing`. `Full-sharing` corresponds to the vanilla graph convolution. We show that there are two ways to interpret the graph convolution depending on whether the feature transformation occurs before or after features are aggregated from the neighborhood of each node. `Pre-aggregation` and `post-aggregation` are obtained by unsharing the feature transformations for each node in these two equivalent forms respectively. `Convolution-style` is motivated by the convolution operation used in CNNs, but the displacement between two entities are defined on the graph. `No-sharing` is on the other extreme of `full-sharing` as it defines a different feature transformation between any two nodes. Furthermore, we notice that the affinity matrix used in a GCN usually includes self-connections, i.e., edges connecting each node and itself. Since they do not model relations between different nodes, we consider decoupling them with the edges connecting each node to their neighbors. This leads to three more variants of `full-sharing`, `pre-aggregation` and `post-aggregation`. After conducting extensive ablation study by controlling the number of parameters, computational complexity and number of channels for these weight sharing methods, we find that (1) decoupling self-connections is critical to achieve good performance, (2) different weight sharing strategies, even with the same number of parameters, have a significant impact on the performance of 3D HPE, and (3) `pre-aggregation` with decoupled self-connections is the optimal weight sharing method in GCNs for 3D HPE.

In sum, the contribution of this paper is twofold.

- To our knowledge, this is the first comprehensive and systematic investigation of weight sharing in GCNs and their impact on articulated pose regression. We study five different weight sharing strategies derived from two perspectives of a graph convolution as well as three more variants based on decoupling self-connections.
- We conduct extensive experiments to compare the different weight sharing methods under controlled settings. We make new conclusions that we believe will benefit not only the research community of human pose estimation but also that of deep learning on graphs.

2 Related Work

3D Human Pose Estimation. The development of 3D HPE has gone through a long time. In the beginning, researchers build 3D pose models based on handcrafted features and geometric constraints [2, 28, 14]. Recent approaches exploit deep neural networks [44, 31, 32, 39, 21, 40, 19] for end-to-end learning and significantly push forward the state-of-the-art performance. Zhou et al. [43] augment a 2D pose estimation sub-network with a 3D depth regression sub-network and introduce a weakly-supervised transfer learning method to make full use of mixed 2D and 3D labels. Sun et al. [32] introduce a simple integral operation to relate and unify the heat map representation and body joint regression. Yang et al. [39] design an adversarial learning framework, which distills the 3D human pose structures learned from the fully annotated dataset to in-the-wild images with

only 2D pose annotations. 3D HPE from videos have been studied in [7, 26]. There is also research on multi-person 3D pose detection [22].

Some researchers divide the 3D HPE task into two sub-tasks, i.e., 2D human pose estimation from an image and 2D-to-3D pose lifting. Our approach falls into this category. The most related work to ours are [40, 8, 5], which also apply GCNs for 3D pose regression. Zhao et al. [40] propose a semantic GCN to capture local and global node relationships not explicitly represented in the graph. Ci et al. [8] extend the GCN to a locally connected network to improve its representation capability. This model is actually equivalent to the `no-sharing` method discussed in this paper. Cai et al. [5] incorporate domain knowledge about the articulated body configurations into the graph convolutional operations and introduce a local-to-global network to learn multi-scale features for the graph-based representations. They classify neighboring nodes according to their semantic meanings and use different kernels for different neighboring nodes. Their form is similar to that of our `convolution-style` method but our derivation is inspired by the spatial convolution and is more general. Moreover, we also consider new weight sharing methods based on two different interpretations of the graph convolution and decoupling self-connections. Although decoupling self-connections in a GCN has been studied in [38] and [40], we investigate it in different weight sharing methods. To our knowledge, this is the first systematic study of weight sharing in GCNs for 3D human pose estimation.

Graph Convolutional Networks. GCNs generalize CNNs by performing convolutions on graphs. They have been widely used to solve problems involving graph data like the citation network [17], news network [13], molecular property prediction [11] and information retrieval [29]. There are two categories of GCNs: spectral approaches and non-spectral (spatial) approaches [42]. The former are defined in the Fourier domain by calculating eigen-decomposition of graph Laplacian [4], and the latter apply neural message passing to features defined on a graph [11]. Our approach falls into the second category. GCNs are conventionally used for graph or node classification and share the feature transformation for each node so that they could work on graphs with arbitrary structures. By contrast, the human body skeleton has a fixed structure consisting of diverse relations. Thus, it is important to study weight sharing for 3D HPE.

3 Our Approach

We first revisit a vanilla GCN and show two different ways to interpret it (Sec. 3.1). We introduce and compare different weight sharing methods in Sec. 3.2. Sec. 3.3 discusses weight sharing based on decoupled self-connections. Finally, we detail the network architecture for 3D HPE in Sec. 3.4.

3.1 Understand GCN

A GCN generalizes CNNs by learning representations on graph data. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph where $\mathcal{V}$ is a set of N nodes and $\mathcal{E}$ is the collection of

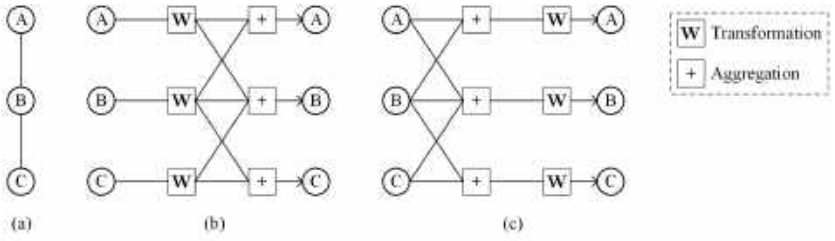

Fig. 2. Illustration of two different ways to interpret a graph convolution. (a) A simple graph consisting of three nodes. (b) The features of each node are updated via feature transformation followed by neighborhood aggregation. (c) The features of each node are updated via neighborhood aggregation followed by feature transformation

all edges. We can encode the edges in an adjacency matrix $\mathbf{A} \in \{0,1\}^{N\times N}$. Let $\mathbf{h}_i \in \mathcal{R}^D$ denote a D-dimensional feature vector associated with each node i. $\mathbf{H} \in \mathcal{R}^{D\times N}$ is the collection of all feature vectors, and its i-th column is $\mathbf{h}_i$. Then a graph convolutional layer [17], the building block of a GCN, updates features defined on the nodes via the following operation:

$$\mathbf{H}' = \sigma(\mathbf{W}\mathbf{H}\hat{\mathbf{A}}) \tag{1}$$

where $\mathbf{H}' \in \mathbb{R}^{D'\times N}$ is the updated feature matrix, D' is the dimension of the updated feature vector of each node, $\sigma(\cdot)$ is an activation function, e.g., ReLU, $\mathbf{W} \in \mathbb{R}^{D'\times D}$ is a learnable weight matrix. $\hat{\mathbf{A}}$ is a normalized version of $\mathbf{A}$:

$$\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\tilde{\mathbf{D}}^{-\frac{1}{2}} \tag{2}$$

Adding an identity matrix $\mathbf{I}$ to $\mathbf{A}$ means to include self-connections in the graph so that the update of a node feature vector also depends on itself. $\tilde{\mathbf{D}}$ is the diagonal node degree matrix of $\mathbf{A}+\mathbf{I}$ and helps the graph convolution to retain the scale of features. A GCN takes as input a feature vector associated with each node and repeatedly transforms them via a composition of multiple graph convolutions to get increasingly more powerful representations, which are used by the last layer to predict the output.

Let $\hat{a}_{ij}$ be the entry of $\hat{\mathbf{A}}$ at (i,j). $\mathcal{N}_i$ and $\hat{\mathcal{N}}_i \equiv \mathcal{N}_i \cup \{i\}$ denote the neighbors of node i excluding and including the node itself respectively. This means $j \in \hat{\mathcal{N}}_i$ if and only if $\hat{a}_{ij} \neq 0$. Then Eq. (1) can be written equivalently as below.

$$\mathbf{h}'_i = \sigma(\sum_{j\in\hat{\mathcal{N}}_i} \mathbf{W}\mathbf{h}_j\hat{a}_{ij}) \tag{3}$$

$$= \sigma(\mathbf{W}\sum_{j\in\hat{\mathcal{N}}_i} \mathbf{h}_j\hat{a}_{ij}) \tag{4}$$

where $i \in \{1,...,N\}$, $\mathbf{h}'_i$ is the i-th column of $\mathbf{H}'$ and also the updated feature vector of node i.

Table 1. Comparison of different weight sharing methods

Method	Definition	Parameters	Complexity
`Full-sharing`	$\mathbf{h}'_i = \sigma(\sum_{j\in\hat{\mathcal{N}}_i} \mathbf{W}\mathbf{h}_j\hat{a}_{ij})$	$D' \times D$	$D' \times D \times \sum_{i=1}^{N} \lvert\hat{\mathcal{N}}_i\rvert$
`Conv-style`	$\mathbf{h}'_i = \sigma(\sum_{j\in\hat{\mathcal{N}}_i} \mathbf{W}_{d(i,j)}\mathbf{h}_j\hat{a}_{ij})$	$D' \times D \times 3$	$D' \times D \times \sum_{i=1}^{N} \lvert\hat{\mathcal{N}}_i\rvert$
`Pre-agg`	$\mathbf{h}'_i = \sigma(\sum_{j\in\hat{\mathcal{N}}_i} \mathbf{W}_j\mathbf{h}_j\hat{a}_{ij})$	$D' \times D \times N$	$D' \times D \times \sum_{i=1}^{N} \lvert\hat{\mathcal{N}}_i\rvert$
`Post-agg`	$\mathbf{h}'_i = \sigma(\mathbf{W}_i\sum_{j\in\hat{\mathcal{N}}_i} \mathbf{h}_j\hat{a}_{ij})$	$D' \times D \times N$	$D' \times D \times \sum_{i=1}^{N} \lvert\hat{\mathcal{N}}_i\rvert$
`No-sharing`	$\mathbf{h}'_i = \sigma(\sum_{j\in\hat{\mathcal{N}}_i} \mathbf{W}_{ij}\mathbf{h}_j\hat{a}_{ij})$	$D' \times D \times \sum_{i=1}^{N} \lvert\hat{\mathcal{N}}_i\rvert$	$D' \times D \times \sum_{i=1}^{N} \lvert\hat{\mathcal{N}}_i\rvert$

This equivalence indicates we can interpret the graph convolution in two different ways. Specifically, Eq. (3) updates the feature vector of each node by first transforming the features of their neighboring nodes via $\mathbf{W}$ and then aggregating those transformed features via a summation. Alternatively, Eq. (4) first aggregates features of neighboring nodes and then transforms the aggregated features via a linear projection. These two interpretations are illustrated in Fig. 2. We will show how they can be used to derive different weight sharing methods.

3.2 Weight Sharing

The transformation matrix in a graph convolution captures the relationships between nodes. It is conventionally shared by all nodes, i.e., Eqs. (3) and (4). This weight sharing method, which we call `full-sharing`, has several advantages. First, it leads to a compact model potentially with better generalization ability. Second, it allows us to apply the same GCN to graphs with arbitrary structures, which is critical to graph or node classification tasks. However, `full-sharing` can be suboptimal for human pose estimation as the relations among different sets of body joints are different. To resolve this potential issue, we try to *unshare* a portion of the weights, which leads to different weight sharing methods.

Pre-aggregation. Motivated by Eq. (3), we consider applying different transformations to the input features of each node before they are aggregated:

$$\mathbf{h}'_i = \sigma(\sum_{j\in\hat{\mathcal{N}}_i} \mathbf{W}_j\mathbf{h}_j\hat{a}_{ij}) \tag{5}$$

where $\mathbf{W}_j \in \mathcal{R}^{D'\times D}$ $(j \in \{1, ..., N\})$ is the weight matrix applied to $\mathbf{h}_j$. Since the weight unsharing occurs before feature aggregation, we call this method `pre-aggregation`.

Post-aggregation. Alternatively, we can also unshare the weights to get the output features of each node after the aggregation step. Eq. (4) is reformulated:

$$\mathbf{h}'_i = \sigma(\mathbf{W}_i\sum_{j\in\hat{\mathcal{N}}_i} \mathbf{h}_j\hat{a}_{ij}) \tag{6}$$

where $\mathbf{W}_i \in \mathcal{R}^{D' \times D}$ ($i \in \{1, ..., N\}$) is the transformation to get the output features of node i from its neighbors. We call it `post-aggregation` as the weight unsharing occurs after feature aggregation.

No-sharing. `Pre-aggregation` and `post-aggregation` differ in whether the weights are unshared for the input or output features of each node. It is straightforward to combine them and unshare the weights between any pair of input and output feature vectors:

$$\mathbf{h}'_i = \sigma(\sum_{j \in \hat{\mathcal{N}}_i} \mathbf{W}_{ij} \mathbf{h}_j \hat{a}_{ij}) \tag{7}$$

where $\mathbf{W}_{ij} \in \mathcal{R}^{D' \times D}$ is the weight matrix corresponding to the input features of node j and the output features of node i. We call this method `no-sharing`. Note the number of different weight matrices is $\sum_{i=1}^{N} |\hat{\mathcal{N}}_i|$ instead of N^2.

Convolution-style. An image is a special kind of graph with its nodes or pixels arranged in a grid. This makes it possible to define the *displacement* between any two pixels by subtracting their 2D coordinates on the grid. Then the spatial convolution on an image can be considered as a weight sharing method by assigning different feature transformations to each displacement value between two nodes. Thus, if we can define the displacement $d(i, j)$ between two nodes i and j on a graph, we can develop a `convolution-style` weight sharing method:

$$\mathbf{h}'_i = \sigma(\sum_{j \in \hat{\mathcal{N}}_i} \mathbf{W}_{d(i,j)} \mathbf{h}_j \hat{a}_{ij}) \tag{8}$$

where $\mathbf{W}_{d(i,j)} \in \mathcal{R}^{D' \times D}$ is the weight matrix corresponding to each displacement value $d(i, j)$. Motivated by the fact that the skeleton graph as shown in Fig. 1 has a star shape and body joints farther away from the body center have larger degrees of freedom, we define the *coordinate* of a body joint node as the length of the shortest path between it and the body center node. Thus for a pair of neighboring nodes $i \in \{1, ..., N\}$ and $j \in \hat{\mathcal{N}}_i$, we have $d(i, j) \in \{-1, 0, 1\}$. In other words, the relationships between a node and (1) itself, (2) a neighboring body joint farther from the body center and (3) a neighboring body joint closer to the body center are modeled separately.

Tab. 1 compares definitions of the five weight sharing methods as well as their parameters and computational complexities based on a unified implementation. Possible ways to reduce the computational complexity are discussed in the supplementary material. With the same dimensions of input and output features, `full-sharing` has the smallest number of parameters while `no-sharing` is on the other extreme. Fig. 3 illustrates different weight sharing methods.

3.3 Decouple Self-connections

The normalized affinity matrix $\hat{\mathbf{A}}$ used in a GCN usually includes self-connections, i.e., edges connecting each node and itself. Unlike edges connecting nodes to their neighbors, self-connections do not involve relational modeling between different

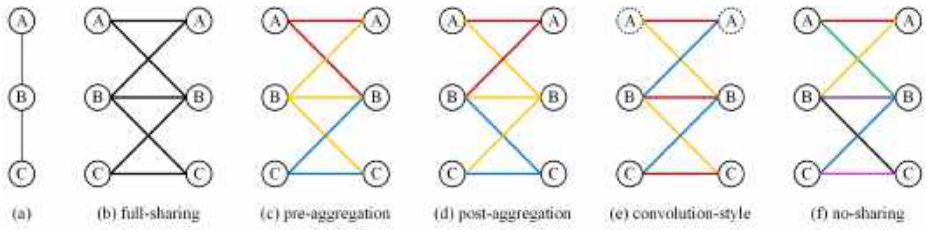

Fig. 3. Illustration of the five weight sharing methods discussed in Sec. 3.2. Weight unsharing is encoded via different colors. For `convolution-style`, we assume the coordinate of a node is defined as its graph distance to node A

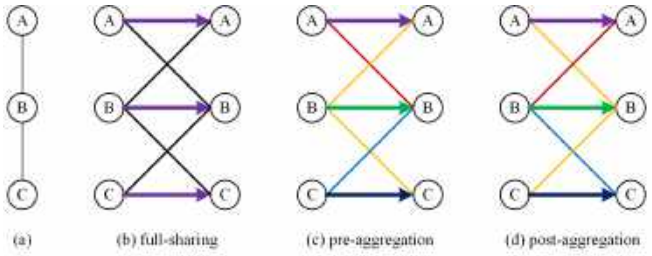

Fig. 4. Illustration of decoupled weight sharing methods. Weight unsharing is encoded via different colors. Arrows denote weights corresponding to self-connections

nodes. This motivates us to decouple their feature transformations, which leads to variants of `full-sharing`, `pre-aggregation` and `post-aggregation`:

$$\mathbf{h}'_i = \sigma(\mathbf{T}\mathbf{h}_i\hat{a}_{ii} + \sum_{j\in\mathcal{N}_i} \mathbf{W}\mathbf{h}_j\hat{a}_{ij}) \tag{9}$$

$$\mathbf{h}'_i = \sigma(\mathbf{T}_i\mathbf{h}_i\hat{a}_{ii} + \sum_{j\in\mathcal{N}_i} \mathbf{W}_j\mathbf{h}_j\hat{a}_{ij}) \tag{10}$$

$$\mathbf{h}'_i = \sigma(\mathbf{T}_i\mathbf{h}_i\hat{a}_{ii} + \mathbf{W}_i\sum_{j\in\mathcal{N}_i} \mathbf{h}_j\hat{a}_{ij}) \tag{11}$$

where $\mathbf{T} \in \mathcal{R}^{D'\times D}$ and $\mathbf{T}_i \in \mathcal{R}^{D'\times D}$ denote feature transformations for self-connections, $\mathcal{N}_i \equiv \hat{\mathcal{N}}_i - \{i\}$. For decoupled `full-sharing`, i.e., Eq. (9), the weight matrix $\mathbf{T}$ is fully shared by all self-connections. For decoupled `pre-aggregation` and decoupled `post-aggregation`, i.e., Eqs. (10) and (11), we modify their respective original formulations so that a different weight matrix is assigned to each self-connection and they are different from weight matrices for other connections. Note `convolution-style` and `no-sharing` decouple self-connections by definition. Fig. 4 illustrates these three decoupled weight sharing methods.

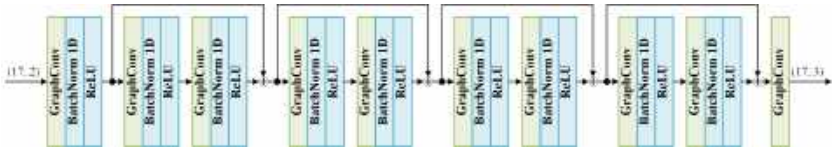

Fig. 5. An example GCN architecture for 2D-to-3D pose lifting. The building block is a residual block composed of two graph convolutional layers with 128 channels. This block is repeated four times. Each graph convolutional layer (except for the last one) is followed by a batch normalization layer and a ReLU activation

Table 2. Ablation study on the impact of decoupling self-connections. We adjust the channels so that each pair of comparing methods have similar model sizes. All errors are measured in millimeters (mm)

Method	Decouple?	Channels	Params	MPJPE	P-MPJPE	Loss
`Full-sharing`	No	180	0.27 M	53.53	43.37	0.000630
`Full-sharing`	Yes	128	0.27 M	**41.96**	**33.69**	0.000140
`Pre-agg`	No	180	4.17 M	40.34	31.59	0.000027
`Pre-agg`	Yes	128	4.22 M	**37.83**	**30.09**	0.000016
`Post-agg`	No	180	4.17 M	41.39	33.68	0.000052
`Post-agg`	Yes	128	4.22 M	**38.92**	**31.33**	0.000022

3.4 Network Architecture

We use the network architecture shown in Fig. 5 for 3D human pose estimation and compare different weight sharing methods in the experiments. Following Martinez et al. [21] and Defferrard et al. [9], we stack multiple cascaded blocks, each of which is composed of two graph convolutional layers interleaved with batch normalization and ReLU. Then, each block is wrapped in a residual connection. The input to the GCN is the 2D coordinates of the body joints in the image space and the output is the corresponding 3D locations in the camera coordinate system. We use an $L2$-norm loss between the prediction and the ground truth. The network can be trained end-to-end.

4 Experiments

4.1 Datasets and Evaluation Protocols

We conduct our experiments on the widely used Human 3.6M dataset [15] and follow the standard evaluation procedure.

Dataset. The Human 3.6M dataset is currently the largest public dataset for 3D HPE. It contains 3.6 million images filmed by 4 synchronized high-resolution progressive scan cameras at 50 Hz [15]. 11 subjects perform 15 different daily activities in the film like eating, phoning, sitting and walking, but only 7 subjects are annotated with 3D poses. We follow previous work [26, 40, 8] for fair and

Table 3. Ablation study on different weight sharing methods (controlled number of channels and computational complexity). All errors are measured in millimeters (mm)

Method	Channels	Params	MPJPE	P-MPJPE	Loss
`Full-sharing`	128	0.27 M	41.96	33.69	0.000140
`Conv-style`	128	0.40 M	42.85	33.32	0.000078
`Pre-agg`	128	4.22 M	**37.83**	**30.09**	0.000016
`Post-agg`	128	4.22 M	38.92	31.33	0.000022
`No-sharing`	128	6.86 M	39.88	31.02	0.000013

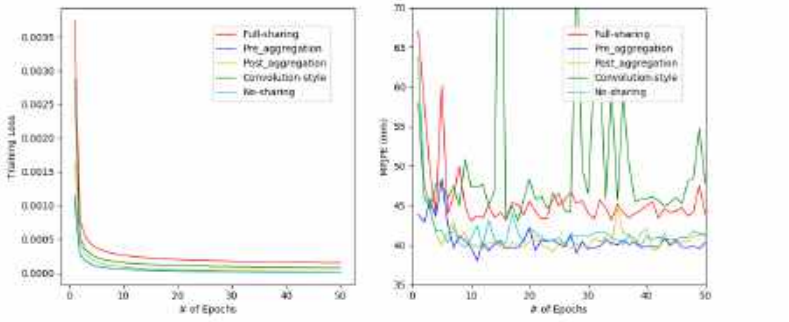

Fig. 6. Training curves (left) and validation errors (right) of different methods (controlled number of channels and computational complexity). Curves corresponding to controlled model size can be found in the supplementary material

effective comparison. The 7 annotated subjects are divided into 5 subjects (S1, S5, S6, S7, S8) for training and 2 subjects (S9 and S11) for testing. A single model is trained and tested on all 15 actions.

Evaluation protocols. There are two protocols for evaluation. **Protocol-1** uses the mean per-joint position error (MPJPE) as the evaluation metric. It computes the mean Euclidean distance error per-joints between the prediction and the ground truth in millimeters. **Protocol-2** computes the error after aligning the root joint of the prediction with ground truth via rigid transformation. This metric is abbreviated as P-MPJPE.

4.2 Ablation Study

We conduct ablation study to compare the five different weight sharing methods and the three variants in controlled settings. To avoid the influence of 2D human pose detector, we utilize the 2D ground truth as the input for all models. We adopt Adam [16] as our optimization method with an initial learning rate 0.001 and a decay rate 0.96 every 100K iterations. We train each model for 50 epochs using a batch size 64. The weights of GCNs are initialized using the method

Table 4. Ablation study on different weight sharing methods (controll the model size to be 4.2 M, 2.1 M or 1.05 M). All errors are measured in millimeters (mm)

Method	4.2M		2.1M		1.05M	
	MPJPE	P-MPJPE	MPJPE	P-MPJPE	MPJPE	P-MPJPE
`Full-sharing`	41.70	33.02	42.20	33.19	42.00	33.37
`Conv-style`	41.19	32.20	43.14	33.32	42.73	34.09
`Pre-agg`	**37.83**	**30.09**	39.86	**31.14**	**40.14**	**31.17**
`Post-agg`	38.92	31.33	39.99	32.06	40.41	32.51
`No-sharing`	39.62	30.93	**39.49**	31.15	40.75	31.31

Table 5. Quantitative comparisons on the Human 3.6M dataset under **Protocol-1**. The MPJPEs are reported in millimeters. The best results are highlighted in bold and second underlined. **Legend:** (+) uses extra data from MPII dataset. (†) uses temporal information. (*) uses pose scales in both training and testing

Protocol # 1	Direct.	Discuss	Eating	Greet	Phone	Photo	Pose	Purch.	Sitting	SittingD.	Smoke	Wait	WalkD.	Walk	WalkT.	Avg.
Lee et al. [18] 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
Hossain et al. [27] ECCV'18 (†)	44.2	46.7	52.3	49.3	59.9	59.4	47.5	46.2	59.9	65.6	55.8	50.4	52.3	43.5	45.1	51.9
Pavllo et al. [26] CVPR'19 (†)	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
Cai et al. [5] ICCV'19 (†)	44.6	47.4	45.6	48.8	50.8	59.0	47.2	43.9	57.9	61.9	49.7	46.6	51.3	37.1	39.4	48.8
Martinez et al. [21] 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
Tekin et al. [36] 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. [21] 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. [31] 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
Yang et al. [39] 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
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. [24] 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
Luvizon et al. [20] 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
Zhao et al. [40] CVPR'19	48.2	60.8	51.8	64.0	64.6	53.6	51.1	67.4	88.7	57.7	73.2	65.6	48.9	64.8	51.9	60.8
Li et al. [19] CVPR'19	**43.8**	48.6	49.1	49.8	57.6	61.5	**45.9**	48.3	62.0	73.4	54.8	50.6	56.0	43.4	45.5	52.7
Zhou et al. [40] CVPR'19	47.3	60.7	51.4	60.5	61.1	**49.9**	47.3	68.1	86.2	**55.0**	67.8	61.0	**42.1**	60.6	45.3	57.6
Sharma et al. [30] ICCV'19	48.6	54.5	54.2	55.7	62.2	72.0	50.5	54.3	70.0	78.3	58.1	55.4	61.4	45.2	49.7	58.0
Ci et al. [8] ICCV'19 (+)(*)	46.8	52.3	**44.7**	**50.4**	52.9	68.9	49.6	46.4	**60.2**	78.9	**51.2**	50.0	54.8	40.4	**43.3**	52.7
Ours	46.3	52.2	47.3	50.7	55.5	67.1	49.2	**46.0**	60.4	71.1	51.5	50.1	54.5	**40.3**	43.7	**52.4**

proposed in [12]. Following Zhao et al. [40], we use 128 as the default number of channels of each graph convolutional layer.

Decouple self-connections. We first study the effect of decoupling self-connections in `full-sharing`, `pre-aggregation` and `post-aggregation`. We do not include `convolution-style` or `no-sharing` here as their self-connections are decoupled by definition. Since introducing a separate weight matrix for self-connections will bring more parameters, we increase the number of channels of the models without decoupling so that each pair of comparing methods have similar model sizes. Tab. 2 shows the results[4]. It is obvious that all these three weight sharing methods benefit from decoupling self-connections, and among

[4] `Pre-agg`, `post-agg` and `conv-style` are short for `pre-aggregation`, `post-aggregation` and `convolution-style` respectively

Table 6. Quantitative comparisons on the Human 3.6M dataset under **Protocol-2**. The P-MPJPEs are reported in millimeters. The best results are highlighted in bold and second underlined. **Legend:** (+) uses extra data from MPII dataset. (†) uses temporal information. (*) uses pose scales in both training and testing

Protocol # 2	Dire.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	Avg.
Lee et al. [18] ECCV'18 (†)	34.9	35.2	43.2	42.6	46.2	55.0	37.6	38.8	50.9	67.3	48.9	35.2	50.7	31.0	34.6	43.4
Hossain et al. [27] ECCV'18 (†)	36.9	37.9	42.8	40.3	46.8	46.7	37.7	36.5	48.9	52.6	45.6	39.6	43.5	35.2	38.5	42.0
Pavllo et al. [26] CVPR'19 (†)	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
Cai et al. [5] ICCV'19 (†)	35.7	37.8	36.9	40.7	39.6	45.2	37.4	34.5	46.9	50.1	40.5	36.1	41.0	29.6	33.2	39.0
Sun et al. [31] 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
Martinez et al. [21] 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
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. [24] 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
Li et al. [19] CVPR'19	35.5	39.8	41.3	42.3	46.0	48.9	**36.9**	37.3	51.0	60.6	44.9	40.2	44.1	33.1	36.9	42.6
Ci et al. [8] ICCV'19 (+)(*)	36.9	41.6	**38.0**	41.0	**41.9**	51.1	38.2	37.6	49.1	62.1	43.1	39.9	43.5	32.2	37.0	42.2
Ours	35.9	40.0	**38.0**	41.5	42.5	51.4	37.8	**36.0**	**48.6**	**56.6**	**41.8**	**38.3**	**42.7**	**31.7**	**36.2**	**41.2**

them `full-sharing` benefits the most. This demonstrates that decoupling self-connections in GCN is very important for 3D HPE. Thus, we will use decoupled weight sharing methods in the remaining experiments.

Weight sharing methods (controlled number of channels and computational complexity). Then we study the impact of different weight sharing methods on the 3D HPE performance. We first fix the number of channels of each graph convolutional layer to be 128 so that the shape of each weight matrix is the same for different weight sharing methods. This also means the computational complexities of all models are the same according to Tab. 1. Note we decouple the self-connections for `full-sharing`, `pre-aggregation` and `post-aggregation` for better performance. As we can see in Tab. 3, the weight sharing methods have a great impact on the localization error. Among them, `pre-aggregation` performs the best. We can observe that `no-sharing` has the smallest training loss, but does not perform as well as `pre-aggregation`. We conjecture that unsharing weights between any pair of input and output feature vectors gives it too much freedom and thus leads to overfitting. The vanilla graph convolution, i.e., `full-sharing`, has the smallest model size, which may bring it some disadvantage. Thus, we will fix the number of parameters of each model in the next ablation study.

Fig. 6 plots the loss and validation error of each method in the training phase. Compared with `full-sharing`, the training losses of other methods decrease very fast. `Pre-aggregation`, `post-aggregation` and `no-sharing` have significantly lower validations errors as the training goes on than `full-sharing` and `convolution-style`.

Weight sharing methods (controlled model size). Next, we fix the number of parameters to be about 4.2 M, 2.1 M or 1.05 M by merely changing the number of channels of each model. For example, to have a model size of about 4.2 M, the channels of the five methods listed in Tab 4 are 512, 415, 128, 128 and 100, respectively. We decouple the self-connections for `full-sharing`,

Table 7. Quantitative comparisons on the Human 3.6M dataset under **Protocol-1**. All approaches take 2D ground truth as input. The MPJPEs are reported in millimeters. **Legend:** (+) uses extra data from MPII dataset. (*) uses pose scales in both training and testing

Protocol # 1	Direct.	Discuss	Eating	Greet	Phone	Photo	Pose	Purch.	Sitting	SittingD.	Smoke	Wait	WalkD.	Walk	WalkT.	Avg.
Zhou et al. [43] ICCV'19 (+)	34.4	42.4	36.6	42.1	38.2	39.8	34.7	40.2	45.6	60.8	39.0	42.6	42.0	29.8	31.7	39.9
Ci et al. [8] ICCV'19 (+)(*)	36.3	38.8	29.7	37.8	34.6	42.5	39.8	32.5	36.2	39.5	34.4	38.4	38.2	31.3	34.2	36.3
Zhao et al. [40] CVPR'19	37.8	49.4	37.6	40.9	45.1	**41.4**	40.1	48.3	50.1	**42.2**	53.5	44.3	40.5	47.3	39.0	43.8
Ours	**36.8**	**40.3**	**33.0**	**36.3**	**37.5**	45.0	**39.7**	**34.9**	**40.3**	47.7	**37.4**	**38.5**	**38.6**	**29.6**	**32.0**	**37.8**

`pre-aggregation` and `post-aggregation`. The result is shown in Tab. 4. Given the same model size, `pre-aggregation` achieves the overall best performance. The performance of `full-sharing` and `convolution-style` do not improve consistently with the increase of model size.

Why `pre-aggregation` performs best? As mentioned, `full-sharing` is inferior because the shared feature transformation prevents it from learning different relational models between different body joints. On the other extreme is `no-sharing`, which assigns a different weight matrix to each pair of related body joints. This can be too much freedom as some common relational patterns do exist, especially as the human body is symmetric and has a star-shape. `Convolution-style`, `pre-aggregation` and `post-aggregation` are between these two extremes. `Convolution-style` mimics the image convolution and shares weights according to the displacement between two nodes on the graph. However, unlike image pixels, the relation between two nodes is not strictly *translation equivariant*, e.g., hand and elbow versus neck and head. `Pre-aggregation` provides the freedom to transform each node independently so that the transformed nodes will affect their neighbors in a unified way (via summation). It overcomes the limitation of `full-sharing` due to the independent transformation of each node. Compared with `no-sharing`, it requires the transformed nodes to share relations. `Post-aggregation` sums the features first, which will unavoidably lose information. By contrast, transforming the features first, as in `pre-aggregation`, provides a chance to retain or extract the most important features that are suitable for aggregation.

Our ablation study proves that using different weight sharing methods has a great impact on 3D HPE performance. It also reveals that choosing an appropriate weight sharing method according to the property of a problem is very important when applying GCN models.

4.3 Comparison with the State of the Art

Following Pavllo et al. [26], we use 2D poses provided by a pre-trained 2D cascaded pyramid network detector (CPN) [6] for benchmark evaluation. We use `pre-aggregation` as our weight sharing method as it outperforms the others in our ablation study. We set the initial learning rate 0.0001, the decay factor 0.95 per 4 epochs and the batch size 256. We add dropout with a factor 0.2 after

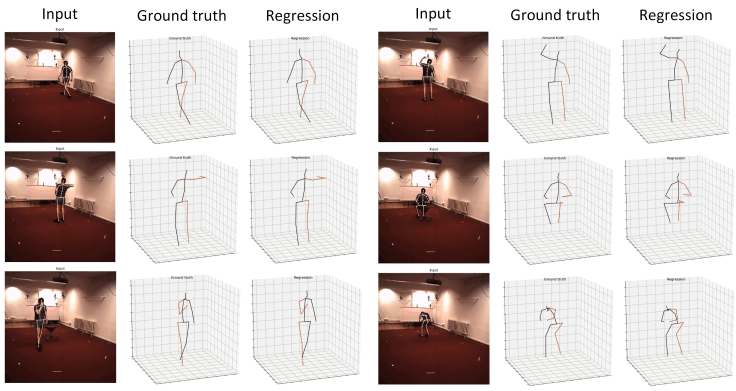

Fig. 7. Qualitative results of our approach on Human 3.6M

each graph convolutional layer to prevent overfitting. Following Zhao et al. [40], we integrate non-local blocks into our network to boost its performance. Note we do not use this kind of complementary tools in our ablation study to exclude their inference and ensure fairness. It takes about 10 hours to train our model for 30 epochs on a single Nvidia RTX 2080Ti GPU.

Tab. 5 and Tab. 6 show the results under two protocols respectively. There are work that exploiting temporal information [18, 27, 26, 5] to assist 3D regression and some using extra data to boost the performance [31, 39, 20, 24, 8]. By contrast, we aim to find the weight sharing method that is optimal for 3D HPE. Therefore, their ideas and strategies are complementary to ours and can also benefit our model. While our approach only takes 2D detections as input, it achieves the state-of-the-art performance by applying the optimal weight sharing method. This indicates that our model can effectively utilize relationships between different joints in the graph. Fig. 7 demonstrates some qualitative results of our approach on the Human3.6M dataset.

5 Conclusions

This paper has had a comprehensive and systematic study of weight sharing in GCNs for 3D HPE. After extensive ablation study and benchmark comparison, we make the following conclusions. (1) Weight sharing methods in GCNs have a great impact on the HPE performance. More parameters do not necessarily lead to better performance. (2) It is always beneficial to decouple self-connections. (3) Among all the variants of graph convolutions discussed in this paper, `pre-aggregation` is the optimal weight sharing method for 3D HPE.

Acknowledgments. This work was supported in part by Wei Tang's start-up funds from the University of Illinois at Chicago and the National Science Foundation (NSF) award CNS-1828265.

References

1. Agarwal, A., Triggs, B.: 3d human pose from silhouettes by relevance vector regression. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004. vol. 2, pp. II–II. IEEE (2004)
2. Agarwal, A., Triggs, B.: Recovering 3d human pose from monocular images. IEEE transactions on pattern analysis and machine intelligence **28**(1), 44–58 (2005)
3. Andriluka, M., Pishchulin, L., Gehler, P., Schiele, B.: 2d human pose estimation: New benchmark and state of the art analysis. In: Proceedings of the IEEE Conference on computer Vision and Pattern Recognition. pp. 3686–3693 (2014)
4. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
5. Cai, Y., Ge, L., Liu, J., Cai, J., Cham, T.J., Yuan, J., Thalmann, N.M.: Exploiting spatial-temporal relationships for 3d pose estimation via graph convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2272–2281 (2019)
6. Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., Sun, J.: Cascaded pyramid network for multi-person pose estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7103–7112 (2018)
7. Cheng, Y., Yang, B., Wang, B., Yan, W., Tan, R.T.: Occlusion-aware networks for 3d human pose estimation in video. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 723–732 (2019)
8. Ci, H., Wang, C., Ma, X., Wang, Y.: Optimizing network structure for 3d human pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2262–2271 (2019)
9. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems. pp. 3844–3852 (2016)
10. Fang, H.S., Xu, Y., Wang, W., Liu, X., Zhu, S.C.: Learning pose grammar to encode human body configuration for 3d pose estimation. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
11. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1263–1272. JMLR. org (2017)
12. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256 (2010)
13. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in neural information processing systems. pp. 1024–1034 (2017)
14. Ionescu, C., Li, F., Sminchisescu, C.: Latent structured models for human pose estimation. In: 2011 International Conference on Computer Vision. pp. 2220–2227. IEEE (2011)
15. Ionescu, C., Papava, D., Olaru, V., Sminchisescu, C.: Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE transactions on pattern analysis and machine intelligence **36**(7), 1325–1339 (2013)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
18. Lee, K., Lee, I., Lee, S.: Propagating lstm: 3d pose estimation based on joint interdependency. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 119–135 (2018)
19. Li, C., Lee, G.H.: Generating multiple hypotheses for 3d human pose estimation with mixture density network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9887–9895 (2019)
20. Luvizon, D.C., Picard, D., Tabia, H.: 2d/3d pose estimation and action recognition using multitask deep learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5137–5146 (2018)
21. Martinez, J., Hossain, R., Romero, J., Little, J.J.: A simple yet effective baseline for 3d human pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2640–2649 (2017)
22. Moon, G., Chang, J.Y., Lee, K.M.: Camera distance-aware top-down approach for 3d multi-person pose estimation from a single rgb image. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 10133–10142 (2019)
23. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European conference on computer vision. pp. 483–499. Springer (2016)
24. Pavlakos, G., Zhou, X., Daniilidis, K.: Ordinal depth supervision for 3d human pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7307–7316 (2018)
25. Pavlakos, G., Zhou, X., Derpanis, K.G., Daniilidis, K.: Coarse-to-fine volumetric prediction for single-image 3d human pose. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7025–7034 (2017)
26. Pavllo, D., Feichtenhofer, C., Grangier, D., Auli, M.: 3d human pose estimation in video with temporal convolutions and semi-supervised training. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7753–7762 (2019)
27. Rayat Imtiaz Hossain, M., Little, J.J.: Exploiting temporal information for 3d human pose estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 68–84 (2018)
28. Rogez, G., Rihan, J., Ramalingam, S., Orrite, C., Torr, P.H.: Randomized trees for human pose detection. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8. IEEE (2008)
29. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference. pp. 593–607. Springer (2018)
30. Sharma, S., Varigonda, P.T., Bindal, P., Sharma, A., Jain, A.: Monocular 3d human pose estimation by generation and ordinal ranking. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2325–2334 (2019)
31. Sun, X., Shang, J., Liang, S., Wei, Y.: Compositional human pose regression. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2602–2611 (2017)
32. Sun, X., Xiao, B., Wei, F., Liang, S., Wei, Y.: Integral human pose regression. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 529–545 (2018)
33. Tang, W., Wu, Y.: Does learning specific features for related parts help human pose estimation? In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1107–1116 (2019)

34. Tang, W., Yu, P., Wu, Y.: Deeply learned compositional models for human pose estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 190–206 (2018)
35. Tekin, B., Katircioglu, I., Salzmann, M., Lepetit, V., Fua, P.: Structured prediction of 3d human pose with deep neural networks. arXiv preprint arXiv:1605.05180 (2016)
36. Tekin, B., Márquez-Neila, P., Salzmann, M., Fua, P.: Learning to fuse 2d and 3d image cues for monocular body pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3941–3950 (2017)
37. Wang, J., Huang, S., Wang, X., Tao, D.: Not all parts are created equal: 3d pose estimation by modeling bi-directional dependencies of body parts. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 7771–7780 (2019)
38. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: Thirty-second AAAI conference on artificial intelligence (2018)
39. Yang, W., Ouyang, W., Wang, X., Ren, J., Li, H., Wang, X.: 3d human pose estimation in the wild by adversarial learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5255–5264 (2018)
40. Zhao, L., Peng, X., Tian, Y., Kapadia, M., Metaxas, D.N.: Semantic graph convolutional networks for 3d human pose regression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3425–3435 (2019)
41. Zhao, X., Ning, H., Liu, Y., Huang, T.: Discriminative estimation of 3d human pose using gaussian processes. In: 2008 19th International Conference on Pattern Recognition. pp. 1–4. IEEE (2008)
42. Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: A review of methods and applications. arXiv preprint arXiv:1812.08434 (2018)
43. Zhou, K., Han, X., Jiang, N., Jia, K., Lu, J.: Hemlets pose: Learning part-centric heatmap triplets for accurate 3d human pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2344–2353 (2019)
44. Zhou, X., Huang, Q., Sun, X., Xue, X., Wei, Y.: Towards 3d human pose estimation in the wild: a weakly-supervised approach. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 398–407 (2017)