GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs

Jiani Zhang[1]*, **Xingjian Shi**[2]*, **Junyuan Xie**[3], **Hao Ma**[4], **Irwin King**[1], **Dit-Yan Yeung**[2]
[1]The Chinese University of Hong Kong, Hong Kong, China, {*jnzhang, king*}*@cse.cuhk.edu.hk*
[2]Hong Kong University of Science and Technology, Hong Kong, China, {*xshiab, dyyeung*}*@cse.ust.hk*
[3]Amazon Web Services, WA, USA, *junyuanx@amazon.com*
[4]Microsoft Research, WA, USA, *haoma@microsoft.com*

Abstract

We propose a new network architecture, *Gated Attention Networks* (GaAN), for learning on graphs. Unlike the traditional multi-head attention mechanism, which equally consumes all attention heads, GaAN uses a convolutional sub-network to control each attention head's importance. We demonstrate the effectiveness of GaAN on the inductive node classification problem on large graphs. Moreover, with GaAN as a building block, we construct the *Graph Gated Recurrent Unit* (GGRU) to address the traffic speed forecasting problem. Extensive experiments on three real-world datasets show that our GaAN framework achieves state-of-the-art results on both tasks.

1 INTRODUCTION

Many crucial machine learning tasks involve graph-structured datasets, such as classifying posts in a social network (Hamilton et al., 2017a), predicting interfaces between proteins (Fout et al., 2017) and forecasting the future traffic speed in a road network (Li et al., 2018). The main difficulty in solving these tasks is how to find the right way to express and exploit the graph's underlying structural information. Traditionally, this is achieved by calculating various graph statistics like degree and centrality, using graph kernels, or extracting human engineered features (Hamilton et al., 2017b).

Recent research, however, has pivoted to solving these problems by *graph convolution* (Duvenaud et al., 2015; Atwood and Towsley, 2016; Kipf and Welling, 2017; Fout et al., 2017; Hamilton et al., 2017a; Veličković et al., 2018; Li et al., 2018), which generalizes the standard definition of convolution over a regular grid topology (Gehring et al., 2017; Krizhevsky et al., 2012) to 'convolution' over graph structures. The basic idea behind 'graph convolution' is to develop a localized parameter-sharing operator on a set of neighboring nodes to aggregate a local set of lower-level features. We refer to such an operator as a *graph aggregator* (Hamilton et al., 2017a) and the set of local nodes as the *receptive field* of the aggregator. Then, by stacking multiple graph aggregators, we build a deep neural network model which can be trained end-to-end to extract the local and global features across the graph. Note that we use the spatial definition instead of the spectral definition (Hammond et al., 2011; Bruna et al., 2014) of graph convolution because the full spectral treatment requires eigen-decomposition of the Laplacian matrix, which is computationally intractable on large graphs, while the localized versions (Defferrard et al., 2016; Kipf and Welling, 2017) can be interpreted as graph aggregators (Hamilton et al., 2017a).

Graph aggregators are the basic building blocks of graph convolutional neural networks. A model's ability to capture the structural information of graphs is largely determined by the design of its aggregators. Most existing graph aggregators are based on either pooling over neighborhoods (Kipf and Welling, 2017; Hamilton et al., 2017a) or computing a weighted sum of the neighboring features (Monti et al., 2017). In essence, functions that are permutation invariant and can be dynamically resizing are eligible graph aggregators. One class of such functions is the neural attention network (Bahdanau et al., 2015), which uses a subnetwork to compute the correlation weight of the elements in a set. Among the family of attention models, the multi-head attention model has been shown to be effective for machine translation task (Lin et al., 2017; Vaswani et al., 2017). It has later been adopted as a graph aggregator to solve the node classification problem (Veličković et al., 2018). A single attention head sums up the elements that are similar

* These two authors contributed equally.

to the query vector in one representation subspace. Using multiple attention heads allows exploring features in different representation subspaces, which provides more modeling power in nature. However, treating each attention head equally loses the opportunity to benefit from some attention heads which are inherently more important than others. For instance, assume there are ten different relationships in the graph and only two of them are valid for each node. If we use ten attention heads to model these relationships and treat them equally, each node will still aggregate features from eight non-existent subspaces. This will mislead the final prediction.

To this end, we propose the *Gated Attention Networks* (GaAN) for learning on graphs. GaAN uses a small convolutional subnetwork to compute a soft gate at each attention head to control its importance. Unlike the traditional multi-head attention that admits all attended contents, the gated attention can modulate the amount of attended content via the introduced gates. From this perspective, the gate-generation network acts as a high-level controller that determines how to aggregate the features extracted by the attention heads. Moreover, since only a simple and light-weighted subnetwork is introduced in constructing the gates, the computational overhead is negligible and the model is easy to train. We demonstrate the effectiveness of our new aggregator by applying it to the inductive node classification problem. To train our model and other graph aggregators on relatively large graphs, we also improve the sampling strategy introduced in (Hamilton et al., 2017a) to reduce the memory cost and increase the run-time efficiency. Furthermore, since our proposed aggregator is very general, we extend it to construct a *Graph Gated Recurrent Unit* (GGRU), which is directly applicable for spatiotemporal forecasting problem. Extensive experiments on two node classification datasets, PPI and the large Reddit dataset (Hamilton et al., 2017a), and one traffic speed forecasting dataset, METR-LA (Li et al., 2018), show that GaAN consistently outperforms the baseline models and achieves the state-of-the-art performance.

In summary, our main contributions include: (a) a new multi-head attention-based aggregator with additional gates on the attention heads; (b) a unified framework for transforming graph aggregators to graph recurrent neural networks; and (c) the state-of-the-art prediction performance on three real-world datasets. To the best of our knowledge, this is the first work to study the attention-based aggregators on large and spatiotemporal graphs.

2 NOTATIONS

We denote vectors with bold lowercase letters, matrices with bold uppercase letters, and sets with calligraphy letters. We denote a single fully-connected layer with a non-linear activation $\alpha(\cdot)$ as $\text{FC}^{\alpha}_{\theta}(\mathbf{x}) = \alpha(\mathbf{W}\mathbf{x} + \mathbf{b})$, where $\theta = \{\mathbf{W}, \mathbf{b}\}$ are the parameters. Also, θ with different subscripts mean different transformation parameters. For activation functions, we denote $h(\cdot)$ to be the LeakyReLU activation (Xu et al., 2015a) with negative slope equals to 0.1 and $\sigma(\cdot)$ to be the sigmoid activation. $\text{FC}_{\theta}(\mathbf{x})$ means applying no activation function after the linear transform. We denote $\oplus$ as the concatenation operation and $||_{k=1}^{K} \mathbf{x}_k$ as sequentially concatenating $\mathbf{x}_1$ through $\mathbf{x}_K$. We denote the Hadamard product as '$\circ$' and the dot product between two vectors as $\langle \cdot, \cdot \rangle$.

3 RELATED WORK

In this section, we will review relevant research on learning on graphs. Our model is also related to many graph aggregators proposed by previous work. We will discuss these aggregators in Section 4.3.

Neural attention mechanism Neural attention mechanism is widely adopted in deep learning literature and many variants have been proposed (Xu et al., 2015b; Seo et al., 2017; Zhang et al., 2017; Vaswani et al., 2017; Cheng et al., 2017; Zhang et al., 2018). Among them, our model takes inspiration from the multi-head attention architecture proposed in (Vaswani et al., 2017). Given a query vector $\mathbf{q}$ and a set of key-value pairs $\{(\mathbf{k}_1, \mathbf{v}_1), ..., (\mathbf{k}_n, \mathbf{v}_n)\}$, a single attention head computes a weighted combination of the value vectors $\sum_{i=1}^{n} w_i \mathbf{v}_i$. The weights are generated by applying softmax to the inner product between the query and keys, i.e., $\mathbf{w} = \text{softmax}(\{\mathbf{q}^T\mathbf{k}_1, ..., \mathbf{q}^T\mathbf{k}_n\})$. In the multi-head case, the outputs of K different heads are concatenated to form an output vector with fixed dimensionality. The difference between the proposed GaAN and the multi-head attention mechanism is that we compute additional gates to control the importance of each head's output.

Graph convolutional networks on large graph Applying graph convolution on large graphs is challenging because the memory complexity is proportional to the total number of nodes, which could be hundreds of thousands of nodes in large graphs (Hamilton et al., 2017a). To reduce memory usage and computational cost, (Hamilton et al., 2017a) proposed the GraphSAGE framework that uses a sampling algorithm to select a small subset of the nodes and edges. On each iteration, GraphSAGE first uniformly samples a mini-batch of nodes. Then, for each node, only a fixed number of neighborhoods are selected for aggregation. More recently, Chen et al. (Chen et al., 2018) proposed a new sampling method that randomly samples two sets of nodes according to a proposed distribution. However, this method is only applicable to one aggregator,

i.e., the *Graph Convolutional Network* (GCN) (Kipf and Welling, 2017). For the usage of the gate mechanism, the gate in Li et al. (2016) refers to the gate in Gated Recurrent Units, which are imposed on the activations of the neural network, while our gates are added to the attention heads to control each head's relative importance.

Graph convolution networks for spatiotemporal forecasting Recently, researchers have applied graph convolution, which is commonly used for learning on static graphs, to spatiotemporal forecasting (Yuan et al., 2017). (Seo et al., 2016) proposed *Graph Convolutional Recurrent Neural Network* (GCRNN), which replaced the fully-connected layers in LSTM (Hochreiter and Schmidhuber, 1997) with the ChebNet operator (Defferrard et al., 2016), and applied it to a synthetic video prediction task. Li et al. (Li et al., 2018) proposed *Diffusion Convolutional Recurrent Neural Network* (DCRNN) to address the traffic forecasting problem, where the goal is to predict future traffic speeds in a sensor network given historical traffic speeds and the underlying road graph. DCRNN replaces the fully-connected layers in GRU (Chung et al., 2014) with the diffusion convolution operator (Atwood and Towsley, 2016). Furthermore, DCRNN takes the direction of graph edges into account. The difference between our GGRU with GCRNN and DCRNN is that we proposed a unified method for constructing a recurrent neural network based on an arbitrary graph aggregator rather than proposing a single model.

4 GATED ATTENTION NETWORKS

In this section, we first give a generic formulation of graph aggregators followed by the multi-head attention mechanism. Then, we introduce the proposed gated attention aggregator. Finally, we review the other kinds of graph aggregators proposed by previous work and explain their relationships with ours.

Generic formulation of graph aggregators Given a node i and its neighboring nodes $\mathcal{N}_i$, a graph aggregator is a function γ in the form of $\mathbf{y}_i = \gamma_\Theta(\mathbf{x}_i, \{\mathbf{z}_{\mathcal{N}_i}\})$, where $\mathbf{x}_i$ and $\mathbf{y}_i$ are the input and output vectors of the center node i. $\mathbf{z}_{\mathcal{N}_i} = \{\mathbf{z}_j | j \in \mathcal{N}_i\}$ is a set of the reference vectors in the neighboring nodes and Θ is the learnable parameters of the aggregator. In this paper, we do not consider aggregators that use edge features. However, it is straightforward to incorporate edges in our definition by defining $\mathbf{z}_j$ to contain the edge feature vectors $\mathbf{e}_{i,j}$.

4.1 MULTI-HEAD ATTENTION AGGREGATOR

We linearly project the center node feature $\mathbf{x}_i$ to get the query vector and project the neighboring node features to get the key and value vectors. We then apply the multi-head attention mechanism (Vaswani et al., 2017) to obtain the final aggregation function. For the multi-head attention mechanism, different heads capture features from different representation subspaces. The detailed formulation of the multi-head attention aggregator is as follows:

$$\begin{aligned}
\mathbf{y}_i &= \mathrm{FC}_{\theta_o}(\mathbf{x}_i \oplus \Big\|_{k=1}^{K} \sum_{j \in \mathcal{N}_i} w_{i,j}^{(k)} \mathrm{FC}^h_{\theta_v^{(k)}}(\mathbf{z}_j)), \\
w_{i,j}^{(k)} &= \frac{\exp(\phi_w^{(k)}(\mathbf{x}_i, \mathbf{z}_j))}{\sum_{l=1}^{|\mathcal{N}_i|} \exp(\phi_w^{(k)}(\mathbf{x}_i, \mathbf{z}_l))}, \\
\phi_w^{(k)}(\mathbf{x}, \mathbf{z}) &= \langle \mathrm{FC}_{\theta_{xa}^{(k)}}(\mathbf{x}), \mathrm{FC}_{\theta_{za}^{(k)}}(\mathbf{z}) \rangle.
\end{aligned} \tag{1}$$

Here, K is the number of attention heads. $w_{i,j}^{(k)}$ is the kth attentional weights between the center node i and the neighboring node j, which is generated by applying a softmax to the dot product values. $\theta_{xa}^{(k)}$, $\theta_{za}^{(k)}$ and $\theta_v^{(k)}$ are the parameters of the kth head for computing the query, key, and value vectors, which have dimensions of d_a, d_a and d_v respectively. The K attention outputs are concatenated with the input vector and passed to a fully-connected layer parameterized by θ_o to get the final output $\mathbf{y}_i$, which has dimension d_o. The difference between our aggregator and that in GAT (Veličković et al., 2018) is that we have adopted the key-value attention mechanism and the dot product attention while GAT does not compute additional value vectors and uses a fully-connected layer to compute $\phi_w^{(k)}$.

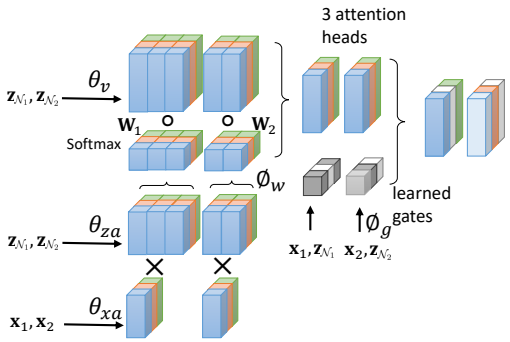
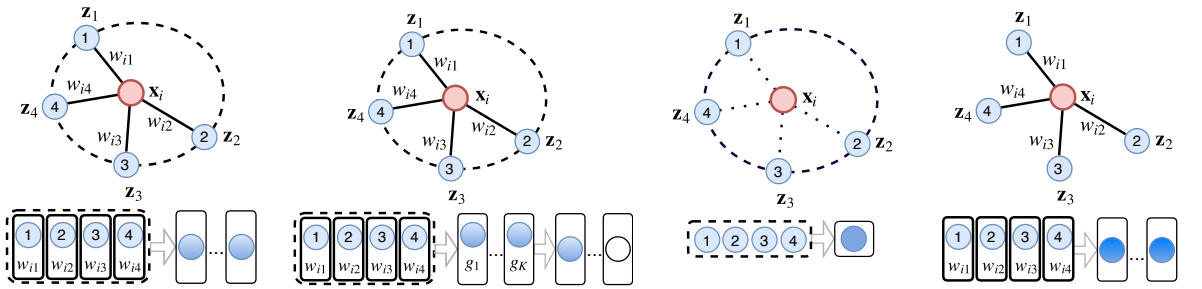


Figure 1: Illustration of a three-head gated attention aggregator with two center nodes in a mini-batch. $|\mathcal{N}_1| = 3$ and $|\mathcal{N}_2| = 2$ respectively. Different colors indicate different attention heads. Gates in darker color stands for larger values. (Best viewed in color)

4.2 GATED ATTENTION AGGREGATOR

While the multi-head attention aggregator can explore multiple representation subspaces between the center node and its neighborhoods, not all of these subspaces are equally important; some subspaces may not even exist for specific nodes. Feeding the output of an attention



(a) Attention Aggregator (b) Gated Attention Aggregator (c) Pooling Aggregator (d) Pairwise Sum Aggregator

Figure 2: Comparison of different graph aggregators. The aggregators are drawn for only one aggregation step. The nodes in red are center nodes and the nodes in blue are neighboring nodes. The bold black lines between the center node and neighbor nodes indicate that a learned pairwise relationship is used for calculating the relative importance. The oval in dash line around the neighbors means the interaction among neighbors is utilized when determining the weights. (Best viewed in color)

head that captures a useless representation can mislead the mode's final prediction.

Therefore, we compute an additional soft gate between 0 (low importance) and 1 (high importance) to assign different importance to each head. In combination with the multi-head attention aggregator, we get the formulation of the gated attention aggregator:

$$\begin{aligned} \mathbf{y}_i &= \mathrm{FC}_{\theta_o}(\mathbf{x}_i \oplus \Big\|_{k=1}^{K} (g_i^{(k)} \sum_{j \in \mathcal{N}_i} w_{i,j}^{(k)} \mathrm{FC}^h_{\theta_v^{(k)}}(\mathbf{z}_j))), \\ \mathbf{g}_i &= [g_i^{(1)}, ..., g_i^{(K)}] = \psi_g(\mathbf{x}_i, \mathbf{z}_{\mathcal{N}_i}), \end{aligned} \tag{2}$$

where $g_i^{(k)}$ is a scalar, the gate value of the kth head at node i. To make sure adding gates will not introduce too many additional parameters, we use a convolutional network ψ_g that takes the center node and neighboring node features as the input to generate the gate values. All the other parameters have the same meanings as in Eqn. (1).

There are multiple possible designs of the ψ_g network. In this paper, we combine average pooling and max pooling to construct the network. The detailed formula is:

$$\mathbf{g}_i = \mathrm{FC}^{\sigma}_{\theta_g}(\mathbf{x}_i \oplus \max_{j \in \mathcal{N}_i}(\{\mathrm{FC}_{\theta_m}(\mathbf{z}_j)\}) \oplus \frac{\sum_{j \in \mathcal{N}_i} \mathbf{z}_j}{|\mathcal{N}_i|}). \tag{3}$$

Here, θ_m maps the neighbor features to a d_m dimensional vector before taking the element-wise max and θ_g maps the concatenated features to the final K gates. By setting a small d_m, the subnetwork for computing the gates will have negligible computational overhead. A visual illustration of GaAN aggregator's structure can be found in Figure 1. Also, we compare the general structures of the multi-head attention aggregator and the gated attention aggregator in Figure 2a and Figure 2b.

4.3 OTHER GRAPH AGGREGATORS

Most previous graph aggregators except attention-based aggregators can be summarized into two general categories: graph pooling aggregators and graph pairwise sum aggregators. In this section, we first describe these two types of aggregators and then explain their relationship with the attention-based aggregator. Finally, we give a list of the baseline aggregators used in the experiments.

Graph pooling aggregators The main characteristic of graph pooling aggregators is that they do not consider the correlation between neighboring nodes and the center node. Instead, neighboring nodes' features are directly aggregated and the center node's feature is simply concatenated or added to the aggregated vector and then passed through an output function ϕ_o:

$$\mathbf{y}_i = \phi_o(\mathbf{x}_i \oplus \mathrm{pool}_{j \in \mathcal{N}_i}(\phi_v(\mathbf{z}_j))). \tag{4}$$

Here, the projection function ϕ_v and the output function ϕ_o can be a single fully-connected layer and the pool$(\cdot)$ operator can be average pooling, max pooling, or sum pooling. The majority of existing graph aggregators are special cases of the graph pooling aggregators. Some models only integrate the node features of neighborhoods (Duvenaud et al., 2015; Kipf and Welling, 2017; Hamilton et al., 2017a), while others integrated edge features as well (Atwood and Towsley, 2016; Fout et al., 2017; Schütt et al., 2017). In Figure 2c, we illustrate the architecture of the graph pooling aggregators.

Graph pairwise sum aggregators Like attention-based aggregators, graph pairwise sum aggregators also aggregate the neighborhood features by taking K weighted sums. The difference is that the weight between node i and its neighbor j is not related to the other neighbors in $\mathcal{N}_i$. The formula of graph pairwise sum aggregator is given as follows:

$$\mathbf{y}_i = \phi_o(\mathbf{x}_i \oplus \Big\|_{k=1}^{K} \sum_{j \in \mathcal{N}_i} w_{i,j}^{(k)} \phi_v^{(k)}(\mathbf{z}_j)),$$
$$w_{i,j}^{(k)} = \phi_w^{(k)}(\mathbf{x}_i, \mathbf{z}_j). \quad (5)$$

Here, $w_{i,j}^{(k)}$ is only related to the pair $\mathbf{x}_i$ and $\mathbf{z}_j$, while in attention-based models $w_{i,j}^{(k)}$ is related to features of all neighbors $\mathbf{z}_{\mathcal{N}_i}$. Models like the adaptive forget gate strategy in *Graph LSTM* (Liang et al., 2016) and MoNet (Monti et al., 2017) employed pairwise sum aggregators with a single head or multiple heads. In Figure 2d, we illustrate the architecture of the graph pairwise sum aggregators.

Baseline aggregators To fairly evaluate the effectiveness of GaAN against previous work, we choose two representative aggregators in each category as baselines:

- **Avg. pooling**: $\mathbf{y}_i = \text{FC}_{\theta_o}(\mathbf{x}_i \oplus \text{pool}_{j \in \mathcal{N}_i}^{\text{avg}}(\text{FC}_{\theta_v}^h(\mathbf{z}_j)))$.
- **Max pooling**: $\mathbf{y}_i = \text{FC}_{\theta_o}(\mathbf{x}_i \oplus \text{pool}_{j \in \mathcal{N}_i}^{\max}(\text{FC}_{\theta_v}^h(\mathbf{z}_j)))$.
- **Pairwise + sigmoid**:
$$\mathbf{y}_i = \text{FC}_{\theta_o}(\mathbf{x}_i \oplus \Big\|_{k=1}^{K} \sum_{j \in \mathcal{N}_i} w_{i,j}^{(k)} \text{FC}_{\theta_v^{(k)}}^h(\mathbf{z}_j)),$$
$$w_{i,j}^{(k)} = \frac{1}{|\mathcal{N}_i|} \sigma(\langle \text{FC}_{\theta_{xa}^{(k)}}(\mathbf{x}_i), \text{FC}_{\theta_{za}^{(k)}}(\mathbf{z}_j) \rangle).$$
- **Pairwise + tanh**: Replace the sigmoid activation in **Pairwise + sigmoid** to tanh.

5 INDUCTIVE NODE CLASSIFICATION

5.1 MODEL

In the inductive node classification task, every node is assigned one or multiple labels. During training, the validation and testing nodes are not observable. And the goal is to predict the labels of the unseen testing nodes. Our approach follows that of (Hamilton et al., 2017a), where a mini-batch of nodes are sampled on each iteration during training and multiple layers of graph aggregators are stacked to compute the predictions.

With a stack of M layers of graph aggregators, we will first sample a mini-batch of nodes $\mathcal{B}_0$ and then recursively expand $\mathcal{B}_\ell$ to be $\mathcal{B}_{\ell+1}$ by sampling the neighboring nodes of $\mathcal{B}_\ell$. After M sampling steps, we can get a hierarchy of node batches: $\mathcal{B}_1, ..., \mathcal{B}_M$. The node representations, which are initialized to be the node features, will be aggregated in reverse order from $\mathcal{B}_M$ to $\mathcal{B}_0$. The representations of the last layer, i.e., the final representations of the nodes in $\mathcal{B}_0$, are projected to get the output. We use the sigmoid activation for multi-label classification and the softmax activation for multi-class classification. Also, we use the cross-entropy loss to train the model.

Table 1: Effect of the merge operation. Both methods sample a maximum of 15 neighborhoods without replacement for three recursive steps on the Reddit dataset. We start from 512 seed nodes. The total number of nodes after the lth sampling step is denoted as $|\mathcal{B}_\ell|$. The sampling process is repeated for ten times and the mean is reported.

Strategy/Sample Step	$\lvert\mathcal{B}_0\rvert$	$\lvert\mathcal{B}_1\rvert$	$\lvert\mathcal{B}_2\rvert$	$\lvert\mathcal{B}_3\rvert$
Sample without merge	512	7.8K	124.4K	1.9M
Sample and merge	512	7.5K	70.7K	0.2M

A naive sampling algorithm is always to sample all neighbors. However, it is not practical on large graphs because the memory complexity is $O(|\mathcal{V}|)$ and the time complexity is $O(|\mathcal{E}|)$, where $|\mathcal{V}|$ and $|\mathcal{E}|$ are the total number of nodes and edges. Instead, similar to GraphSAGE (Hamilton et al., 2017a), we only sample a subset of the neighborhoods for each node. In our implementation, at the ℓth sampling step, we sample $\min(|\mathcal{N}_i|, S_\ell)$ neighbors without replacement for the node i, where S_ℓ is a hyperparameter that controls the maximum number of sampled neighbors at the ℓth step. Moreover, to improve over GraphSAGE and further reduce memory cost, we merge repeated nodes that are sampled from different seeds' neighborhoods within each mini-batch. This greatly reduces the size of $\mathcal{B}_\ell$s as shown in Table 1.

Note that $\min(|\mathcal{N}_i|, S_\ell)$ is not the same for all the nodes i. Thus, instead of padding the sampled neighborhood set to the same size for utilizing fast tensor operation, we implemented new GPU kernels that directly operate on inputs with variable lengths to accelerate computations.

5.2 EXPERIMENTAL SETUP

We performed a thorough comparison of GaAN with the state-of-the-art models, five aggregator-based models in our framework and a two-layer fully connected neural network on the PPI and Reddit datasets (Hamilton et al., 2017a). The five baseline aggregators include the multi-head attention aggregator, two pooling based aggregators, and two pairwise sum based aggregators mentioned in Section 4.3. We also conducted comprehensive ablation analysis.

The PPI dataset was collected from the molecular signatures database (Subramanian et al., 2005). Each node represents a protein and edges represent the interaction between proteins. Labels represent the cellular functions of each protein from gene ontology. Reddit is an online discussion forum where users can post and discuss contents on different topics. Each node represents a post and

Table 2: Datasets for inductive node classification. 'multi' stands for multilabel classification and 'single' otherwise.

Data	#Nodes	#Edges	#Fea	#Classes
PPI	56.9K	806.2K	50	121(multi)
Reddit	233.0K	114.6M	602	41(single)

two nodes are connected if they are commented by the same user. The labels indicate the community a post belongs to. Detailed statistics are listed in Table 2.

5.3 MODEL ARCHITECTURES AND IMPLEMENTATION DETAIL

The GaAN and other five aggregator-based networks are stacked with two graph aggregators. Each aggregator is followed by the LeakyReLU activation with negative slope equals to 0.1 and a dropout layer with dropout rate set to be 0.1. The output dimension d_o of all layers are fixed to be 128 except when we compare the relative performance with different output dimensions. To keep the number of parameters comparable for the multi-head models with a different number of heads, we fix the product of the dimension of the value vector and the number of heads, i.e., $d_v \times K$ to be the same when evaluating the effect of varying the number of heads. Also, the hyper-parameters of the first and the second layer are assumed to be the same if no special explanation is given.

In the PPI experiments, both pooling aggregators have $d_v = 512$, where d_v means the dimensionality of the value vector projected by θ_v. For the pairwise sum aggregators, the dimension of the keys d_a is set to be 24, $d_v = 64$, and $K = 8$. For both GaAN and the multi-head attention based aggregator, d_a is set to be 24 and the product $d_v \times K$ is fixed to be 256. For GaAN, we set d_m to be 64 in the gate-generation network. Also, we use the entire neighborhoods in the mini-batch training algorithm. In the Reddit experiments, both pooling aggregators have $d_v = 1024$. For the pairwise sum aggregators, $d_a = 32$, $d_v = 256$ and $K = 4$. For the attention based aggregators, d_a is set to be 32 and $d_v \times K$ is fixed to be 512. We set the gate-generation network in GaAN to have $d_m = 64$. Also, the number of heads is fixed to 1 in the first layer for both attention-based models. The maximum number of sampled neighbors in the first and second sampling steps are denoted as S_1 and S_2 and are respectively set to be 25 and 10 in Table 3. In the ablation analysis, we also show the performance when setting them to be (50, 20), (100, 40), and (200, 80).

To illustrate the effectiveness of incorporating graph structures, we also evaluate a two-layer fully-connected neural network with the hidden dimension of 1024 and ReLU activation. It only takes node features as input and ignores graph structures.

Table 3: Summary of different models' test micro F1 scores in the inductive node classification task. In the first block, we include the best-reported results in the previous papers. In the second block, we report the results obtained by our models. For the PPI dataset, we do not use any sampling strategies. For the Reddit dataset, we use the maximum number sampling strategy with S_1=25 and S_2=10.

Models / Datasets	PPI	Reddit
GraphSAGE (Hamilton et al., 2017a)	(61.2)[1]	95.4
GAT (Veličković et al., 2018)	97.3 ± 0.2	-
Fast GCN (Chen et al., 2018)	-	93.7
2-Layer FNN	54.07±0.06	73.58±0.09
Avg. pooling	96.85±0.19	95.78±0.07
Max pooling	98.39±0.05	95.62±0.03
Pairwise+sigmoid	98.39±0.05	95.86±0.08
Pairwise+tanh	98.32±0.18	95.80±0.03
Attention-only	98.46±0.09	96.19±0.07
GaAN	**98.71±0.02**	**96.36±0.03**

We train all the aggregator-based models with Adam (Kingma and Ba, 2015) and early stopping on the validation set. Besides, we use the validation set to perform learning rate decay scheduler. For Reddit, before training we normalize all the features and project all the features to a hidden dimension of 256. The initial learning rate is 0.001 and gradually decreases to 0.0001 with the decay rate of 0.5 each time the validation F1 score does not decrease in a window of 4 epochs and early stopping occurs for 10 epochs. The gradient normalization value clips no larger than 1.0. For the PPI dataset, all the input features are projected to a 64-dimension hidden state before passing to the aggregators. The learning rate begins at 0.01 and decays to 0.001 with the decay rate of 0.5 if the validation F1 score does not increase for 15 epochs and stops training for 30 epochs.

The training batch size is fixed to be 512. Also, in all experiments, we use the validation set to select the optimal hyperparameters for training. The training, validation, and testing splits are the same as that in (Hamilton et al., 2017a). The micro-averaged F1 score is used to evaluate the prediction accuracy for both datasets. We repeat the training five times for Reddit and three times for PPI with different random seeds and report the average test F1 score along with the standard deviation.

[1]The performance reported in the paper is relatively low because the author has not trained their model into convergence. Also, it is not fair to compare it with the other scores because it uses the sampling strategy while the others have not.

Table 4: Comparison of the test F1 score on the Reddit and PPI datasets with different sampling neighborhood sizes and attention head number K. S_1 and S_2 are the maximum number of sampled neighborhoods in the 1st and 2nd sampling steps. 'all' means to sample all the neighborhoods.

Models	Reddit					PPI	
	#Param	S_1, S_2 25,10	S_1, S_2 50,20	S_1, S_2 100,40	S_1, S_2 200,80	#Param	S_1, S_2 all, all
2-Layer FNN	1.71M	73.58±0.09	73.58±0.09	73.58±0.09	73.58±0.09	1.23M	54.07±0.06
Avg. pooling	866K	95.78±0.07	96.11±0.07	96.28±0.05	96.35±0.02	274K	96.85±0.19
Max pooling	866K	95.62±0.03	96.06±0.09	96.18±0.11	96.33±0.04	274K	98.39±0.05
Pairwise+sigmoid	965K	95.86±0.08	96.19±0.04	96.33±0.05	96.38±0.08	349K	98.39±0.05
Pairwise+tanh	965K	95.80±0.03	96.11±0.05	96.26±0.03	96.36±0.04	349K	98.32±0.18
Attention-only-K1	562K	96.15±0.06	96.40±0.05	96.48±0.02	96.54±0.07	168K	96.31±0.08
Attention-only-K2	571K	96.19±0.07	96.40±0.04	96.52±0.02	96.57±0.02	178K	97.36±0.08
Attention-only-K4	587K	96.11±0.06	96.40±0.02	96.49±0.03	96.56±0.02	196K	98.09±0.07
Attention-only-K8	620K	96.10±0.03	96.38±0.01	96.50±0.04	96.53±0.02	233K	98.46±0.09
GaAN-K1	620K	96.29±0.05	96.50±0.08	96.67±0.04	96.73±0.05	201K	96.95±0.09
GaAN-K2	629K	96.33±0.02	96.59±0.02	96.71±0.05	96.82±0.05	211K	97.92±0.05
GaAN-K4	645K	**96.36±0.03**	**96.60±0.03**	96.73±0.04	**96.83±0.03**	230K	98.42±0.02
GaAN-K8	678K	96.31±0.13	**96.60±0.02**	**96.75±0.03**	96.79±0.08	267K	**98.71±0.02**

5.4 MAIN RESULTS

We compare our model with the previous state-of-the-art methods on inductive node classification. This includes GraphSAGE (Hamilton et al., 2017a), GAT (Veličković et al., 2018), and FastGCN (Chen et al., 2018). The GraphSAGE model used a 2-layer sample and aggregate model with a neighborhood size of $S^{(1)} = 25$ and $S^{(2)} = 10$ without dropout. The 3-layer GAT model consisted of 4, 4, and 6 heads in the first, second, and third layer respectively. Each attention head had 256 dimensions. GAT did not use neighborhood sampling, L2 regularization, or dropout. The FastGCN model is a fast version of the 3-layer, 128-dimension GCN with sampled neighborhood size being 400, 100, and 400 for each layer and no sampling is done during testing.

Table 3 summarizes all results of the state-of-the-art models and the models proposed in this paper. We denote the multi-head attention aggregator as 'Attention-only' in the tables and figures. We find that the proposed model, GaAN, achieves the best F1 score on both datasets and the other baseline aggregators can also show competitive results to the state-of-the-art. We note that aggregator-based models produce much higher F1 score than the fully-connected model, which shows the effectiveness of the graph aggregators. Our max pooling and avg. pooling baselines have higher scores on Reddit than that in the original GraphSAGE record. This mainly contributes to our usage of dropout and the LeakyReLU activation.

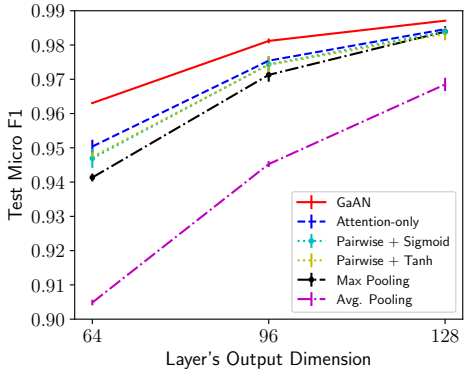
Regarding the training time, the average training time of the attention-only model for the first 100 epochs on PPI is 36.5s and that of GaAN is 37.0s when we run on the machine with a single TitanX GPU and Intel Xeon CPU 3.70 GHz. This shows that the computational overhead of adding the gates is negligible.

5.5 ABLATION ANALYSIS

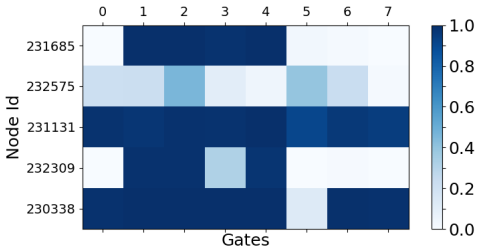
We ran some ablation experiments to analyze the performance of different graph aggregators when different hyperparameters were used. We also visualized the gates of the GaAN model.

Effect of the number of attention heads and the sample size We compare the performance of the aggregators when a different number of attention heads and sampling strategies are used. Results are shown in Table 4. We find that attention-based models consistently outperform pooling and pairwise sum based models with the fewer number of parameters, which demonstrates the effectiveness of the attention mechanism in this task. Moreover, GaAN consistently beats the multi-head attention model with the same number of attention heads K. This proves that adding additional gates to control the importance of the attention heads is beneficial to the final classification performance. From the last two row blocks of Table 4, we note that increasing the number of attention heads will not always produce better results on Reddit. In contrast, on PPI, the larger the K, the better the prediction results. Also, we can see steady improvement with larger sampling sizes, which is consistent with the observation in (Hamilton et al., 2017a).

Effect of output dimensions in the PPI dataset We changed the output dimension to be 64, 96, and 128 in the models for training in the PPI dataset. The test F1 score



(a) Performance of different models with a varying number of output dimensions on PPI.



(b) Visualization of 8 gate values of 5 example nodes on Reddit. Each row represents a learned gate vector for one node.

Figure 3: Ablation analysis on PPI and Reddit

is shown in Figure 3a. All multi-head models have K=8. We find that the performance becomes better for larger output dimensions and the proposed GaAN consistently outperforms the other models.

Visualization of gate values In Figure 3b, we visualized the gate values of five different nodes output by the GaAN-K8 model trained on the Reddit dataset. It illustrates the diversity of the learned gate combinations for different nodes. In most cases, the gates vary across attention heads, which shows that the gate-generation network can be learned to assign different importance to different heads.

6 TRAFFIC SPEED FORECASTING

6.1 GRAPH GRU

Following (Lin et al., 2017), we formulate traffic speed forecasting as a spatiotemporal sequence forecasting problem where the input and the target are sequences defined on a fixed spatiotemporal graph, e.g., the road network. To simplify notations, we denote $\mathbf{Y} = \Gamma_\Theta(\mathbf{X}, \mathbf{Z}; \mathcal{G})$ as applying the γ aggregator for all nodes in $\mathcal{G}$, i.e., $\mathbf{y}_i = \gamma_\Theta(\mathbf{x}, \mathbf{z}_{\mathcal{N}_i})$. Based on a given graph aggregator Γ, we can construct a GRU-like RNN structure

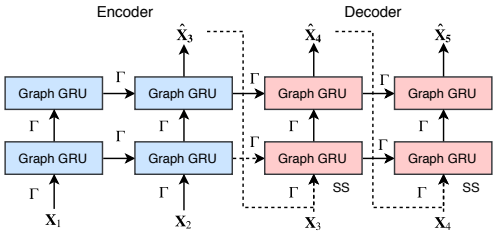


Figure 4: Illustration of the encoder-decoder structure used in the paper. We use two layers of Graph GRUs to predict a length-3 output sequence based on a length-2 input sequence. 'SS' denotes the scheduled sampling step.

Table 5: The Dataset used for traffic speed forecasting.

Data	#Nodes	#Edges	#Timestamps
METR-LA	207	1,515	34,272

using the following equations:

$$
\begin{aligned}
\mathbf{U}_t &= \sigma(\Gamma_{\Theta_{xu}}(\mathbf{X}_t, \mathbf{X}_t; \mathcal{G}) + \Gamma_{\Theta_{hu}}(\mathbf{X}_t \oplus \mathbf{H}_{t-1}, \mathbf{H}_{t-1}; \mathcal{G})), \\
\mathbf{R}_t &= \sigma(\Gamma_{\Theta_{xr}}(\mathbf{X}_t, \mathbf{X}_t; \mathcal{G}) + \Gamma_{\Theta_{hr}}(\mathbf{X}_t \oplus \mathbf{H}_{t-1}, \mathbf{H}_{t-1}; \mathcal{G})), \\
\mathbf{H}'_t &= h(\Gamma_{\Theta_{xh}}(\mathbf{X}_t, \mathbf{X}_t; \mathcal{G}) + \mathbf{R}_t \circ \Gamma_{\Theta_{hh}}(\mathbf{X}_t \oplus \mathbf{H}_{t-1}, \mathbf{H}_{t-1}; \mathcal{G})), \\
\mathbf{H}_t &= (1 - \mathbf{U}_t) \circ \mathbf{H}'_t + \mathbf{U}_t \circ \mathbf{H}_{t-1}.
\end{aligned} \tag{6}
$$

Here, $\mathbf{X}_t \in \mathbb{R}^{|\mathcal{V}| \times d_i}$ are the input features and $\mathbf{H}_t \in \mathbb{R}^{|\mathcal{V}| \times d_o}$ are the hidden states of the nodes at the tth timestamp. $|\mathcal{V}|$ is the total number of nodes, d_i is the dimension of the input, and d_o is the dimension of the state. $\mathbf{U}_t$ and $\mathbf{R}_t$ are the update gate and reset gate that controls how $\mathbf{H}_t$ is calculated. $\mathcal{G}$ is the graph that defines the connection structure between all the nodes.

We refer to this RNN structure as *Graph GRU* (GGRU). GGRU can be used as the basic building block for RNN encoder-decoder structure (Lin et al., 2017) to predict the future K steps of traffic speeds, i.e., $\hat{\mathbf{X}}_{J+1}, \hat{\mathbf{X}}_{J+2}, ..., \hat{\mathbf{X}}_{J+K}$, based on the previous J steps of observed traffic speeds, i.e., $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_J$. In the decoder, we use the scheduled sampling (Bengio et al., 2015) technique described in (Lin et al., 2017). Figure 4 illustrates our encoder-decoder structure. When attention-based aggregators are used, i.e., the multi-head attention aggregator or our GaAN aggregator, the connection structure in the recurrent step will also be learned based on the attention process. This can be viewed as an extension of *Trajectory GRU* (TrajGRU) (Shi et al., 2017) on irregular or graph-structured data.

6.2 EXPERIMENTAL SETUP

To evaluate the proposed GGRU model on traffic speed forecasting, we use the METR-LA dataset from (Li et al.,

Table 6: Performance comparison of different models for traffic speed forecasting on the METR-LA dataset. Models marked with '†' treat sensor map as a directed graph while other models convert it into an undirected graph. Scores under "τmin" are the scores at the $\frac{\tau}{5}$th predicted frame. The last three columns contain the average scores of the 15 min, 30 min, and 60 min forecasting horizons.

Models / T	15 min			30 min			60 min			Average		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
FC-LSTM (Li et al., 2018)	3.44	6.30	9.6%	3.77	7.23	10.9%	4.37	8.69	13.2%	3.86	7.41	11.2%
GCRNN (Li et al., 2018)	2.80	5.51	7.5%	3.24	6.74	9.0%	3.81	8.16	10.9%	3.28	6.80	9.13%
DCRNN† (Li et al., 2018)	2.77	5.38	7.3%	3.15	6.45	8.8%	**3.60**	**7.60**	**10.5%**	3.17	6.48	8.87%
Avg Pool	2.79	5.42	7.26%	3.20	6.52	8.84%	3.69	7.69	10.73%	3.22	6.54	8.94%
Max Pool	2.77	5.36	7.21%	3.18	6.45	8.78%	3.69	7.73	10.80%	3.21	6.51	8.93%
Pairwise + Sigmoid	2.76	5.36	7.14%	3.18	6.46	8.72%	3.70	7.73	10.77%	3.22	6.52	8.88%
Pairwise + Tanh	2.76	5.34	7.14%	3.18	6.46	8.73%	3.70	7.73	10.73%	3.21	6.51	8.87%
Attention-only	2.74	5.33	7.09%	3.16	6.45	8.69%	3.67	7.61	10.77%	3.19	6.49	8.85%
GaAN	**2.71**	**5.24**	**6.99%**	**3.12**	**6.36**	**8.56%**	3.64	7.65	10.62%	**3.16**	**6.41**	**8.72%**

2018). The nodes in the dataset represent sensors measuring traffic speed and edges denote proximity between sensor pairs measured by road network distance. The sensor speeds are recorded every five minutes. Complete dataset statistics are given in Table 5. We follow (Li et al., 2018)'s way to split the dataset. The first 70% of the sequences are used for training, the middle 10% are used for validation, and the final 20% are used for testing. We also use the same evaluation metrics as in (Li et al., 2018) for evaluation, including *Mean Absolute Error* (MAE), *Root Mean Squared Error* (RMSE), and *Mean Absolute Percentage Error* (MAPE). A sequence of length 12 is used as the input to predict the future traffic speed in one hour (12 steps).

6.3 MAIN RESULTS

We compare six variations of the proposed GGRU architecture with three baseline models, including fully-connected LSTM, GCRNN, and DCRNN (Li et al., 2018). We use the same set of six aggregators as in the inductive node classification experiment to construct the GGRU and we use two layers of GGRUs with the state dimension of 64 both in the encoder and the decoder. For attention based models, we set $K = 4$, $d_a = 16$, and $d_v = 16$. For GaAN, we set $d_m = 64$ and only use max pooling in the gate-generation network. For pooling based aggregators, we set $d_v = 128$. For pairwise sum aggregators, we set $K = 4$, $d_a = 32$, and $d_v = 16$.

Since the road map is directed and our model does not deal with edge information, we first convert the road map into an undirected graph and use it as the $\mathcal{G}$ in Eqn. (6). All models are trained by minimizing MAE loss with Adam optimizer. The initial learning rate is set to 0.001 and the batch-size is 64. We use the same scheduled sampling strategy as in (Li et al., 2018). Table 1 shows the comparison of different approaches for 15 minutes, 30 minutes and 1 hour ahead forecasting on both datasets.

The scores for 15 minutes, 30 minutes, and 1 hour ahead forecasting as well as the average scores over three forecasting horizons are shown in Table 6. For the average score, we can see that the proposed GGRU models consistently give better results than GCRNN, which models the traffic network as an undirected graph. Moreover, the GaAN based GGRU model, which does not use edge information, achieves higher accuracy than DCRNN, which uses edge information in the road network.

7 CONCLUSION AND FUTURE WORK

We introduced the GaAN model and applied it to two challenging tasks: inductive node classification and traffic speed forecasting. GaAN beats previous state-of-the-art algorithms in both cases. In the future, we plan to extend GaAN by integrating edge features and processing massive graphs with millions or even billions of nodes. Moreover, our model is not restricted to graph learning. A particularly exciting direction for future work is to apply GaAN to natural language processing tasks like machine translation.

8 ACKNOWLEDGEMENT

The work described in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 and No. HKUST 16207316 of the General Research Fund) and 2016 MOE of China (Project No. D.01.16.00101).

References

J. Atwood and D. Towsley. Diffusion-convolutional neural networks. In *NIPS*, pages 1993–2001, 2016.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, pages 1171–1179, 2015.

J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

J. Chen, T. Ma, and C. Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *ICLR*, 2018.

J. Cheng, S. Zhao, J. Zhang, I. King, X. Zhang, and H. Wang. Aspect-level sentiment classification with heat (hierarchical attention) network. In *CIKM*, pages 97–106, 2017.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS, Workshop on Deep Learning and Representation Learning*, 2014.

M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016.

D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, pages 2224–2232, 2015.

A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur. Protein interface prediction using graph convolutional networks. In *NIPS*, pages 6533–6542, 2017.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *ICML*, pages 1243–1252, 2017.

W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1025–1035, 2017a.

W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017b.

D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2): 129–150, 2011.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. In *ICLR*, 2016.

Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*, 2018.

X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan. Semantic object parsing with graph lstm. In *ECCV*, pages 125–143, 2016.

Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. In *ICLR*, 2017.

F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pages 5115–5124, 2017.

K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8:13890, 2017.

M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. In *ICLR*, 2017.

Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson. Structured sequence modeling with graph convolutional recurrent networks. *arXiv preprint arXiv:1612.07659*, 2016.

X. Shi, Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. In *NIPS*, pages 5622–5632, 2017.

A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102 (43):15545–15550, 2005.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, pages 6000–6010, 2017.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.

B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015a.

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057, 2015b.

Y. Yuan, X. Liang, X. Wang, D. Yeung, and A. Gupta. Temporal dynamic graph LSTM for action-driven video object detection. In *ICCV*, pages 1819–1828, 2017.

H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal. Context encoding for semantic segmentation. In *CVPR*, 2018.

J. Zhang, X. Shi, I. King, and D.-Y. Yeung. Dynamic key-value memory networks for knowledge tracing. In *WWW*, pages 765–774, 2017.