This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Optimizing Network Structure for 3D Human Pose Estimation

Hai Ci[1], Chunyu Wang[2], Xiaoxuan Ma[1], and Yizhou Wang[1,3,4]
[1] Computer Science Dept., Peking University
[2] Microsoft Research, Asia
[3] Deepwise AI Lab [4] Peng Cheng Laboratory
{cihai, maxiaoxuan, yizhou.wang}@pku.edu.cn, chnuwa@microsoft.com

Abstract

A human pose is naturally represented as a graph where the joints are the nodes and the bones are the edges. So it is natural to apply Graph Convolutional Network (GCN) to estimate 3D poses from 2D poses. In this work, we propose a generic formulation where both GCN and Fully Connected Network (FCN) are its special cases. From this formulation, we discover that GCN has limited representation power when used for estimating 3D poses. We overcome the limitation by introducing Locally Connected Network (LCN) which is naturally implemented by this generic formulation. It notably improves the representation capability over GCN. In addition, since every joint is only connected to a few joints in its neighborhood*, it has strong generalization power. The experiments on public datasets show it: (1) outperforms the state-of-the-arts; (2) is less data hungry than alternative models; (3) generalizes well to unseen actions and datasets.*

1. Introduction

A 3D human pose is naturally represented by a skeletal graph parameterized by the 3D locations of the body joints such as elbows and knees. See Figure 1. When we project a 3D pose to a 2D image by the camera parameters, the depth of all joints is lost. The task of 3D pose estimation solves the inverse problem of depth recovery from 2D poses. This is an ambiguous problem because multiple 3D poses may correspond to the same 2D pose after projection. But it is practically solvable because 3D poses lie on a low-dimensional manifold which provides strong structural priors to reduce the ambiguities [36].

A recent work [18] introduces a variant of Fully Connected Network (FCN) to map a 2D pose to 3D space. It achieves promising results on the benchmark datasets. But we experimentally find it has degraded cross action and cross dataset performance which is also observed in [17]. This may be attributed to the dense connections in FCN and limited variations in the training set, which increase the chance of linking unrelated joints. This also contradicts the fact that human can perceive a 3D joint by only seeing the 2D joints in its neighborhood.

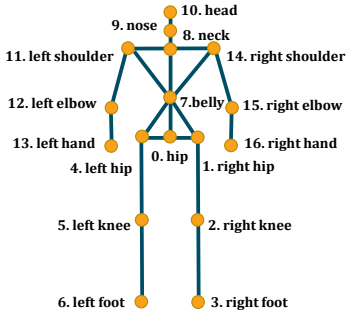

Figure 1. Human pose is essentially a skeletal graph model consisting of a number of body joints linked by the bones. The figure shows the graph model used in this paper.

The Graph Convolutional Network (GCN) [3, 5, 6, 8, 10, 14, 27, 28, 34] is a promising alternative for 3D pose estimation because it only aggregates the features of the *"selected"* nodes to compute features for a node of interest. See Figure 2 (b) for conceptual illustration. We propose a generic formulation in which both GCN and FCN are special cases. We factor the Laplacian operator in GCN [6] into the product of a *structure matrix* which encodes the dependence relation among the nodes, and a *weight matrix* which defines how to aggregate the dependent features. Based on the formulation, we discover two main limitations of GCN. First, the weight matrix has an inherent weight sharing scheme which harms the model's representation ability. For example, in Figure 2 (b), the learnable operators T are the same for all nodes. Second, the structure matrix is directly determined by node distance which lacks flexibility to support customized node dependence.

In this work, we present *Locally Connected Network* (LCN) on top of our generic formulation to overcome the limitations of GCN. First, we discard the weight sharing scheme by freeing all of the parameters in the weight ma-

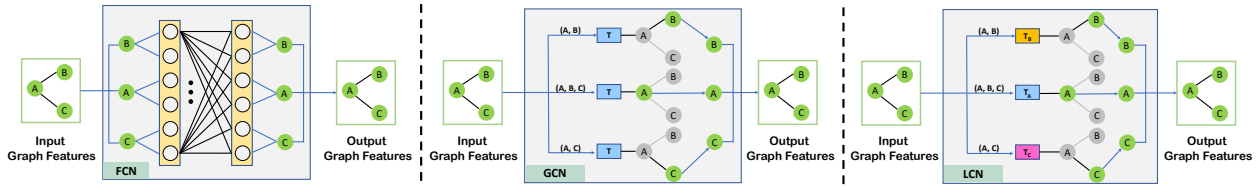

Figure 2. Conceptual difference between FCN, GCN and LCN. The input is a graph with three nodes and two edges. Each node is associated with a 2-dimensional feature vector, *e.g.* its 2D location in the task of 3D pose estimation. **(a)** In FCN, the input features of different nodes are mixed by dense connections making every output feature dependent on the input features of all nodes. **(b)** In GCN, the output features of a node only depend on the nodes which are regarded as "related" determined by the Laplacian matrix. For example, when we compute the features for the node B (top branch), it only takes the features of nodes A and B as input. Different nodes share the same filter T (the blue rectangles). **(c)** In LCN, each node has a different filter. In addition, the dependence between the joints are specified in a more straightforward and flexible way than GCN which we will discuss in detail in the paper.

trix to fully unleash its representation ability. For example, see Figure 2 (c) that the three nodes (branches) have their own operators T_A, T_B and T_C. Second, we propose a more straightforward and flexible way to construct the structure matrix according to human anatomy which allows us to freely determine the joint dependence. In summary, LCN combines the advantages of GCN and FCN. First, it has sparse joint connections among the nodes which reduces the risk of over-fitting the datasets with limited variations and enhances the model's generalization ability. Second it has strong representation ability by freeing all of the learnable parameters in the weight matrix.

We evaluate LCN on the public datasets H36M [11] and MPI-INF-3DHP [19]. It outperforms FCN, GCN and the state-of-the-arts on both datasets. First, when the input 2D poses are the ground truth, the 3D pose error of LCN is smaller than that of FCN [18] which has dense connections. This indicates LCN has sufficient representation ability although the connections are sparse. Second, when the 2D poses are estimated from images and inaccurate, the 3D pose error of LCN is smaller than the state-of-the-arts although some of them even use additional training datasets. Third, when we apply our model learned on H36M to the MPI-INF-3DHP dataset, it achieves better performance than the state-of-the-arts. The promising cross-datasets results suggest that the generalization ability of our approach is strong because the unrelated joints are not connected which reduces the risk of over-fitting.

2. Related Work

We classify the 3D human pose estimators into *unsupervised* and *supervised* classes. Unsupervised approaches [16, 30, 35, 36, 37] explicitly model the relations between 2D features, 3D model and camera parameters, and optimize the parameters of the 3D model such that its projection matches the 2D features. For example, Lee and Cohen [16] use data-driven MCMC to search the high dimensional parameter space of the 3D model to maximize the likelihood of the image features including skin color, contour and ridges. Some works [30, 36, 37] propose to estimate 3D poses from 2D poses by minimizing the distance between the projected 2D poses and the estimated 2D poses. They use limb length priors to reduce the ambiguity. Ijaz and Black [2] propose to learn a pose-dependent bending angle prior to prevent invalid poses. Later works [25, 36] propose to learn a low dimensional representation to suppress illegitimate estimations.

Another class of approaches (*e.g.*, [12, 22, 23, 24, 29, 31, 33, 38, 39]) treat 3D pose estimation as a supervised regression problem. Agarwal and Triggs [1] extract shape descriptors from images and learn a relevance vector regression machine to map the descriptors to 3D poses. Similarly, Paul *et al.* [21] extract edge histograms and hash the features to 3D poses. Recently, since 2D human pose estimation is relatively accurate, many approaches (*e.g.*, [18, 22, 26]) have focused on learning the mapping from 2D poses to 3D and achieved the state-of-the-art results. In particular, Martinez *et al.* [18] propose a variant of FCN to map a 2D pose to 3D. Similarly, Sun *et al.* [29] propose an end-to-end learning method to estimate 3D poses from 2D pose heat maps. Our work belongs to this class and learns a mapping from 2D poses to 3D poses. But the difference is that our work focuses on optimizing the joint dependence based on human anatomy which is not addressed in the previous works.

3. Reformulate GCN

We first revisit a popular implementation of GCN [6]. Then we factor the Laplacian operator in GCN into the product of a structure matrix and a weight matrix. Based on the formulation, we can clearly see why GCN has limitations when it is used for 3D pose estimation. Then we obtain a more generic model based on the formulation which overcomes the limitations. Finally, we discuss the relation between the generic model, FCN and GCN.

3.1. Revisit GCN

GCN processes features defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ where $\mathcal{V}$ denotes a set of N nodes, $\mathcal{E}$ denotes a set of edges and $\boldsymbol{W} \in \mathcal{R}^{N\times N}$ denotes a weighted adjacency matrix encoding the dependence between the nodes. We denote $\boldsymbol{x} \in \mathcal{R}^N$ as a feature defined on the N nodes where each dimension corresponds to one node. There are M features in total for each node. We put all features of all nodes into a big matrix $\boldsymbol{X} \in \mathcal{R}^{M\times N}$. We use $\boldsymbol{X}(:, n)$ to denote the features of the n_{th} node and use $\boldsymbol{X}(m, :)$ to denote the m_{th} feature of all nodes. We use $\boldsymbol{X}_r$ and $\boldsymbol{X}_c$ to denote the flattened copies of $\boldsymbol{X}$ in row and column major order, respectively.

The essential operator in GCN is Graph Laplacian. The combinatorial definition is $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} \in \mathcal{R}^{N\times N}$ where $\boldsymbol{D}$ is the diagonal degree matrix with $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{W}_{ij}$. The Laplacian can be diagonalized by the Fourier basis $\boldsymbol{U} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_N] \in \mathcal{R}^{N\times N}$ such that $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$. The graph Fourier transform of a feature vector $\boldsymbol{x} \in \mathcal{R}^N$ is

$$\boldsymbol{y} = g_\theta(\boldsymbol{L})\boldsymbol{x} = g_\theta(\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top)\boldsymbol{x} = \boldsymbol{U}g_\theta(\boldsymbol{\Lambda})\boldsymbol{U}^\top\boldsymbol{x}, \quad (1)$$

where $g_\theta(\boldsymbol{\Lambda})$ can be parameterized with the use of a polynomial filter $g_\theta(\boldsymbol{\Lambda}) = \sum_{k=0}^{K-1} \theta_k \boldsymbol{\Lambda}^k$ where K is the expansion order and θ_k is the learnable parameter.

3.2. Reformulation

We obtain an output feature vector $\boldsymbol{y} \in \mathcal{R}^N$ for N nodes by applying a filter g_θ to the input features $\boldsymbol{X}$

$$\boldsymbol{y} = g_\theta(\boldsymbol{X}) = \sum_{k=0}^{K-1}\sum_{m=1}^{M} \theta_{km} \cdot \boldsymbol{L}^k \cdot \boldsymbol{X}(m, :)^\top \quad (2)$$

The filter has different θ for different feature dimensions (M in total) and expansion orders (K in total). *However, different nodes in the graph share the same filter* θ. See the above equation that the same set of θ's is used for computing different dimensions of $\boldsymbol{y}$ which correspond to different nodes. Let us take a closer look at the output feature corresponding to the q_{th} node y_q, which is the q_{th} dimension of $\boldsymbol{y}$, by modifying the above formulation

$$y_q = \sum_{k=0}^{K-1}\sum_{m=1}^{M} \theta_{km} \cdot \boldsymbol{L}^k(q, :) \cdot \boldsymbol{X}(m, :)^\top, \quad (3)$$

Based on the definition of the Laplacian matrix L^k [9, 6], if the minimum number of edges connecting the joints i and j (*i.e.* their distance on the graph) is larger than k, then $L^k(i, j) = 0$. So the above formulation can be interpreted as aggregating features from the neighboring nodes whose distance is less than K. The feature y_q can be written as:

$$\begin{aligned} y_q &= \sum_{k=0}^{K-1}\sum_{m=1}^{M} \boldsymbol{X}(m, :) \cdot \boldsymbol{L}^k(q, :)^\top \cdot \theta_{km} \\ &= \sum_{k=0}^{K-1}\sum_{m=1}^{M} \boldsymbol{X}(m, :) \cdot \left(\boldsymbol{L}^k(q, :)^\top \odot \boldsymbol{\Theta}_{km}\right) \end{aligned} \quad (4)$$

where $\boldsymbol{\Theta}_{km} \in \mathcal{R}^{N\times 1}$ with θ_{km} repeated N times. Then we reformulate the inner summation over m by a more compact matrix form and obtain

$$\begin{aligned} y_q &= \sum_{k=0}^{K-1} \left[\Big\|_{m=1}^{M} \boldsymbol{X}(m, :)\right] \left[\Big\|_{m=1}^{M} \left(\boldsymbol{L}^k(:, q) \odot \boldsymbol{\Theta}_{km}\right)\right] \\ &= \sum_{k=0}^{K-1} \boldsymbol{X}_r \left[\left(\Big\|_{m=1}^{M} \boldsymbol{L}^k(:, q)\right) \odot \left(\Big\|_{m=1}^{M} \boldsymbol{\Theta}_{km}\right)\right] \\ &= \sum_{k=0}^{K-1} \boldsymbol{X}_r \left(\boldsymbol{S}_q^k \odot \boldsymbol{W}_q^k\right), \end{aligned} \quad (5)$$

where $\|$ denotes concatenation. $\boldsymbol{S}_q^k \in \mathcal{R}^{MN\times 1}$ is a vector with $\boldsymbol{L}^k(:, q)$ repeated N times. It contains the neighbourhood information of node q. We can generalize the formulation to all nodes in the graph as follows:

$$\boldsymbol{y} = \sum_{k=0}^{K-1} \boldsymbol{X}_r \left(\boldsymbol{S}^k \odot \boldsymbol{W}^k\right) \quad (6)$$

where both $\boldsymbol{S}^k$ and $\boldsymbol{W}^k$ are 2D matrices with the shape of $MN \times N$. Specifically, $\boldsymbol{S}_q^k$ is the the q_{th} column of $\boldsymbol{S}^k$ and $\boldsymbol{W}_q^k$ is the q_{th} column of $\boldsymbol{W}^k$.

3.3. Limitations

The main limitation of GCN lies in the weight sharing scheme in $\boldsymbol{W}^k$. First, we can see from Eq. (5) that each $\boldsymbol{W}_q^k$ has only M unique parameters because each $\boldsymbol{\Theta}_{km}$ has one unique parameter. Recall $\boldsymbol{\Theta}_{km}$ is obtained by repeating θ_{km} N times. Second, GCN computes features for different nodes using the same set of parameters, *i.e.* $\boldsymbol{W}_q^k = \boldsymbol{W}_p^k$ for all p and q. See Figure 2 (b) that the three nodes share the same operator T. The weight sharing in $\boldsymbol{W}$ enhances the model's generalization ability on the semi-supervised node classification task [6, 14] where the number of training data is small. However, it will harm its representation capability when it is used for 3D pose estimation because each joint needs to aggregate features from its neighbors in unique ways in order to infer its 3D location.

Another minor limitation lies in the way GCN constructs the structure matrix $\boldsymbol{S}$. As shown in Eq.(6), it treats all neighbors of the same distance to the node of interest without discrimination, leaving us no flexibility to freely connect the joints of arbitrary distance.

3.4. Generalization

We obtain a more generic model by dropping the structure constraints in $\boldsymbol{S}^k$ and $\boldsymbol{W}^k$:

$$\boldsymbol{y} = \boldsymbol{X}\left(\boldsymbol{S} \odot \boldsymbol{W}\right) \tag{7}$$

We will show in the next subsection that both FCN and GCN are special cases of the generic model. More importantly, on top of the generic model, we propose a straightforward approach to construct the structure matrix $\boldsymbol{S}$ to directly reflect the joint dependence. See Figure 3 for the conceptual illustration. Together with the constraint-free weight matrix, we obtain the LCN model which has enhanced representation capability. We will describe the approach in more detail in the next section.

3.5. Relation to FCN and GCN

We now discuss the relation between the above generic model and FCN. First, FCN is essentially represented by the product of a weight matrix and a feature matrix. Customizing the generic model (7) into FCN can be achieved by setting all values in $\boldsymbol{S}$ to be 1

$$\boldsymbol{y} = \boldsymbol{X}\left(\boldsymbol{1} \odot \boldsymbol{W}\right), \tag{8}$$

which means all of the nodes are connected. The parameters in $\boldsymbol{W}$ are all free, and are learned end-to-end from training datasets. We can see that FCN does not take advantage of the graph structures but simply connects all nodes. See Figure 2 (a). We observe in experiments this impacts its generalization ability.

GCN can be obtained by initializing $\boldsymbol{S}^k$ and $\boldsymbol{W}^k$ according to Eq. (4 and 5) and stacking $\boldsymbol{S}^k, \boldsymbol{W}^k, (k = 0, \cdots, K-1)$ vertically to generate $\boldsymbol{S}$ and $\boldsymbol{W}$ in Eq. (7). The input features $\boldsymbol{X}$ should also be repeated for K times. So GCN is a specialization of the generic model.

4. Locally Connected Network

LCN is also a specialization of the generic model in Eq.(7). But it combines the advantages of FCN and GCN: (1) it has adequate representation ability as there are no constraints in $\boldsymbol{W}$; (2) it has strong generalization ability as the joints are sparsely connected in $\boldsymbol{S}$.

We first present how we implement an LCN layer as shown in Figure 3, where the core is to construct $\boldsymbol{S}$ and learn $\boldsymbol{W}$. Then we describe how we build a deep network for 3D pose estimation using the LCN layers. Note that $\boldsymbol{S}$ is shared for all LCN layers which is offline constructed based on the specified joint dependence. Different layers have their own $\boldsymbol{W}$ which is learned end-to-end.

4.1. Joint Dependence

Our principal for determining the joint dependence relation is "locality" which means each joint only depends on those which have short *manifold distance* to it. The manifold distance between two joints is defined by their distance on the graph (*i.e.* the skeleton body model). For instance, the manifold distance between the left-elbow and left-hand is one because they are directly connected. See Figure 1 for the definition of our skeleton model.

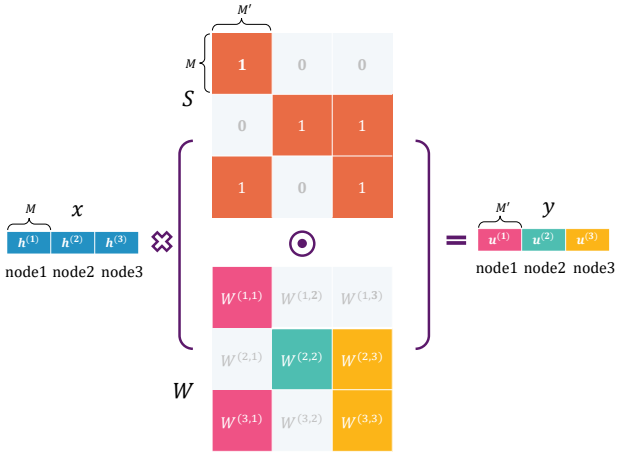

Figure 3. Illustration of an LCN layer. Suppose we have 3 nodes whose features are represented as the blue bars. The weight matrix $\boldsymbol{W}$ and structure matrix $\boldsymbol{S} \in \mathcal{R}^{3M \times 3M'}$ are evenly divided into 3×3 blocks. We fill zeros in the structure matrix at appropriate locations to remove the dependence between the corresponding pairs of joints. For example, output features of node 1 only depend on the input features of nodes 1 and 3.

In this paper, we investigate a simple way to determine the joint dependence, *i.e.* each joint depends on the neighbors whose manifold distance to the joint is less equal than K. For instance, when $K = 2$, the joint 12 is dependent on the joints $13, 11, 8$ and 7. We denote this approach as LCN (K-NN). We will evaluate the 3D estimation accuracy when different K is used. It is worth noting that there are other alternatives to determine the joint dependence but it is not the focus of this paper to explore them. The point is that our model can leverage any pre-determined joint dependence relation as a prior.

4.2. Structure Matrix

We construct $\boldsymbol{S}$ to reflect the above defined joint dependence which is very straightforward due to the reformulation. If joint j is dependent on the joint i, then we set the (i, j) block of $\boldsymbol{S}$ to be ones. Otherwise, we set it to be zeros. In this way, the features of the i_{th} node will not contribute to the computation of the output features of the j_{th} node. See Figure 3 for the conceptual illustration.

We now mathematically verify that the constructed S has the desired property. It is helpful to also refer to Figure 3. For notation simplicity, we use $\boldsymbol{h}^{(i)}$ to denote the M input features of the i_{th} joint $\boldsymbol{h}^{(i)} = \boldsymbol{X}(:, i)^\top$ where $\boldsymbol{h}^{(i)} \in \mathcal{R}^{1 \times M}$. Similarly, we use $\boldsymbol{u}^{(j)}$ to denote the M' output

features of the node j where $\boldsymbol{u}^{(j)} \in \mathcal{R}^{1\times M'}$. The operation in LCN is then defined as:

$$\sum_{i=1}^{N} \boldsymbol{h}^{(i)}(\boldsymbol{S}^{(i,j)} \odot \boldsymbol{W}^{(i,j)}) = \boldsymbol{u}^{(j)} \tag{9}$$

We can see that if $\boldsymbol{S}^{(i,j)}$ is zero, then $\boldsymbol{h}^{(i)}$ will not contribute to the computation of $\boldsymbol{u}^{(j)}$ as we expect. We call this operation, which is illustrated in Figure 2 as one **LCN layer** which generates output features for all nodes.

We can further replace the ones in $\boldsymbol{S}$ by continuous values to reflect the "importance" among the joints. Instead of manually specifying the values, we can also learn them from the training dataset end-to-end by replacing the non-zero values in $\boldsymbol{S}$ with learnable parameters.

4.3. Application to 3D Pose Estimation

For the task of 3D pose estimation, we build a deep neural network which uses the above discussed LCN layer as the basic building block. Inspired by the network structure proposed by Martinez *et al*. [18] and Defferrard *et al*. [6], our LCN has several cascaded blocks with each consisting of two LCN layers, interleaved with BN, LeakyReLU and Dropout. Each block is wrapped in a residual connection. The number of output features M' in each LCN layer is set to be 64. It is worth noting that different layers share the same structure matrix but have different weight matrices.

The input to the LCN network is the 2D locations of the body joints and the output is the corresponding 3D locations. We use L_2 loss between the outputs and the ground truth. The network can be trained end-to-end.

5. Experiments

5.1. Datasets and Metrics

We evaluate our approach on the public datasets H36M [11] and MPI-INF-3DHP [19]. For H36M, we use subjects $1, 5, 6, 7, 8$ for training and subjects $9, 11$ for testing following [18]. We train a single model for all actions. We compute the Mean Per Joint Position Error (MPJPE) between the ground truth and the 3D pose estimation [18] after aligning the mid-hip joints. We refer to this as protocol #1. We also report results when the estimations are aligned with the ground truth via a rigid transformation. We call this post-processing protocol #2. For MPI-INF-3DHP, we directly apply the model trained on H36M to the test set to validate the generalization ability of our approach. We use the metrics of average PCK and AUC. Following previous works [40, 17], we assume global scale is known for experimental evaluation.

5.2. Implementation Details

Coordinate system Denote the 3D location of a joint in the world Coordinate System (CS) as $\boldsymbol{P}_{\text{W}}$. We first transform it to the *camera* CS by the extrinsic camera parameters: $\boldsymbol{P}_{\text{C}} = \boldsymbol{R}(\boldsymbol{P}_{\text{W}} - \boldsymbol{T})$. Then we project $\boldsymbol{P}_{\text{C}} = (X_c, Y_c, Z_c)$ to the *pixel* CS: $\boldsymbol{P}_{\mathbf{p}} = (u, v)$ using the intrinsic camera parameters where $u = f\frac{X_c}{Z_c} + c_x$ and $v = f\frac{Y_c}{Z_c} + c_y$.

Model	Coordinate System	Error
Martinez *et al*. [18]	camera CS	67.50
Martinez *et al*. [18]	pixel CS	63.21
LCN (3-NN)	camera CS	62.95
LCN (3-NN)	pixel CS	57.56

Table 1. The average 3D pose estimation error of two models when using different coordinate systems on the H36M dataset under protocol #1. The 2D poses used for training and testing are estimated by SH [20] which is trained on MPII.

Many works [18] learn a mapping from $\boldsymbol{P}_{\mathbf{p}}$ to $\boldsymbol{P}_{\text{C}}$. However, it is impossible to determine the pose scale from a single image because a "large-but-far" person may have the same projection (u, v) as a "small-but-close" person. Although they claim to estimate 3D poses in the camera CS (which has scale information), they actually assume the pose scales and focal lengths in the training and testing sets are similar which is a limitation.

We propose to remove the scale in $\boldsymbol{P}_{\text{C}}$. Specifically, we seek for a scalar λ which makes $\lambda\boldsymbol{P}_{\text{C}}$ have similar scale as $\boldsymbol{P}_{\mathbf{p}}$ by minimizing $\|\lambda\hat{\boldsymbol{P}}_{\text{C}} - \hat{\boldsymbol{P}}_{\mathbf{p}}\|_2$ where $\hat{\boldsymbol{P}}_{\text{C}}$ and $\hat{\boldsymbol{P}}_{\mathbf{p}}$ denote the poses centered around their pelvis joints. We only use the x, y coordinates of the poses when computing λ.

We propose to estimate $\lambda\boldsymbol{P}_{\text{C}}$ from $\boldsymbol{P}_{\mathbf{p}}$ which is independent of the actual body scale. For evaluation purpose, the estimation is transformed back to the camera CS using λ. We find this coordinate system benefits for different models as shown in Table 1.

2D detections The input to our network is 2D poses estimated by the Stacked Hourglass (SH) [20] model trained on the MPII dataset. In some experiments, we also finetune SH on the H36M dataset which will be described clearly.

Training details We train our model for 200 epochs using Adam, a start learning rate of 0.001 and exponential decay, using mini-batches of size 200. During testing, it can process about 47K samples per second using batch processing mode (200 samples per batch) on a single GTX 1080ti GPU.

5.3. Baselines

The first baseline is an *FCN* variant proposed by Martinez *et al*. [18]. The second baseline is a *GCN* variant proposed by Defferrard *et al*. [6], which is originally designed for semi-supervised node classification tasks. We slightly modify its public implementation to make it suitable for 3D pose estimation. We also evaluate different variants of *LCN*.

Model	Error	# Parameters
Martinez *et al*. [18]	63.21	$4.3M$
Defferrard *et al*. [6]	66.37	$0.05M$
LCN (1-NN)	58.77	$0.95M$
LCN (2-NN)	57.73	$1.85M$
LCN (3-NN)	57.56	$2.92M$
LCN (4-NN)	58.37	$3.96M$
LCN (2-NN)-Learn	57.80	$1.85M$+111
LCN -Learn	59.22	$4.3M$+289

Table 2. The average 3D pose estimation error of different models on the H36M dataset under protocol #1. The 2D poses used for training and testing are estimated by SH [20] which is trained on MPII. The third column shows the number of learnable parameters of the "weight matrix" + "structure matrix".

In particular, we investigate LCN (K-NN) where K ranges from 1 to 4. In addition, for LCN (K-NN), we can replace the blocks of ones in S by learnable parameters to reflect its degree of dependence on the other joints. We denote this approach as *LCN (K-NN)-Learn*. We also investigate an approach when we do not specify $\boldsymbol{S}$ according to human anatomy but completely learn it from data. This approach is denoted as *LCN-Learn*.

5.4. Comparison to the Baselines

The results shown in Table 2 are obtained when the 2D poses are estimated by the SH model [20] which is only trained on the MPII dataset. First, the FCN model [18] gets an error of 63.21mm. Second, directly using GCN [6] increases the error to 66.37mm. The degraded accuracy should be attributed to the weight sharing scheme in GCN which harms the model's representation ability.

LCN obtains a smaller error than its two rivals. When a joint only depends on its nearest neighboring joints, *i.e. LCN (1-NN)*, the error is already reasonably small. This indicates the 3D location of a joint can be estimated by observing a small number of 2D joints in its neighborhood. Increasing the number of dependent joints further decreases the error when it is smaller than four. Then the error begins to increase. This may be because the redundant connections with the unrelated joints have negative impacts on the model's generalization ability.

The estimation error of *LCN (2-NN)-Learn* is similar to *LCN (2-NN)*. Figure 4 shows the learned structure matrix. First, the learned joint dependence is approximately symmetric which agrees with our common sense. Second, the strongest dependence is usually from the directly connected joints. When we only rely on data to learn the structure matrix, *i.e. (LCN-Learn)*, it gets a larger error which implies it is important to leverage anatomy priors to the model to prevent it from over-fitting. Although learning structure matrix

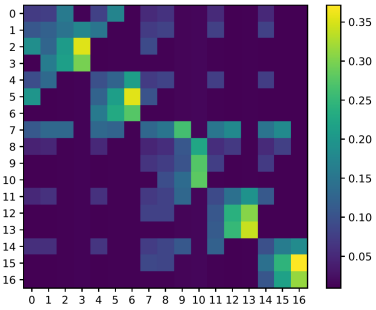

Figure 4. The structure matrix learned by LCN (2-NN)-Learn. X-axis and y-axis indicate indices of joints.

$\boldsymbol{S}$ doesn't show a better performance, we find it has better robustness to noise, which will be demonstrated later in section 5.6.

5.5. Comparison to the State-of-the-arts

The H36M Dataset We compare our approach (*LCN (3-NN)*) to the state-of-the-arts on the H36M dataset. The 2D pose model SH is first pre-trained on the MPII dataset and then finetuned on H36M. The results using the protocol #1 and #2 are shown in Table 3 and 5, respectively.

Protocol #1: First, when the 2D poses are estimated by Stacked Hourglass [20], the 3D error is 52.7mm which is smaller than previous state-of-the-arts. Second, our approach improves the lower bound of [18] by a notable margin when trained on GT 2D poses.

Protocol #2: Our approach also achieves comparable results with previous state-of-the-art [22] even though they use additional ordinal annotation.

The MPI-INF-3DHP Dataset We apply the model trained on H36M to the test set of MPI-INF-3DHP. Table 4 shows the results. First, we can see that the FCN model [18] has poor results which indicates that the dense connections in FCN impact the generalization ability. Second, our approach outperforms the previous state-of-the-arts [19, 40] which are dedicated to address the generalization issue across different datasets. In particular, Pavlakos *et al*. [22] use additional annotations and datasets when they train their model. This validates the strong generalization capability of our model to new datasets.

5.6. The Generalization Ability

We systematically evaluate the generalization ability of our approach from three aspects.

Cross Actions We train our model on *ONE* of the 15 actions in the H36M dataset and test on all actions. Figure 5 shows the results. We can see that the MPJPE of our approach is about 20mm smaller than that of [18]. Recall that the gap is about 10mm when we use all actions for training.

Method	Dire.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD	Smoke	Wait	WalkD	Walk	WalkT	Avg
Pavlakos *et al.* [23]	67.4	71.9	66.7	69.1	72.0	77.0	65.0	68.3	83.7	96.5	71.7	65.8	74.9	59.1	63.2	71.9
Tekin *et al.* [32]	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
Katircioglu *et al.* [13]	54.9	63.3	57.3	62.3	70.3	77.4	56.7	57.1	79.0	97.1	64.3	61.9	67.1	49.8	62.3	65.4
Zhou *et al.* [40]	54.8	60.7	58.2	71.4	62.0	65.5	53.8	55.6	75.2	111.6	64.2	66.1	51.4	63.2	55.3	64.9
Sun *et al.* [29]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	64.1
Martinez [18]	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
Fang *et al.* [7]	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
Yang *et al.* [39]	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
Pavlakos *et al.* [22]	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
Ours	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**
Martinez *et al.* (GT) [18]	37.7	44.4	40.3	42.1	48.2	54.9	44.4	42.1	54.6	58.0	45.1	46.4	47.6	36.4	40.4	45.5
Ours (GT)	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**

Table 3. 3D estimation errors of different methods on H36M under protocol #1. GT means the 2D poses are from the ground truth.

	Training Data	GS	noGS	Outdoor	ALL (PCK)	ALL (AUC)
Martinez [18]	H36m	49.8	42.5	31.2	42.5	17.0
Mehta [19]	H36m	70.8	62.3	58.8	64.7	31.7
Yang [39]	H36m+MPII	-	-	-	69.0	32.0
Zhou [40]	H36m+MPII	71.1	64.7	72.7	69.2	32.5
Luo [17]	H36m	71.3	59.4	65.7	65.6	33.2
Pavlakos [22]	H36m+MPII+LSP	-	-	-	44.3	19.8
Pavlakos* [22]	H36m+MPII+LSP	**76.5**	63.1	**77.5**	71.9	35.3*
Ours	H36m	74.8	**70.8**	77.3	**74.0**	**36.7**

Table 4. Results on the test set of MPI-INF-3DHP by scene. GS indicates green screen background. The results of [18] are directly taken from [17]. * uses extra ordinal annotation.

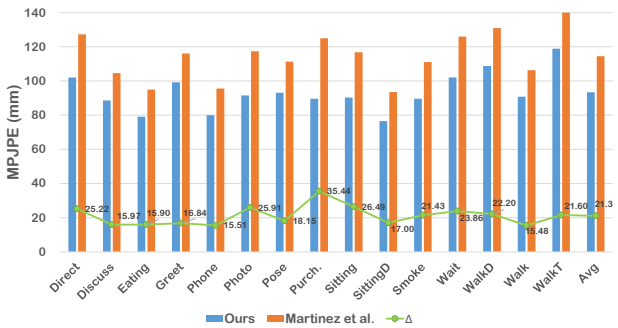

Figure 5. The 3D pose estimation errors when we train the FCN [18] and our LCN models on each of the 15 actions and test on all actions. X-axis indicates the action used in training.

This indicates our approach generalizes better to unseen actions.

Number of Training Data We investigate the impact of the number of training data. We conduct two experiments. In the first one, we randomly sample a predefined number of training data from all actions. We denote this experiment as "rich". In the second one, we only sample from the first action "Direction" which is denoted as "scarce".

Figure 6 shows the results. In the "scarce" experiment, we can see that our approach only needs about 2k examples

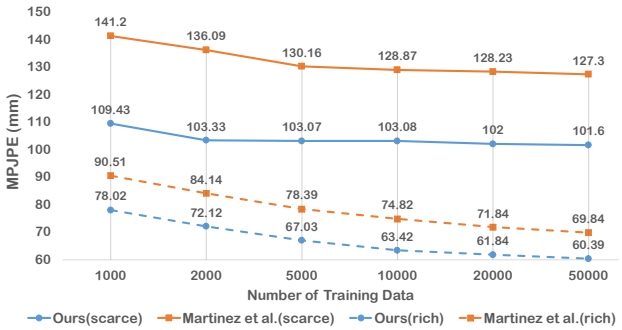

Figure 6. The 3D pose estimation errors when we train our models using different number of training data. X-axis indicates the number of data used in training. "rich" means the training data are from the 15 actions. "scarce" means the training data are only from the first "Direction" action.

to reach the optimal performance. In contrast, [18] needs more than 5K. This reflects our approach is less data hungry to achieve reasonable generalization ability.

Robustness to Noise We evaluate the robustness of our approach to inaccurate 2D joint locations. In this experiment, all approaches are trained using the ground truth 2D and 3D pose pairs. In testing, we sample random noises from the Gaussian distributions of different variances, and add them to the ground truth 2D poses. In particular, for each 2D pose, one joint will be corrupted. Then we estimate the 3D pose from the corrupted 2D pose. We compute the average 3D error increase for the rest of joints. Results are shown in Figure 7. We can see that when we manually specify the dependence between joints and assign them the same "importance", *i.e. LCN (2-NN)*, the error increase is smaller than [18], owing to the sparse connections. If we further learn the dependence partially *i.e. LCN (2-NN)-Learn* or completely *i.e. LCN -Learn*, the error increase is

Method	Dire.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD	Smoke	Wait	WalkD	Walk	WalkT	Avg
Bogo *et al.* [4]	62.0	60.2	67.8	76.5	92.1	77.0	73.0	75.3	100.3	137.3	83.4	77.3	86.8	79.7	87.7	82.3
Pavlakos *et al.* [23]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	51.9
Martinez *et al.* [18]	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
K.Lee *et al.* [15]	38.0	39.3	46.3	44.4	49.0	55.1	40.2	41.1	53.2	68.9	51.0	39.1	33.9	56.4	38.5	46.2
Fang *et al.* [7]	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.* [22]	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***
Ours	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
Martinez *et al.* (GT) [18]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	37.1
Ours (GT)	24.6	28.6	24.0	27.9	27.1	31.0	28.0	25.0	31.2	35.1	27.6	28.0	29.1	24.3	26.9	**27.9**

Table 5. 3D estimation errors of different methods on H36M under protocol #2. GT means the 2D poses are from the ground truth. * uses extra ordinal annotations.

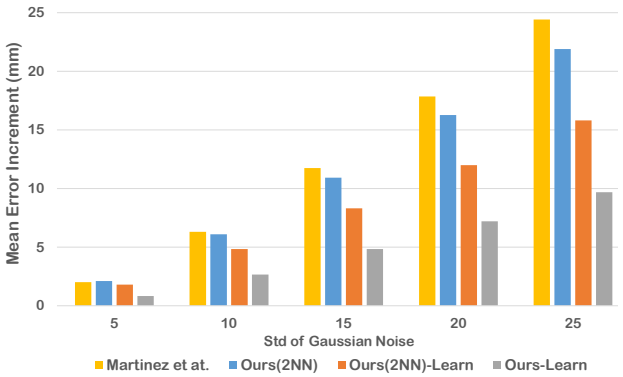

Figure 7. The average 3D error increment for the rest of the joints when one joint is corrupted by different levels of noises. We show the results for [18] and ours.

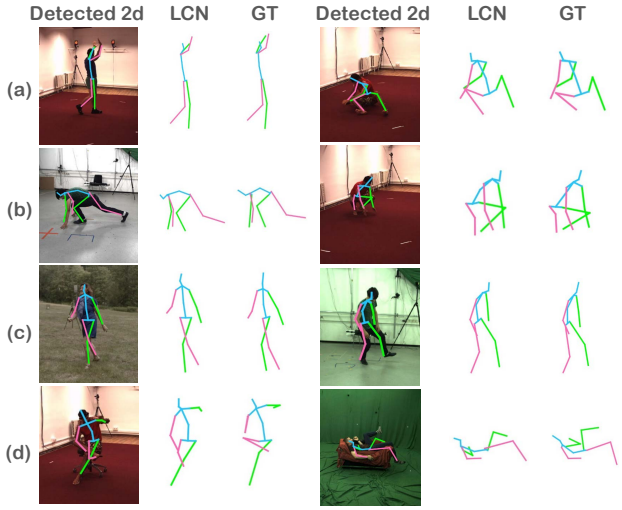

Figure 8. Sample 3D poses estimated by our LCN approach on the H36M and MPI-INF-3DHP datasets. Row (d) shows results for inputs with some joints wrongly detected.

further reduced by a large margin. This may be because the model has greater freedom to establish a connection and determine its strength, so that some auxiliary connections are weakened as shown in Figure 4.

5.7. Qualitative Results

We show several estimated 3D poses by LCN in Figure 8. The 2D poses are estimated by SH [20]. The 2D pose estimations are not perfect especially when occlusion happens. See the first example of Figure 8 (d) where the 2D locations of the right hand and foot are not correct. Our approach still generates reasonable 3D poses for the rest of the body parts. This suggests that the impact of the inaccurate 2D poses is constrained to be local.

6. Conclusion

We present LCN to estimate 3D human poses from 2D poses. It can be regarded as a generalization of GCN which overcomes its limitations. In particular, it has strong representation ability and generalization ability due to the appropriate joint dependence design. It outperforms the state-of-the-arts on two public datasets H36M and MPI-INF-3DHP. More importantly, it generalizes well to unseen actions, datasets and even noisy 2D poses.

7. Acknowledgement

This was supported in part by NSFC grants 61625201, 61527804 and Qualcomm University Research Grant.

References

[1] A Agarwal and B Triggs. 3d human pose from silhouettes by relevance vector regression. In *CVPR*, volume 2, pages II–II. IEEE, 2004.

[2] Ijaz Akhter and Michael J Black. Pose-conditioned joint angle limits for 3d human pose reconstruction. In *CVPR*, pages 1446–1455, 2015.

[3] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *NIPS*, pages 1993–2001, 2016.

[4] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl:

Automatic estimation of 3d human pose and shape from a single image. In *ECCV*, pages 561–578. Springer, 2016.

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016.

[7] Hao-Shu Fang, Yuanlu Xu, Wenguan Wang, Xiaobai Liu, and Song-Chun Zhu. Learning pose grammar to encode human body configuration for 3d pose estimation. In *AAAI*, 2018.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034, 2017.

[9] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[10] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[11] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *T-PAMI*, 36(7):1325–1339, 2014.

[12] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018.

[13] Isinsu Katircioglu, Bugra Tekin, Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. Learning latent representations of 3d human pose with deep neural networks. *IJCV*, pages 1–16, 2018.

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[15] Kyoungoh Lee, Inwoong Lee, and Sanghoon Lee. Propagating lstm: 3d pose estimation based on joint interdependency. In *ECCV*, pages 119–135, 2018.

[16] Mun Wai Lee and Isaac Cohen. Proposal maps driven mcmc for estimating human body pose in static images. In *CVPR*, volume 2, pages II–II. IEEE, 2004.

[17] Chenxu Luo, Xiao Chu, and Alan Yuille. Orinet: A fully convolutional network for 3d human pose estimation. *BMVC*, page 92, 2018.

[18] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *ICCV*, volume 1, page 5, 2017.

[19] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *3DV*, pages 506–516. IEEE, 2017.

[20] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, pages 483–499. Springer, 2016.

[21] Gregory Shakhnarovich Paul, Paul Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*. Citeseer, 2003.

[22] Georgios Pavlakos, Xiaowei Zhou, and Kostas Daniilidis. Ordinal depth supervision for 3d human pose estimation. In *CVPR*, pages 7307–7316, 2018.

[23] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *CVPR*, pages 1263–1272. IEEE, 2017.

[24] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *CVPR*, pages 459–468, 2018.

[25] Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Reconstructing 3d human pose from 2d image landmarks. In *ECCV*, pages 573–586. Springer, 2012.

[26] Helge Rhodin, Mathieu Salzmann, and Pascal Fua. Unsupervised geometry-aware representation for 3d human pose estimation. In *ECCV*, pages 750–767, 2018.

[27] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[28] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *NIPS*, pages 2244–2252, 2016.

[29] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *ECCV*, pages 529–545, 2018.

[30] Camillo J Taylor. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *CVIU*, 80(3):349–363, 2000.

[31] Bugra Tekin, Isinsu Katircioglu, Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. Structured prediction of 3d human pose with deep neural networks. *arXiv preprint arXiv:1605.05180*, 2016.

[32] Bugra Tekin, Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. Learning to fuse 2d and 3d image cues for monocular body pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3941–3950, 2017.

[33] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *CVPR*, pages 4627–4635. IEEE, 2017.

[34] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 1(2), 2017.

[35] Chunyu Wang, Yizhou Wang, Zhouchen Lin, and Alan L Yuille. Robust 3d human pose estimation from single images or video sequences. *IEEE transactions on pattern analysis and machine intelligence*, 41(5):1227–1241, 2018.

[36] Chunyu Wang, Yizhou Wang, Zhouchen Lin, Alan L Yuille, and Wen Gao. Robust estimation of 3d human poses from a single image. In *CVPR*, pages 2361–2368, 2014.

[37] Xiaolin K Wei and Jinxiang Chai. Modeling 3d human poses from uncalibrated monocular images. In *ICCV*, pages 1873–1880. IEEE, 2009.

[38] Wei Yang, Wanli Ouyang, Xiaolong Wang, Jimmy Ren, Hongsheng Li, and Xiaogang Wang. 3d human pose estimation in the wild by adversarial learning. In *CVPR*, volume 1, 2018.

[39] Wei Yang, Wanli Ouyang, Xiaolong Wang, Jimmy Ren, Hongsheng Li, and Xiaogang Wang. 3d human pose estimation in the wild by adversarial learning. In *CVPR*, June 2018.

[40] Xingyi Zhou, Qixing Huang, Xiao Sun, Xiangyang Xue, and Yichen Wei. Towards 3d human pose estimation in the wild: a weakly-supervised approach. In *ICCV*, 2017.