DenseGATs: A Graph-Attention-Based Network for Nonlinear Character Deformation

Tianxing Li
The University of Tokyo
Tokyo, Japan
litianxing@g.ecc.u-tokyo.ac.jp

Rui Shi
The University of Tokyo
Tokyo, Japan
shirui@graco.c.u-tokyo.ac.jp

Takashi Kanai
The University of Tokyo
Tokyo, Japan
kanait@acm.org

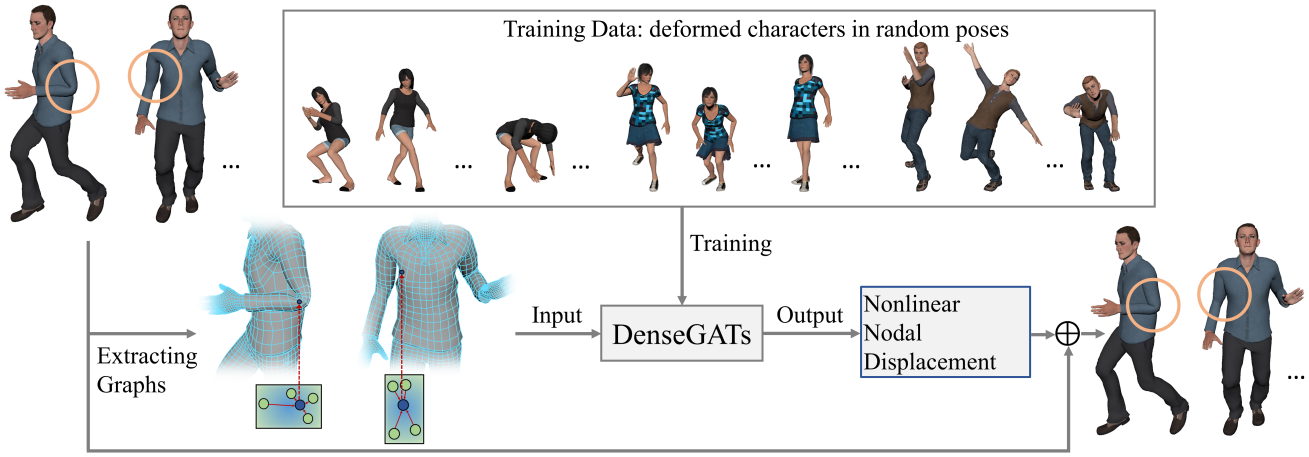

Figure 1: For multiple animated characters with different garments, our method uses a graph-attention-based network "DenseGATs" for correcting nodal linear deformations to nonlinear ones. By learning skinning features in the train set, our method yields more accurate deformations for new character meshes, thereby significantly reducing the time and efforts taken for performing the skinning process.

ABSTRACT

In animation production, animators always spend significant time and efforts to develop quality deformation systems for characters with complex appearances and details. In order to decrease the time spent repetitively skinning and fine-tuning work, we propose an end-to-end approach to automatically compute deformations for new characters based on existing graph information of high-quality skinned character meshes. We adopt the idea of regarding mesh deformations as a combination of linear and nonlinear parts and propose a novel architecture for approximating complex nonlinear deformations. Linear deformations on the other hand are simple and therefore can be directly computed, although not precisely. To enable our network handle complicated graph data and inductively predict nonlinear deformations, we design the graph-attention-based (GAT) block to consist of an aggregation stream and a self-reinforced stream in order to aggregate the features of the neighboring nodes and strengthen the features of a single graph node. To reduce the difficulty of learning huge amount of mesh features, we introduce a dense connection pattern between a set of GAT blocks called "dense module" to ensure the propagation of features in our deep frameworks. These strategies allow the sharing of deformation features of existing well-skinned character models with new ones, which we call densely connected graph attention network (DenseGATs). We tested our DenseGATs and compared it with classical deformation methods and other graph-learning-based strategies. Experiments confirm that our network can predict highly plausible deformations for unseen characters.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Animation**.

KEYWORDS

character rig, mesh deformation, graph learning

ACM Reference Format:
Tianxing Li, Rui Shi, and Takashi Kanai. 2020. DenseGATs: A Graph-Attention-Based Network for Nonlinear Character Deformation. In *Symposium on Interactive 3D Graphics and Games (I3D '20), May 5–7, 2020, San Francisco, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3384382.3384525

I3D '20, May 5–7, 2020, San Francisco, CA, USA

ACM ISBN 978-1-4503-7589-4/20/05…$15.00
https://doi.org/10.1145/3384382.3384525

1 INTRODUCTION

During the process of animating a character, rigging is a crucial task which involves defining motion, control, and deformation systems. The process of developing a deformation system is always labor intensive and this has been receiving considerable attention in recent years. Traditional skinning methods such as Linear Blend Skinning (LBS) and Dual Quaternion Skinning (DQS) are widely adopted in real-time applications due to their simplicity and efficiency. However, both methods produce unrealistic deformation artefacts. To create more realistic skin deformations that make the appearance look more detailed and complex, skinning processes including interative weight painting and additional modification are unavoidably needed, which require animators to spend significant efforts and time on the manipulation of character models. Few studies have attempted to address this problem by automatically generating skinning weight [6, 18]. However, the drawback of these approaches is that they are not able to generate highly plausible nonlinear mesh behavious which need further manual interventions. Lately, Bailey et al. [3] used a simple feed-forward network to approximate complex nonlinear deformations to achieve film-quality accuracy while enabling real-time animation. However, once the networks were trained, they could only work for one specific character model and the application to new character models was limited.

Based on the above, two main challenges should be addressed. The first is to explore an integrated deformation system, which directly results in complicated deformations that have more convincing effects. The second is make this deformation system generalized that allows to apply available artist-designed skinning mesh features to new character meshes to avoid time-consuming manual painting and fine-tuning work.

In this paper, we present an approach to leverage a novel densely connected graph attention network, named "DenseGATs" to achieve the goal of automatically generating satisfactory deformations for new characters. Following the deformation decomposition with linear and nonlinear parts, our proposed graph-attention-based network can be applied to clarify the intricate relations of the mesh graph-displacement of the nonlinear part. The outline of the method is shown in Fig. 1. When input with a linear-based deformation graph which encodes the mesh and skeleton attributes of a vertex, our network (detailed structure is shown in Fig. 2) end-to-end maps it to a corrective displacement per mesh vertex for more complicated nonlinear effects.

Specifically, our technical contributions are two-fold:

- To deal with arbitrarily structured mesh graph data, our method leverages graph-attention-based (GAT) blocks for effectively learning complicated mesh features. Specifically, in each GAT block (details in Fig. 3), we extend the original GAT structure by adding a self-reinforced stream that linearly maps the individual features of each node. This stream is then concatenated with the graph-attention-based aggregation stream to form a GAT block. In this way, the GAT block can effectively compile information on complex graph features from neighboring vertices as well as information on self-features.
- To extract high-level features with deep layers and ensure the propagation of information on large amount of features, we introduce "dense module" that adopts dense connectivity patterns between several GAT blocks to resolve the vanishing gradient problem and effectively improve information flow by fusing multiple levels of features.

For training purposes, we built a dataset which involves a set of differently dressed characters with high-quality rigs. Furthermore, we also animated these characters with possible poses. After training, the trained network is then used to predict nonlinear deformation which is the vertex offset for new character samples in the test set.

2 RELATED WORK

Skinning method. Common approaches for deforming the skin of articulated characters can roughly be classified into geometry-based, example-based, and physics-based methods. Geometry-based methods deform the skin mesh by the weighted transformation of bones applied to each vertex. The basic idea of well-known algorithm linear blend skinning (LBS) [21] widely adopted in real-time applications is to linearly blend transformation metrics. LBS is simple and highly efficient, but always suffers from artefacts such as "candy-wrapper" and volume loss. Kavan et al. [11] proposed a skinning method named dual quaternion skinning (DQS) by replacing linear blending with nonlinear blending based on dual quaternions. It eliminates the artefact effects of LBS but produces undesirable joint-bulging artefacts.

Example-based methods permit more complex skinning effects such as skin slide, muscle bulges, and wrinkling of clothing. They treat deformations as a shape interpolation problem which takes, as input, a series of scattered data poses to obtain the desired deformation. Pose space deformation (PSD) [14] uses a radial basis function for scattered interpolation. Based on this, weighted pose space deformation [12] was then proposed with limited number of poses. More recently, Le et al.[13] presented a method for generating linear blend skinning models by using a set of example poses. The obtained model includes skeletal structure, skinning weights, joint positions, and corresponding bone transformations. Studies in [5, 19] used the skinned vertex-based models to fix skinning artefacts and enrich the nonlinear effects. However, their models are unable to be applied to a new mesh with different number of vertices from the training set.

Realistic character deformations can also be realized through physics-based methods [8]. McAdams et al. [22] proposed a robust method to simulate a deformed surface based on a hexahedral lattice. Xu et al. [27] presented a method to add physically-based dynamics to pose-space deformation and character rigging. Recently, Pan et al. [24] improved the traditional position-based dynamics by adding energy constraints to realize automatic skinning and weight retargeting with new characters.

Neural-Network-based deformation. Past researches have explored methods to use neural networks for computer animation. To clarify the force-displacement relation of nonlinear materials, Luo et al. [20] used a simple neural network to map linear elasticity deformations to nonlinear ones. Similar to the idea of decomposing a deformation with linear and nonlinear parts, Bailey et al. [3] used feed forward neural networks to approximate nonlinear

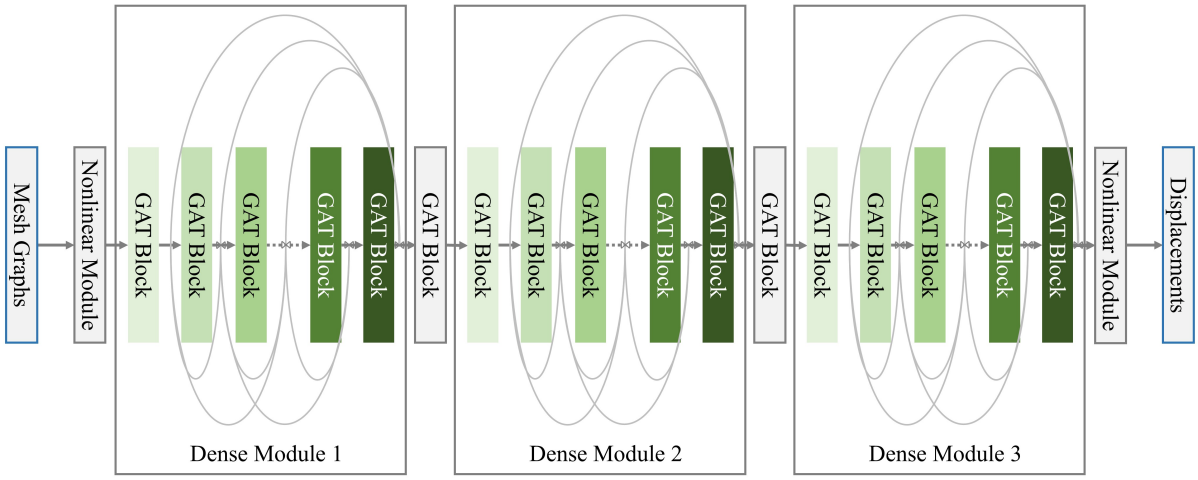

Figure 2: Structure of "DenseGATs". The three boxes denote our proposed "dense modules". Each dense module comprises six densely connected GAT blocks. Fig. 3 shows the details of each GAT block.

deformations based on linear blend skinning. They feed the transformation matrix and translation vector to the network and output the corrected displacement at interactive rates. One problem of this method is that the trained networks cannot be fitted to the new characters because the networks are optimized for specific character bones and meshes. Recently, Liu et al. [18] presented a method to automatically predict skin weights especially for game characters with complicated dressing. By using a graph convolution network, new characters with predicted skin weight maps can yield satisfactory deformations, but the algorithm only computes fixed weights and only assumes deformations are a function of the skeleton.

Graph neural networks. Graph neural networks (GNNs) are deep-learning-based methods capable of reasoning about 3D data. Zhou et al. [28] categorized GNNs into several groups (convolution, gate mechanism, skip connection and attention mechanism) based on their propagation step. Existing works with convolution operations on graph can be divided into spectral approaches [4, 9, 10, 17] and spatial approaches [1, 2, 7, 23]. The former methods define the convolution operation in the Fourier domain by computing the eigen decomposition of graph Laplacian, while the latter directly defines convolution operations on spatially close neighboring nodes. Focusing on attention mechanism, the graph attention network proposed by [26] applies an attention mechanism to the propagation step. It is able to specify different weights to different nodes while aggregating different sized neighboring nodes, and does not depend on knowing the graph structure upfront. Moreover, the established models can be successfully applied to inductive learning.

3 DENSEGATS

Deformation rigging is a quite difficult process, due to the need to bind the skeleton to the character mesh with the skinning algorithms and other refinements to determine how the skin meshes move. Our method provides a novel, direct way to compute deformation that expresses the deformation of the mesh as the sum of two functions: a linear-based deformation function $t(\boldsymbol{S})$ which linearly maps the skeleton transformation $\boldsymbol{S}$ to the mesh vertex position, as well as a graph-attention-based nonlinear deformation function $h(\boldsymbol{\mathcal{G}})$ which further refines linear deformation $t(\boldsymbol{S})$ based on its mesh graph features $\boldsymbol{\mathcal{G}}$.

$t(\boldsymbol{S})$ is computed with LBS which is simple and direct, and characters are deformed by transforming vertices through a weighted combination of bone transformations. For more attractive effects, our method provides an additional step that uses a graph-attention-based network to generate nonlinear corrected displacements of nodes based on linear deformation. We treat the nonlinear function $h(\boldsymbol{\mathcal{G}})$ as a black box which allows for more general deformations based on mesh features and we approximate it with our expressive DenseGATs network. This section describes our proposed method on how to compute $h(\boldsymbol{\mathcal{G}})$ in detail. We start with a description of how to construct our input graph. We then present our framework including GAT block, dense module and the overall network successively. Implementation detail will also be explained at last.

3.1 Graph Construction and Features Mapping

We expect the input of our network to be an undirected graph of linear skinning mesh $\boldsymbol{\mathcal{G}} = (\mathcal{V}, \mathcal{E}, \mathbf{U})$ with $\mathcal{V} = \{1, ..., N\}$ being the set of nodes, which also indicates the vertices of mesh, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges between nodes, and $\mathbf{U} \in [0, 1]^{N \times N}$ is the adjacency matrix where $\mathbf{u}(i, j) \in [0, 1]$ indicating whether there is an edge between nodes i and j, $(i, j) \in \mathcal{E}$. For a node $v_i \in \mathcal{V}$, its neighboring nodes set is represented by $\mathcal{N}(i)$.

For a node $v_i \in \mathcal{V}$, we apply the same definition as in [18]. The attribute vector is defined as $v_i = [p_i^T, n_i^T, s_i^T]$, where $p_i \in \mathbb{R}^3$ is the vertex position, $n_i \in \mathbb{R}^3$ is the normal of the vertex, and $s_i \in \mathbb{R}^J$ is the distance vector to all J skeleton joints. Thus, v_i contains both mesh skinning appearance attributes which indicate the features of the vertex itself, connectivity between other vertices, and the distance attributes from the vertex to joints which implies a positional relationship between the vertex and control skeleton.

Next, for the input of the whole network, features of N nodes are described as $\boldsymbol{g}^{[0]} = \{\vec{g}_1, ..., \vec{g}_i, ..., \vec{g}_N\} \in \mathbb{R}^{N \times d^{[0]}}$, to represent the features of each vertex in graph $\boldsymbol{\mathcal{G}}$ whose spatial relations are locally defined by the pseudo-coordinates in $\mathbf{U}$. $d^{[0]}$ is the input dimension.

The features of every node i are first transformed using a nonlinear feature transformation module. The module consists of linear layers followed by nonlinear activation and normalization. Suppose that $\boldsymbol{L} \in \mathbb{R}^{d^{[0]} \times d^{[1]}}$ is a linear weight matrix of the first linear layer where $d^{[1]}$ is the dimension of the output features after this layer. The feature dimension is transformed from $d^{[0]}$ to $d^{[1]}$ through this layer where the trained weight $\boldsymbol{L}$ is independent of the varying numbers of vertices. By this features mapping process, the input features are transformed into normalized high-dimensional features that could provide rotation and translation invariance in some degree with enough training data.

3.2 GAT Block

Graph neural networks are capable of dealing with non-Euclidean data like mesh. For our character deformation prediction, one challenge is to use prior mesh graph information to inductively generalize the graph features of mesh that have never been seen before. [26] introduces an graph-attention-based architecture to compute the hidden representation of each node by aggregating neighborhood features with different weights, without the need to know the entire graph structure upfront. Inspired by this, we proposed a GAT block architecture shown in Fig. 3. The GAT block consists of a graph-attention-based aggregation stream and a self-reinforced stream. The aggregation stream is used to compute the hidden representations of each node in graphs, by applying its adjacent features using a graph attention network. In addition to the neighboring nodes aggregated by graph convolution, strengthening single node features is also necessary to accurately convey information to deeper layers. Therefore, we designed a self-reinforced stream to transform the node's own features and concatenate them with the output of the graph convolution. These features, along with the aggregation steam obtained features, are then fed into next GAT block.

After nonlinear transformation, the node features are input into GAT blocks. Here, we use $\boldsymbol{g}^{[l]} = \left\{\vec{g}_1^{[l]}, ..., \vec{g}_i^{[l]}, ..., \vec{g}_N^{[l]}\right\}, \vec{g}_i^{[l]} \in \mathbb{R}^{d^{[l]}}$ and $\boldsymbol{g}^{[l+1]} = \left\{\vec{g}_1^{[l+1]}, ..., \vec{g}_i^{[l+1]}, ..., \vec{g}_N^{[l+1]}\right\}, \vec{g}_i^{[l+1]} \in \mathbb{R}^{d^{[l+1]}}$ to represent a input and output of one GAT block, where $d^{[l]}$ and $d^{[l+1]}$ indicate the feature dimensions after previous l and $l+1$ layers' feature transformation. The overall process of feature transformation inside one GAT block can be expressed as $\vec{g}_i^{[l+1]} = f_{\mathrm{GATB}}(\vec{g}_i^{[l]})$.

Specifically, for the graph-attention-based aggregation streams, given a set of node features, they firstly need to be pre-processed so that they can be applied to each node by linear transformation to obtain higher dimensional expressions. The transformed features can be expressed as:

$$\vec{z}_i^{[l]} = \boldsymbol{W}\vec{g}_i^{[l]} \tag{1}$$

where the trainable parameter of the transformation $\boldsymbol{W}$ is a weight matrix $\boldsymbol{W} \in \mathbb{R}^{F \times d^{[l]}}$. F denotes the transformed feature dimension of a node in the aggregation stream.

Although aggregation streams could integrate features from first-order neighboring nodes, the features of the node itself are diminished in this aggregation. We designed a self-reinforced stream which is a linear transformation process for $\vec{g}_i^{[l]}$:

$$\vec{r}_i^{[l]} = \boldsymbol{H}\vec{g}_i^{[l]} \tag{2}$$

where $\boldsymbol{H}$ is a trainable weight matrix $\boldsymbol{H} \in \mathbb{R}^{\beta F \times d^{[l]}}$. Here, β is the hyper-parameter determining the importance of the self-reinforce stream.

For aggregation steam, as stated in [26], a shared, masked attention mechanism is performed that only computes attention coefficients with the neighboring nodes $\mathcal{N}(i)$ of a node which ensures that structural information is not lost. Furthermore, to make the results of attention coefficients across different neighborhoods comparable, they are normalized using the softmax function to obtain attention weights:

$$\alpha_{ij}^{[l]} = \frac{\exp(\mathrm{LeakyReLU}(\vec{a}^{[l]^T}(\vec{z}_i^{[l]} \| \vec{z}_j^{[l]})))}{\sum_{k \in \mathcal{N}(i)} \exp(\mathrm{LeakyReLU}(\vec{a}^{[l]^T}(\vec{z}_i^{[l]} \| \vec{z}_k^{[l]})))} \tag{3}$$

where $\alpha_{ij}^{[l]} \in \mathbb{R}$. $\vec{a}^{[l]} \in \mathbb{R}^{2F}$ indicates the weight vector and $(\cdot)^T$ represents its transposition. $\|$ is the concatenation operation that features $\vec{z}_i{}^{[l]}$ and features from neighboring nodes are firstly concatenated, and then this concatenation embedding result with a learnable weight vector $\vec{a}^{[l]^T}$ are executed by dot product. During the process of calculating attention coefficient, to enhance nonlinear expression, LeakyReLU is applied as the activation function.

The obtained attention weights are then linearly combined with their corresponding features to yield the output features of the aggregation steam. To improve the stability and representation ability of the model, multi-head attention mechanism K introduced in [25] is also adopted to execute transformation K times with different training parameters of aggregation operations. In addition to the aggregation steam, to enhance the expression of self-features, our designed self-reinforced steam is concatenated with the features from the graph-attention-based aggregation steam (as shown in Fig. 3) to form the final output features of one GAT block:

$$\begin{aligned}\vec{g}_i^{[l+1]} &= f_{\mathrm{GATB}}(\vec{g}_i^{[l]}) \\ &= \sigma(\|_{k=1}^{K}(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{[l]k}\boldsymbol{W}^k\vec{g}_j^{[l]}) \| \vec{r}_i^{[l]})\end{aligned} \tag{4}$$

where σ represents the nonlinear transformation tanhshrink. Since the output type is displacement which could be positive and negative, tanhshrink is able to retain negative information. Meanwhile, it can alleviate gradient vanishing problem (tanh cannot) and therefore is selected in our network. $\alpha_{ij}^{[l]k}$ and $\boldsymbol{W}^k$ respectively indicate the attention weights and input linear transformation's weight matrix computed by the k_{th} attention mechanism in the l_{th} layer. For one GAT block, the dimension of output features $d^{[l+1]}$ equals the concatenation dimension of aggregation stream and self-reinforced stream $KF + \beta KF$.

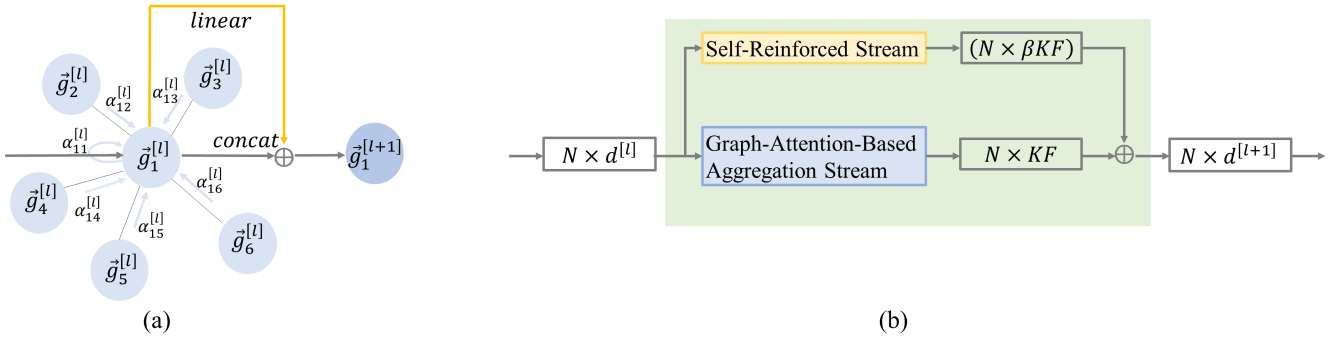

Figure 3: Illustration of inside of GAT block by the node 1 (feature vector is $\vec{g}_1^{[l]}$) on its neighboring nodes. The blue color indicates the process of graph-attention-based aggregation stream gathers features from neighboring nodes with different weights [26]. The yellow color indicates the process of self-reinforced stream that linearly transforms the features of the node 1. These two streams are then concatenated to produce new feature representation. In (a), we show the situation of multi-head attention $K = 1$. In (b), the input feature dimension into one GAT block is $d^{[l]}$, the output dimension of aggregation stream is F, the number of multi-head attention is K, and the hyper-parameters β. So the output feature dimension is $d^{[l+1]} = KF + \beta KF$.

3.3 Dense Module

Due to the complexity and significant amount of features in our mesh graph, there is a need to apply very deep networks to benefit from their advantages. However, for training deep graph convolutional networks, the problem of gradient vanishing will become more serious as the number of network layer increases. To address this problem, DenseGCNs[16] borrows the concept from DenseNet [15] by introducing dense connections to deep GCN frameworks and successfully be applied on segmentation task with specific point cloud graph structure. Inspired by it, we densely connect our GAT blocks as a dense module for further training deep layers of GAT blocks to better help process large volumes of complicated mesh data. As opposed to DeepGCNs, our dense module allows for handling unseen graph structures and efficiently computing with difference importance to nodes of a same neighborhood. For simplicity, we use a single function f_{GATB} to represent the transformation process of one GAT block in the rest of this paper. In a dense module, among several GAT blocks, direct connections are introduced from one block to all subsequent blocks. Suppose, there are m GAT blocks in one dense module, the layer of the first GAT block is l, thus, the $(l+m)_{th}$ layer receives features of all layers starting from the l_{th} layer. The propagation can be defined as:

$$\begin{aligned}\vec{g}_i^{[l+m]} &= f_{\text{GATB}}(\vec{g}_i^{[l+m-1]}, \theta^{[l+m-1]}) \| \vec{g}_i^{[l+m-1]} \\ &= f_{\text{GATB}}(\vec{g}_i^{[l+m-1]}, \theta^{[l+m-1]}) \| \ldots \| f_{\text{GATB}}(\vec{g}_i^{[l]}, \theta^{[l]}) \| \vec{g}_i^{[l]}\end{aligned} \tag{5}$$

where $\theta^{[\cdot]}$ represents all parameters of the transformation function f_{GATB} in different layers. The $\vec{g}_i^{[l+m]}$ is the result of fusing all the intermediate GAT block layer outputs. Since we connect GAT blocks in a dense pattern, we refer to this architecture as "dense module", and the whole network refers to "DenseGATs". Note that, because of the dense connection, for each GAT block (except the first block) in a dense module, the output features will consist of all preceding blocks' features, not only one block feature ($KF + \beta KF$) for each node.

Between two adjacent dense modules, we adopt one GAT block as the transition. This operation plays the role of information integration.

3.4 Implementation Details

Based on the amount of training data available, we explored a set of parameters which can ensure the satisfied result while balancing the size of network.

As shown in Fig. 2, we first feed the input graph features into a nonlinear transformation module which involves two fully connected hidden layers with 32 hidden units and followed by tanh activation. Then the transformed features are fed into dense modules. The network we propose involves three dense modules, each of which consists of six GAT blocks. We found this setting is sufficient to approximate satisfactory results without excessively increasing training time. To ensure maximum information flow between GAT blocks in one module, all blocks are densely connected. For the internal structure of the GAT block, we adopt a graph-attention-based aggregation stream with the hidden features number of 8 and the multi-head number of 8. To increase the effectiveness of the interpenetration, in the self-reinforced stream, input features are processed by a linear layer with the feature size of 0.125 times the aggregation stream output feature size ($\beta = 0.125$). The final layer of one GAT block is applied a tanhshrink activation function. We refer to GAT blocks between each dense module as transition blocks which have the same parameters with GAT blocks inside the dense module. After all three dense modules, the resulting features are taken as being input to two fully connected hidden layers with 512 and 128 hidden units and followed by tanh activation. All the outputs of layers in the network are applied with 1D batch normalization.

We trained our model with NVIDIA GeForce RTX2080Ti GPU and set batch size of 8. During the training process, we trained the model using the Adam optimization method, with the initial

learning rate of 1e-3. We further set the reducing learning rate with decay factor of 0.75 when the loss has stopped decreasing beyond eight epochs. The lower boundary on the learning rate of all param groups is set as 5e-7. For the loss function of our network, we choose Mean Squared Error (MSE) to minimize the distance between predicted displacement distributions and the ground truth displacement distribution.

It is noteworthy that, the training parameters in our network is independent of mesh vertices, but is correlated with the feature dimension of input graph. Thus, our network architecture is applicable to inductive learning where the trained network could be generalized to approximate deformation for new character mesh, regardless of the number of vertices.

4 EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we evaluate the performance of our framework both qualitatively and quantitatively. There are about 150 character models with different customizations in our dataset created with Adobe Fuse CC, and all of them are embedded with corresponding skeleton structures properly. To produce basic skinning data for training, the linear blend skinning method is used with eight maximum of joints influencing each vertex for rough deformation. In addition, to produce ground truth data, we manually refined the deformed skin that allows the character models to achieve satisfactory deformations under any of the poses. Fig. 4 shows several test example characters with rest pose in our dataset. These example characters with the statistics are shown in Tab. 1 including the number of vertices, heights, and number of joints. With all characters in our dataset, in order to save GPU memories and training time, several character bodies without head part are taken into account. All characters are set with a standard skeleton structure, for a total of 65 joints for the entire skeleton. To accurately predict the mesh deformation generated under any poses, we used two strategies to create the training examples. Firstly, in order to cover the range of all possible poses, we manually set each skeletal joint in a reasonable range for the rotation and scaling, and then generate poses by independently and randomly sampling in the range of all joints. Furthermore, we animate our character models with motions that appear frequently in animation such as walking, jumping, running and dancing provided by TurboSquid. In total, there are about 8500 poses generated with these methods for our characters, and several examples are shown in Fig. 5. Due to the large amount of computation during the training, we randomly select one thousand consecutive vertices from each character mesh at each training epoch to ensure stable training while saving memories. To verify the effectiveness of our proposed network, we used 125 training character models and five validation character models of the dataset with about 8500 poses. The remaining 20 character models with random poses are used for testing the network.

4.1 Model Accuracy

To quantitatively evaluate the performance of our network, we measured the average distance error, max distance error, and minimum distance error of our predicted deformations. For each character model in the test set, we animated them using walking motions

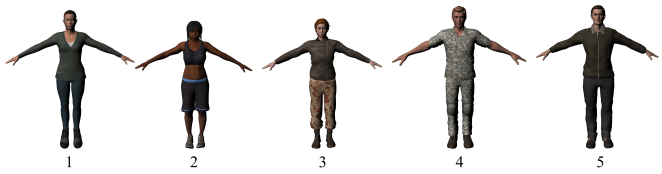

Figure 4: Example characters with index 1-5 in our dataset. These characters have completely different representative dresses (tops, bottoms, shoes) and the same skeleton structure.

Table 1: Statistics for our example characters (as shown in Fig. 4).

Character Index	Vertices	Heights	Joints
1	9035	177 cm	65
2	10220	166 cm	65
3	9303	167 cm	65
4	9005	181 cm	65
5	10320	182 cm	65

Figure 5: Example poses of our character. These includes typical poses (walking, running, jumping, and dancing) and random poses.

and computed the prediction vertex errors across all frames. Tab. 2 shows the prediction errors for the test models. It can be observed that our proposed method can effectively predict the deformation for new characters with very low errors. With our network, the prediction time for one character (e.g., No.2 character) in each frame is about 31ms. In Fig. 6, we further plot the average error of the No.2 test character to visualize error distribution over the mesh for all frames of dancing and running animations. From these two plots, most of deviation distances are are around 0.1cm that within the allowable range of accuracy. And the number of vertices decreases exponentially with increasing distance errors. We found that the performance of predicting deformation with running animation is better than with dancing animation, where there exists several vertices in the dancing motion that have the estimated errors of more than 0.5cm. This is because the dancing animation includes some extreme poses which are beyond the range of network it was trained on.

To verify that our trained network is able to predict satisfactory and natural deformation results, we animate the test models with

Table 2: Prediction errors in "cm" between the ground truth and predicted vertices.

Character Index	mean error	max error	min error
1	0.0662	1.1639	0.0003
2	0.1045	1.5832	0.0006
3	0.0947	1.4909	0.0003
4	0.0820	1.4201	0.0003
5	0.1289	1.6966	0.0005

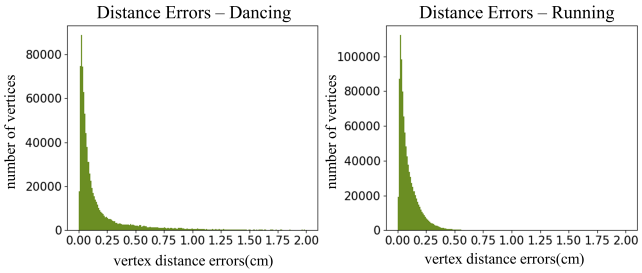

Figure 6: Distribution of vertex errors. The left is for dancing animation, and the right is for running animation.

weightlifter motion as shown in Fig. 7. We focus on some contact regions where mesh deformation volume always cannot be maintained and is highly prone to artefacts. The results approximated by our method are hardly distinguishable from the ground truth, which demonstrates that our network is able to correct the rough linear deformation to the more complex nonlinear one.

We provide further intuitive deformation results with multiple poses in different animations using error color map to qualitatively evaluate our method in Fig. 8. In Fig. 9, we also provide a side-by-side comparison of the ground truth, the approximated deformation, and the error color map of a test character with a walking posture. With our proposed network, the approximated deformation results are visually similar to the ground truth. It can be noted that the largest deformation errors are mainly in the trousers (especially the top of trousers) of the character. The deformations in these regions are always influenced by multiple bones. In addition, the trousers worn by the test character are quiet loose, whose mesh features are significantly different from those in our training set. And these reasons result in some inaccurate approximation.

4.2 Comparisons

We compare our method with classical deformation methods LBS and DQS. As shown in Fig. 10, the deformation with LBS method shows the obvious artefact of volume loss when twisting the elbow. Unnatural deformations are always found in the areas near the shoulder, elbow, waist and the bottom of the pants. It is hard to assign reliable weights with direct LBS method especially for those areas influenced by multiple bones. For DQS result, bulging artifacts near the joints are very obvious which still need artistic corrections. Our method, in contrast, given the inaccurate LBS deformation, can

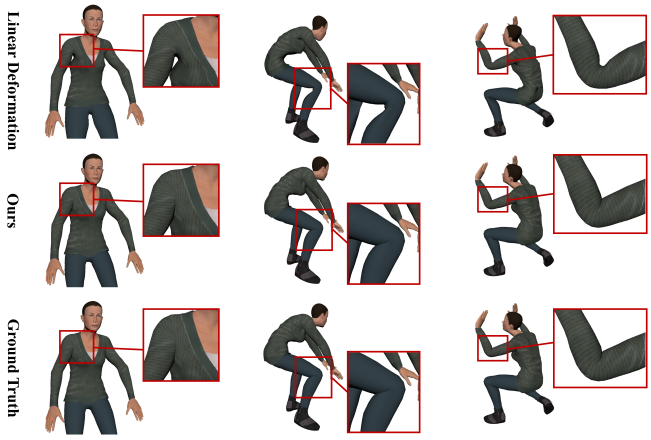

Figure 7: Close-up of our test character's contact regions in three frames of weightlifter motions. Based on rough linear deformation, our method corrects linear deformations to nonlinear ones by per vertex displacement correction.

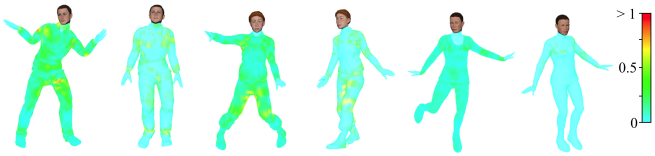

Figure 8: Result of deformation approximation among different characters with different poses. The vertices of the predicted mesh are colored to indicate per-vertex distance prediction errors.

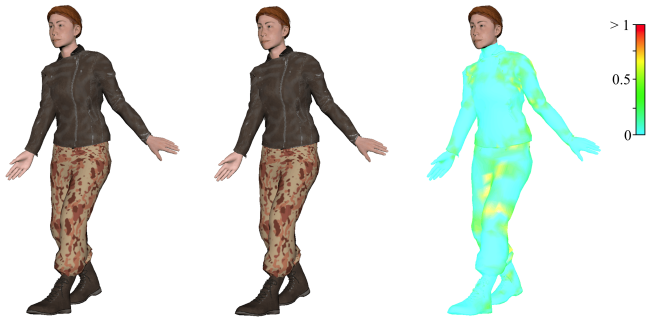

Figure 9: Comparison of ground truth (left), prediction results (center), and color map (right) indicating per-vertex distance error.

accurately approximate most displacements of vertices and correct them to nonlinear ones with no noticeable errors.

To verify the effectiveness of our proposed network, we also conducted experiments on character models in the test set with hip-hop dancing motion to compare the performance with different network structures like those shown in Tab. 3. We first compare

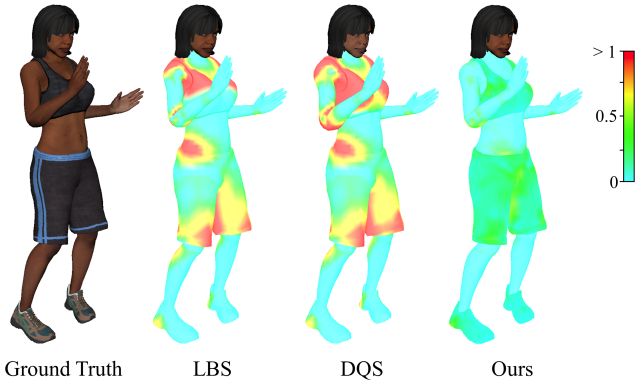

Figure 10: Comparison of ground truth, color map of LBS, DQS, and our prediction.

Table 3: Evaluation of predicted distance errors (cm) using different network architectures.

Network	mean error	max error	min error
DenseGATs (Ours)	0.0961	1.2551	0.0004
Ori.GAT	0.1693	2.1643	0.0009
NeuroSkinning network	0.1378	1.7689	0.0006
Ours w/o self-reinforcement	0.1153	1.3986	0.0002
Ours w/o dense connection	0.1425	1.4852	0.0006

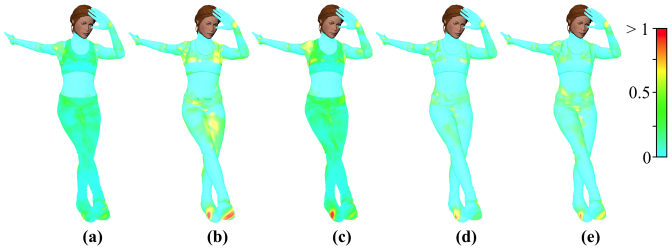

Figure 11: Comparison of different network architectures. From left to right: (a) ours, (b) original GAT, (c) NeuroSkinning, (d) ours (no self-reinforcement stream), (e) ours (no dense connection).

with the original GAT [26] network which contains four graph convolutional layers. Next, we evaluated the network described in NeuroSkinning [18] and followed the same experimental settings. Based on our proposed structure, we respectively removed the designed self-reinforced stream in each GAT block and dense connections between each GAT block. The prediction results are summarized in Tab. 3 and an example frame of the animation are shown in Fig. 11. It could be observed that the original GAT network sometimes fails to represent complicated graph information and produces highest errors due to the limited representation of the shallow convolutional layers. The NeuroSkinning network has the same problem as the original GAT because there are only three graph convolutional layers in the whole structure. For DenseGATs structure without self-reinforced stream and the structure of multiple GAT blocks without dense connection, features can be well learned and the obvious approximation errors can be improved to a certain extent, but they still cause some undesirable deformation in several joint regions. In contrast, with our proposed network , the best prediction result can be obtained with improvement rates of about 16.7% and 32.6% compared with the DenseGATs without self-reinforced stream and without dense connection. This suggests that out network has better generalization ability owing to its self-reinforcement step and dense propagation step.

5 CONCLUSION AND FUTURE WORK

We have presented a DenseGATs network that leverages existing well-designed skinning features of characters to accurately predict deformation for new characters. Our GAT blocks and dense modules allow for efficient utilization and transmission of self and adjacent information throughout the network. Experimental results demonstrate that through these strategies, the skinning features from other characters can be reused and the network is able to generate realistic nonlinear deformation results that are very close to ground truth. The high-quality deformation predicted with our method can be directly used for masses of similar characters in films, thus reducing the efforts made by artists to re-rig for new characters each time.

There are also a number of limitations and potential improvements that could be addressed as future work. First, our method requires many training samples (both characters and poses), and its generalization ability is substantially dependent on the character mesh information that has already been learned. Our network may fail to accurately predict deformations if the mesh geometry and poses vary dramatically. In the future, we would like to expand our training set as well as design more efficient feature representation methods (e.g., the graph topology with rotational-invariance). Meanwhile, with the limited number of characters in dataset, next we aim to study related graph learning methods such as applying down-sampling to graphs to further improve the generalization ability of our network. Secondly, we add displacement to each vertex to make the deformation nonlinear. However, this operation will sometimes unexpectedly displace individual vertices and make the deformation become not smooth. In the future, we hope to address this limitation by investigating a new regression method guaranteed to smooth deformations while realizing rich details accurately. Thirdly, currently, we train all characters together and do not consider the material attributes of different garments and skin. To perform perfect visual effects of deformations, future work can independently train networks with different materials. Additionally, our dataset involves human characters with the same skeleton structure. In the future, we hope to use our network to generalize other models with different number of joints.

ACKNOWLEDGMENTS

Tianxing Li and Rui Shi acknowledge receipt of Japanese government (MEXT) scholarships. This work was partially supported by JSPS KAKENHI, Grant Number JP19K11990.

REFERENCES

[1] Nesreen K. Ahmed, Ryan A. Rossi, Rong Zhou, John Boaz Lee, Xiangnan Kong, Theodore L. Willke, and Hoda Eldardiry. 2017. Inductive Representation Learning in Large Attributed Graphs. arXiv:1710.09471

[2] James Atwood and Don Towsley. 2016. Diffusion-convolutional Neural Networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*. Curran Associates Inc., USA, 2001–2009. http://dl.acm.org/citation.cfm?id=3157096.3157320

[3] Stephen W. Bailey, Dave Otte, Paul Dilorenzo, and James F. O'Brien. 2018. Fast and Deep Deformation Approximations. *ACM Trans. Graph.* 37, 4, Article 119 (July 2018), 12 pages. https://doi.org/10.1145/3197517.3201300

[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral Networks and Locally Connected Networks on Graphs. arXiv:1312.6203

[5] Dan Casas and Miguel A. Otaduy. 2018. Learning Nonlinear Soft-Tissue Dynamics for Interactive Avatars. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 1, Article Article 10 (July 2018), 15 pages. https://doi.org/10.1145/3203187

[6] Olivier Dionne and Martin de Lasa. 2013. Geodesic Voxel Binding for Production Character Meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '13)*. ACM Press, New York, NY, USA, 173–180. https://doi.org/10.1145/2485895.2485919

[7] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-Scale Learnable Graph Convolutional Networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM Press, New York, NY, USA, 1416–1424. https://doi.org/10.1145/3219819.3219947

[8] Fabian Hahn, Sebastian Martin, Bernhard Thomaszewski, Robert Sumner, Stelian Coros, and Markus Gross. 2012. Rig-space Physics. *ACM Trans. Graph.* 31, 4, Article 72 (July 2012), 8 pages. https://doi.org/10.1145/2185520.2185568

[9] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (March 2011), 129–150. https://doi.org/10.1016/j.acha.2010.04.005

[10] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep Convolutional Networks on Graph-Structured Data. arXiv:1506.05163

[11] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with Dual Quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (I3D '07)*. ACM Press, New York, NY, USA, 39–46. https://doi.org/10.1145/1230100.1230107

[12] Tsuneya Kurihara and Natsuki Miyata. 2004. Modeling Deformable Human Hands from Medical Images. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '04)*. Eurographics Association, Goslar, DEU, 355–363. https://doi.org/10.1145/1028523.1028571

[13] Binh Huy Le and Zhigang Deng. 2014. Robust and Accurate Skeletal Rigging from Mesh Sequences. *ACM Trans. Graph.* 33, 4, Article 84 (July 2014), 10 pages. https://doi.org/10.1145/2601097.2601161

[14] J. P. Lewis, Matt Cordner, and Nickson Fong. 2000. Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-driven Deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 165–172. https://doi.org/10.1145/344779.344862

[15] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. 2017. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Washington, DC, USA, 2261–2269. https://doi.org/10.1109/CVPR.2017.243

[16] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. 2019. Can GCNs Go as Deep as CNNs? (2019). arXiv:1904.03751

[17] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018. Adaptive Graph Convolutional Neural Networks. In *Proc. 32nd AAAI Conference on Artificial Intelligence*. AAAI Press, Palo Alto, CA, 3546–3553.

[18] Lijuan Liu, Youyi Zheng, Di Tang, Yi Yuan, Changjie Fan, and Kun Zhou. 2019. NeuroSkinning: Automatic Skin Binding for Production Characters with Deep Graph Networks. *ACM Trans. Graph.* 38, 4, Article 114 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3322969

[19] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graph.* 34, 6, Article Article 248 (Oct. 2015), 16 pages. https://doi.org/10.1145/2816795.2818013

[20] Ran Luo, Tianjia Shao, Huamin Wang, Weiwei Xu, Xiang Chen, Kun Zhou, and Yin Yang. 2018. NNWarp: Neural Network-Based Nonlinear Deformation. *IEEE Transactions on Visualization and Computer Graphics* (2018), 14. https://doi.org/10.1109/TVCG.2018.2881451 Early Access.

[21] Nadia Magnenat-Thalmann, Richard Laperrière, and Daniel Thalmann. 1988. Joint-dependent Local Deformations for Hand Animation and Object Grasping. In *Proceedings on Graphics Interface '88*. Canadian Information Processing Society, Toronto, Ont., Canada, 26–33. http://dl.acm.org/citation.cfm?id=102313.102317

[22] Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011. Efficient Elasticity for Character Skinning with Contact and Collisions. *ACM Trans. Graph.* 30, 4, Article 37 (July 2011), 12 pages. https://doi.org/10.1145/2010324.1964932

[23] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein. 2017. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Washington, DC, USA, 5115–5124. https://doi.org/10.1109/CVPR.2017.576

[24] Junjun Pan, Lijuan Chen, Yuhan Yang, and Hong Qin. 2018. Automatic Skinning and Weight Retargeting of Articulated Characters Using Extended Position-Based Dynamics. *The Visual Computer* 34, 10 (2018), 1285–1297.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., USA, 5998–6008.

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph Attention Networks. arXiv:1710.10903

[27] Hongyi Xu and Jernej Barbič. 2016. Pose-space Subspace Dynamics. *ACM Trans. Graph.* 35, 4, Article 35 (July 2016), 14 pages. https://doi.org/10.1145/2897824.2925916

[28] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2018. Graph Neural Networks: A Review of Methods and Applications. arXiv:1812.08434