Improving Graph Networks through Selection-based Convolution

David Hart
East Carolina University
hartda23@ecu.edu

Bryan Morse
Brigham Young University
morse@cs.byu.edu

Abstract

Graph Convolutional Networks (GCNs) provide a general framework that can learn in a variety of data domains, such as 3D geometry, social networks, and chemical structures. GCNs, however, often ignore intrinsic relationships among nodes in the graph, and these relationships need to be learned indirectly during the training process through mechanisms such as attention or local-kernel approximation. This paper introduces selection-based graph convolution, a method for preserving these intrinsic relationships within the graph convolution operator which provides improved performance over attention-based counterparts on various tasks. We demonstrate the effectiveness of selection to improve the performance of many types of GCNs on tasks such as spatial graph classification. Furthermore, we demonstrate the ability to improve state-of-the-art graph networks for road traffic estimation and molecular property prediction.

1. Introduction

Graph Neural Networks (GNNs) have shown incredible power to learn with irregular data that can be represented with graph structures. Early on, spectral methods were used to learn with these graphs [6,9]. Later, Graph Convolutional Networks (GCNs) were introduced, which employ simple neighborhood aggregation functions [19, 23, 31]. These allowed for building multi-layer networks similar to prevalent 2D image convolutional networks. Common methods use attention or other metrics to determine how much to weight incoming node features during an aggregation step [5, 43, 44]. Some even use multi-layer perceptrons within the aggregation steps themselves [8, 45, 51].

Because of the generality of GCNs, they are used for a large variety of data domains, including 3D geometry, social networks, and chemical structures. This generality, however, makes it difficult to represent certain kinds of information intrinsically in the graph. For example, points in 3D data have natural spatial relationships relative to each other (above, below, in front of, etc.), but this spatiality cannot be represented in the graph structure itself. Instead, this information is represented with positional encodings that could be included with the node features, and the network has to learn the spatial relationships indirectly. Alternatively, a specially designed graph network must be used to approximate local kernels around nodes in 3D space [4, 12, 32]. As another example, many graphs contain data where it would be beneficial to have different kinds of edges for connections to different kinds of data, such as distinguishing spatial dimensions from the temporal features [24] or representing different bond types in molecules [50].

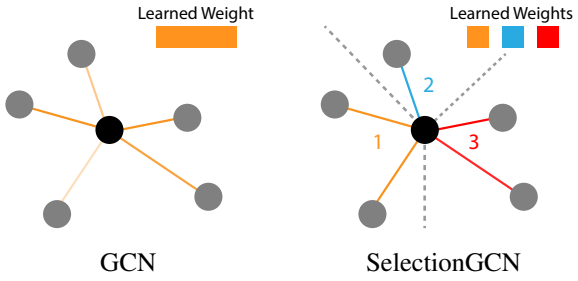


Figure 1. In Graph Convolutional Networks (left), a single learned weight matrix or function is applied to all nodes during the aggregation step, using fixed or learned edge weights between 0 and 1. In contrast, Selection-based Graph Convolutional Networks (right) partition the graph edges into multiple groups to allow different weights to be applied to different kinds of edges at each channel of the node features.

This paper explores the possibility of maintaining these intrinsic relationships through the process of *selection-based convolution*, which was introduced for images in [20]. In a regular graph convolution, a single learned weight matrix or function is applied to all nodes during the aggregation step. Fixed or learned edge weights may determine the amount of influence a node has during the aggregation, but this is limited to a single value in the range $[0, 1]$ for each target node. In contrast, in selection-based convolution, the graph edges are partitioned into multiple groups to allow different weights to be applied to different kinds of edges without breaking the permutation-invariant constraint

of graph aggregation operations. Unlike traditional edge weights, these weights are applied per channel on node features and can have any real-numbered value, including negative ones. This gives the network more expressive power in solving graph tasks. The differences between regular and selection-based graph convolution are illustrated in Fig. 1.

In [20], it was shown that graphs could maintain spatial-relationship information by use of a particular selection function, which is used for preprocessing the graph and partitioning edges into multiple adjacency matrices. This selection function matched each edge to the closest cardinal or ordinal direction, thus maintaining a structure similar to a 3×3 convolution for images. This approach was used to perform convolution in the 2D domain for irregular image types, such as those with discontinuities caused by projection or arbitrary domain limitations. In this work, by comparison, we generalize that approach to different kinds of spatial graph data, as well as explore other graph-based representations that benefit from selection-based convolution.

Previous research has proposed adding relationship information to graph convolutional networks, specifically for knowledge graphs [7, 38], but partitioning edges into groups using a selection function is specific to the data, not to the network architecture that is used. Thus, we show that *selection can improve the performance of specific tasks regardless of the kind of graph network*. Those tasks include situations where spatial or other attribute information may be available, and we present possible selection functions for each kind of data. Additionally, we demonstrate how to incorporate selection into common network designs. We compare our results to common attention-based methods and show that selection can have superior performance while using fewer parameters. Finally, we show the effectiveness of our results by improving the performance of state-of-the-art graph convolutional networks. Specifically, we incorporate selection into TGCN [54], a recurrent network designed for traffic prediction, and DimeNet [16], a GCN designed specifically for analyzing molecules and predicting their quantum properties.

In summary, our contributions are as follows:

- We demonstrate how to incorporate selection into many common graph network designs.
- We propose selection functions that incorporate directional, distance, and other attribute information into graph convolutions.
- We demonstrate the ability of selection to improve the performance of many graph networks and outperform attention-based counterparts on various tasks.
- We show that selection improves the performance of state-of-the-art GCNs on tasks such as road traffic estimation and molecular property prediction.

2. Related Work

2.1. Graph Convolutional Networks

Graph Convolutional Networks using deep learning were first introduced by Kipf *et al*. [23] with a basic learned weight multiplier during the aggregation step. Many methods build on this technique through improving the message passing system. These techniques incorporate higher-order aggregations [19, 31], use attention-based approaches [5, 43, 44], and utilize MLPs within the aggregation function [8, 45, 51]. For a survey and overview of Graph Neural Network techniques, see [47].

Of particular interest to our work are Relational and Selection-based Networks. Relational Graph Convolution [38] incorporated known node relationships into the aggregation step of the convolution for knowledge-graph tasks. This allowed the network to assign different weights to each relation type. Other methods have improved this aggregation [7] and used it for various applications [11, 24]. SelectionConv [20, 21] used a formulation similar to [38], but specifically applied the graph networks to image and surface data by determining relationships (i.e., directions) using a selection function as described further in Sec. 3.

2.2. Common Graph Network Tasks

In this work we demonstrate our method applied to a variety of graph-based problems. We summarize here relevant research on each such task.

Spatial Graph Classification

3D mesh and point-cloud classification is a common task for graph networks. PointNet [33] was a foundational work for using multi-layer perceptrons to learn on point clouds, and was further expanded in [34]. Wang *et al*. [45] achieve state-of-the-art performance by dynamically building a graph from point clouds based on feature values at each layer of the network. More recently, many have addressed the need for an anisotropic convolution that approximates a local kernel similar to 2D CNNs [4, 12, 32]. Other researchers have focused on using attention mechanisms and transformers within spatial graphs and point clouds [25, 35, 53].

Traffic Prediction

Since early in the age of deep learning, the task of road traffic prediction has been approached using neural networks [29, 46]. Many have proposed using recurrent networks to handle the temporal nature of the data [2, 27, 54] while others use explicit temporal convolutions [18, 52]. Attention is also used to increase performance [1, 18]. For this work, we use a spatio-temporal network called TGCN [54]. This method uses a recurrent network to learn over time steps, while using a GCN as an aggregator over the spatial

information. For a survey of traffic prediction techniques, see [22].

Molecular Property Prediction

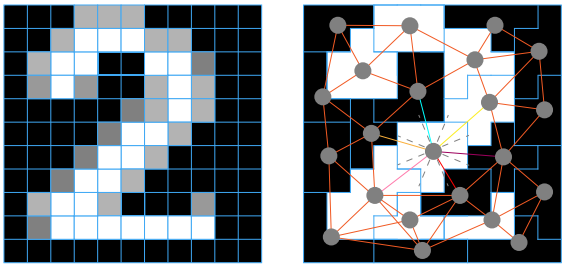
Many chemistry datasets map naturally to a graph structure, whether it be representing protein structures as in the PPI dataset [56] or molecules as in the QM9 dataset [50]. Gilmer *et al*. [17] were one of the first to introduce a message passing scheme designed specifically for molecules. Schutt *et al*. [39] proposed a continuous filter space for describing properties such as energy throughout the molecule. DimeNet [15, 16] uses radial and spherical basis functions to more accurately capture direction and angle between the atoms, with further improvements made in [14]. Rotation and translation equivariant graph neural networks have also been used [3, 40] with current state-of-the-art results using spherical message passing [28].

3. Selection in Graph Convolution

As introduced in [20], selection-based convolution partitions the edges of a graph based on relationships between adjacent nodes. Edges are assigned to an individual partition by a *selection function* designed for a specific problem domain. The partition an edge is assigned to is called its *selection*. These selections are then used as the basis for a set of unique learned weights. Thus, *node features are aggregated with different learned weights according to the assigned selection of their connecting edges.*

As a simple illustration of why one would want to incorporate selection into a network, consider the task of graph classification on MNIST Superpixels [30]. This dataset takes images from the original MNIST dataset [10] and represents them as superpixel regions. A single node is used to represent each region, and nearby nodes are connected together as shown in Fig. 2. Each node contains a binary label (0 for black, 1 for white) and the position of the centroid of the region. The desired output is the digit (0 through 9) represented in the original image.

To solve this task, a standard GCN could be designed that takes as input the label and position of each node. As demonstrated later in this section, most kinds of GCNs would perform poorly on this task since they perform a naive aggregation across all nearby nodes. Even though position is included as input, there is no intrinsic understanding of locations in the network. In comparison, using a selection-based approach, positional relationships could be preserved during the convolution. This would be done by preprocessing the edges based on the direction that they feed into the source node. One partition could contain all the edges coming from the left, another all the edges coming from the bottom right, etc. By giving those different kinds of edges different learned weights, the directional relationship between nodes is preserved during the convolution. This is represented by the different edge colors shown on the right side of Fig. 2.



Original Image Superpixel Graph

Figure 2. Example superpixel graph. Regions of similar pixels are grouped together and represent a single node in the graph. Nearby nodes are connected to each other. Our selection function then partitions edges based on their closest cardinal or ordinal direction.

Though this is a simple example, there are many cases where selection naturally fits into a graph problem. In order to use selection-based graph convolution, the graph network that is used for the task needs to employ selection information. We now demonstrate how to augment many common graph convolution operators to include selection information.

3.1. Selection-based GCNs

The primary way to augment a GCN to incorporate selection is to multiply incoming node features by unique learned weights during the aggregation step. For example, consider the common message passing system used in [31], which is

$$\mathbf{x}'_i = \mathbf{W}_a \mathbf{x}_i + \mathbf{W}_b \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j \tag{1}$$

where $\mathbf{x}_i$ is the vector of source features for node i, each $\mathbf{x}_j$ is the target feature vector for corresponding node j in the neighborhood $\mathcal{N}(i)$, and $\mathbf{x}'_i$ is the resulting feature vector at the source node for the next layer. The weights $\mathbf{W}_a$ and $\mathbf{W}_b$ are learned during the training process. Note in this form that all neighboring nodes $j \in \mathcal{N}(i)$ are treated identically.

To add selection information to this convolution layer, we first define a selection function $\sigma(i, j)$ that assigns the edge between node i and adjacent node j to one of a set of disjoint groups based on properties relevant to the problem. We denote the set of all possible selection values as $S = \{0, 1, 2, ..., s_{\text{max}}\}$ such that $\sigma(i, j) = s \in S$. Once these are defined, we can modify the graph network to associate learned weights to specific selection groups and complete an additional summation across selections. This updates our message passing system to be

$$\mathbf{x}'_i = \mathbf{W}_a \mathbf{x}_i + \mathbf{W}_b \sum_{s \in S} \mathbf{W}_s \sum_{j \in \mathcal{N}_s(i)} \mathbf{x}_j \tag{2}$$

where $\mathbf{W}_s$ is a learned weight for each selection s in the set S of all possible selections, and the neighborhood $\mathcal{N}(i)$ is partitioned into multiple selection-based neighborhoods $\mathcal{N}_s(i)$ based on each selection s. Thus, selection weights $\mathbf{W}_s$ are applied selectively to the nodes in $\mathcal{N}_s(i)$ first before they are aggregated and multiplied by $\mathbf{W}_b$.

In general, the message passing scheme for all convolution-based graph networks [13] can be written as

$$\mathbf{x}'_i = \gamma_{\mathbf{\Theta}} \left(\mathbf{x}_i, \square_{j \in \mathcal{N}(i)} \, \phi_{\mathbf{\Theta}} \left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{j,i} \right) \right) \tag{3}$$

where $\mathbf{e}_{j,i}$ is a possible edge weight, $\gamma_{\mathbf{\Theta}}$ and $\phi_{\mathbf{\Theta}}$ are differentiable functions with possible learned weights, and $\square$ denotes a differentiable and permutation-invariant aggregation function such as sum, mean, or max. To incorporate selection, we use the same summation over selection weights as in Eq. 2. A selection-based message passing scheme is thus

$$\mathbf{x}'_i = \gamma_{\mathbf{\Theta}} \left(\mathbf{x}_i, \sum_{s \in S} \square_{j \in \mathcal{N}_s(i)} \, \phi_{\mathbf{\Theta}} \left(\mathbf{x}_i, \mathbf{W}_s \mathbf{x}_j, \mathbf{e}_{j,i} \right) \right) \tag{4}$$

which modifies the original message passing scheme in Eq. 3 to include aggregation across selection without loss of generality.

Note that these selection weights are different from edge weights that are sometimes included in graphs. This is because edge weights are usually a single value between 0 and 1 describing how much influence the node should have during an aggregation, whereas selection weights are learned, can be positive or negative, and apply the weight per channel. This provides a nonlinear piecewise aggregation that has more expressive power than fixed or learned edge weights.

To illustrate this approach, we add selection to multiple graph convolutional networks and test their performance on the MNIST Superpixel [30] classification task described previously. For the regular graph networks, the input includes both the label and position of each node. For the selection-based graph networks, node position is used to process the selections, but only the node labels are given as input to the network. We use an 8-way directional selection function that assigns edges based on the best alignment of the spatial relationship between the superpixels and the eight cardinal/ordinal directions (or zero if extremely close to the connected node) as described in more detail in Sec. 4.1.

We compare performance of various baseline GCNs to various selection-based GCNs. We use the naming conventions of PyTorch Geometric [13] for each type of convolution and denote their selection-based enhancements with "Sel". This comparison is shown in Fig. 3. Additional results are presented in Sec. 5.1.

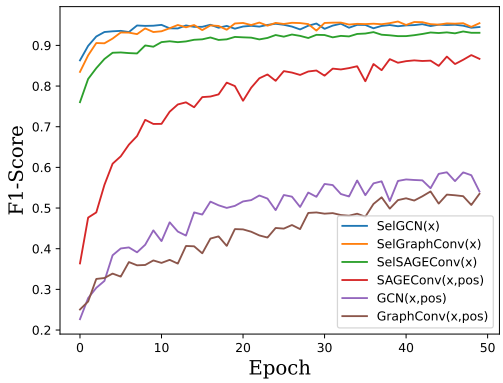


Figure 3. Performance of various networks on MNIST Superpixels [30]. The regular networks had both the label (x) and the position (pos) information, while the selection-based networks only had the label as input and used position only for preprocessing the selections. Note that the selection-based networks perform better than their regular counterparts even though they feed less information into the network.

As can be seen in this simple example, adding selection to these networks gives them superior performance. Specifically, note how a network such as GCNConv [23] performs poorly on this task, but its selection-equivalent (SelGCNConv) performs well. In Sec. 5.1, we show that selection-based network performance is comparable to state-of-the art approaches such as [45] despite omitting explicit position information.

3.2. Why assigned selection over learned attention?

The description of selection may remind readers of attention-based graph networks such as GAT [44]. The multiple selection weights are similar to the multiple heads in an attention-based network, the main difference being that selection weights are assigned while attention heads are learned. So what benefit does a predetermined selection have over a modifiable attention? While attention may perform better when given general graph structures to work with, if the graph has inherent properties that are easy to understand and mathematically define, selection may be preferable to attention because it more directly incorporates the intuition that we often hope attention mechanisms will eventually learn.

Selection can perform similarly or superior to attention-based networks while having fewer parameters.

A single learned selection weight can be thought of as having a similar number of parameters to a single head on a multi-headed attention framework. Attention networks,

however, often concatenate features and add additional linear layers to handle the aggregation of the learned features. Thus, selection-based networks generally take less memory than their attention-based counterparts, but this reduction of parameters does not mean a reduction in performance. For example, as we will show in Sec. 5.2, our selection-based traffic-prediction networks outperform attention-based networks while using fewer parameters.

Selection can be included in complex network designs that attention cannot.

Attention for graphs is designed to look at properties of node features. In comparison, selection functions can be designed to look at node features, edge features, or even groups of nodes together and can be added to existing networks with just a few lines of code (which are provided in the supplemental material). For example, DimeNet [16] is a complex graph network that is designed for molecule classification and operates on groups of edge and even angles between triplets of atoms. It would be quite difficult to incorporate attention into such a framework, but in Sec. 5.3 we will show that it is simple to add selection and improve the performance on this network.

4. Selection Functions for Graphs

In our previous example, we used an 8-directional selection function to enforce spatial relationships between nodes in a graph, but selection can be generalized to work with any graph data given a properly-designed selection function. We now explain and generalize directional selection and demonstrate possible selection functions for various types of graph data.

4.1. Directional Selections

Given an edge between two nodes in a graph, the original selection function presented in [20] determines the cardinal/ordinal direction that the edge most closely aligns with. This can be mathematically written as

$$\sigma(i,j) = \begin{cases} 0 \text{ if } \|\mathbf{x}_j - \mathbf{x}_i\| < \epsilon \\ \underset{k}{\operatorname{argmax}}\ \mathbf{D}_k \cdot (\mathbf{x}_j - \mathbf{x}_i) \text{ otherwise} \end{cases} \tag{5}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the spatial coordinates of nodes i and j and $\mathbf{D}_k$ is list of the unit vectors for the 8 different directions of interest. If we are working with 2D graph data, we can use this same selection function, but since we are not restricted like [20] to match the 3×3 convolution kernels of images, we can use any set of unit vectors that are equally spaced by angle. Thus, we can include more or fewer directional vectors if desired. We show the results of such adjustments in Sec. 5.1.

We can extend this selection function to work with 3D data by expanding the list $\mathbf{D}_k$ accordingly. One possible formulation is to take the unit vectors formed by all combinations of -1, 0, and 1 for each dimension, giving 26 vectors that align with the various faces, edges, and corners of a cube, as shown in Fig. 4. (We exclude the null vector $[0,0,0]$ and account for this explicitly in Eq. 5.) Again, any set of equally-spaced unit vectors could theoretically be used, but we empirically find this to be an effective set as demonstrated in Sec. 5.1.

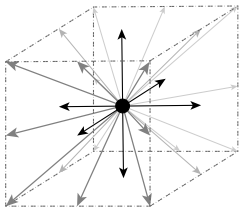


Figure 4. The 26 vectors representing the different directions of interest in a 3D selection function. The vectors are normalized during selection calculations.

Limitations of Directional Selections

As often pointed out in related work [19, 33], 3D data usually does not maintain a canonical orientation between samples in the dataset. Because of the ambiguous rotation, directional selections would be ineffective in such a case without using techniques as such learned data alignment [33]. Thus, we specifically experiment on datasets where a consistent global orientation exists. Our method could also be applied to datasets where a local orientation exists for each node, such as on surfaces or other manifolds.

4.2. Distance Selections

In addition to the spatial direction associated with incoming edges, some tasks may be sensitive to the distance between the target and source nodes. In the task of road traffic prediction, for example, one would expect during a particular time step that nearby nodes would have greater effect on local traffic than distant nodes. Distances are often used to inform edge weights in traffic datasets [27], but this simply dilutes messages in most aggregation schemes, and distances are otherwise not accounted for. A selection function provides more expressive depth to the GCN through its piecewise aggregation of learned weights. This creates a nonlinear function over distance.

One possible formulation of a distance selection function is to bin distances of similar value, forming various equivalence classes. Mathematically, this would have the form

$$\sigma(i,j) = \left\lceil b \frac{\sqrt{(\mathbf{x}_j - \mathbf{x}_i)^2}}{d_{\max}} \right\rceil \tag{6}$$

where b is the number of desired bins and $d_{\max}$ is the maximum distance between nodes in the graph. Note that nodes

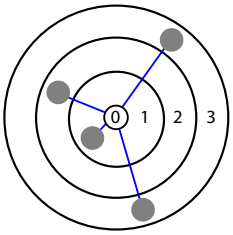


Figure 5. A distance selection function bins nodes based on their distance from the source node.

that are a distance of 0 apart (i.e., nodes connected to themselves) receive their own bin, then the rest of the distances are linearly separated, creating $b+1$ partitions as illustrated in Fig. 5. Depending on the data, other binning strategies such as quadratic or log-based separations could be used by preprocessing the distance values.

4.3. Other Selections

Many other possible selection functions exist based on the specific nature of the graph data that is being used. A selection function can be designed whenever there is a node or edge property that can be divided into different groups. For example, temporal selection could be used to separate past, present, and future nodes for spatiotemporal graph data. This could especially benefit dynamic graphs such as those generated in [24]. Another example includes pairings of node attributes, such relationships between members of different age groups in a social network. Selection functions can even be introduced into more complex networks that learn on groups of nodes at a time. As we will show in Sec. 5.3, a selection function can be used to distinguish between different angles between atoms in a molecule. This selection function bins angles together in fashion similar to that presented in Eq. 6. As long as there is an intuitive property that can be retrieved before the aggregation step of a network, a selection function can be used.

4.4. Combined Selection Functions

Any of the described selection functions can be combined to increase usability. For example, the directional selection function could be combined with the distance selection function for increased performance in spatial tasks. A spatial selection function could be combined with temporal selections for time-dependant graphs. There may be multiple node attributes you wish to compare at the same time. All of these are scenarios where a combined selection function could be beneficial.

To combine two selection functions together, a new selection number is assigned for each possible pairing of the selections. For selection functions σ_1 and σ_2, this gives

$$\sigma(i,j) = \sigma_1(i,j) + |S_1|\,\sigma_2(i,j) \tag{7}$$

where $|S_1|$ is the number of possible selections for σ_1. In this combined selection function, the new number of learned weights is equal to the product $|S_1|\,|S_2|$ of the number of selections for each original function.

5. Results

We demonstrate the effectiveness of our approach in the following tasks: Spatial Graph Classification, Traffic Prediction, Molecular Property Prediction. We pick these particular tasks because they all have intrinsic spatial properties or other attributes that are well suited for the selection functions described in Sec. 4.

5.1. Spatial Graph Classification

First, we show additional results for MNIST Superpixels [30] in Table 1. Also, we also show performance on the CoMA 3D Faces dataset [36], which contains 3D meshes of faces in 12 different expressions. We train various GCNs on the expression classification task. These results are provided in Table 2. For these tasks, we use the 2D and 3D directional selection functions from Sec. 4.1. As described in Sec. 3.1, we use the naming conventions of PyTorch Geometric [13] for each type of convolution and denote their selection-based enhancements with "Sel". An ablation study of selection functions is provided for the SelGraphConv network.

Additional ablation studies of selection functions as well as our training configurations are provided in the accompanying supplemental materials.

Table 1. F1 scores on the test set for MNIST superpixels [30]. Note that selection allows many networks to have near state-of-the-art performance, even without explicitly receiving position information as input. Also note that EdgeConv and GAT, common SOTA methods, cannot perform well without the position information.

Network	Input	Selection	F1 Score
GCNConv [23]	label + pos	-	0.645
SelGCNConv	label	8 Directions	**0.959**
SAGEConv [19]	label + pos	-	0.893
SelSAGEConv	label	8 Directions	**0.940**
GraphConv [31]	label + pos	-	0.625
SelGraphConv	label	4 Distances	0.681
SelGraphConv	label	4 Directions	0.955
SelGraphConv	label	8 Directions	0.961
SelGraphConv	label	12 Directions	0.963
SelGraphConv	label	8 Dir + 4 Dist	**0.968**
GINConv [51]	label + pos	-	0.966
SelGINConv	label	8 Directions	**0.970**
EdgeConv [45]	label + pos	-	**0.987**
EdgeConv	label	-	0.594
GAT [44]	label + pos	-	0.979
GAT	label	-	0.739

Table 2. F1 scores on the expression classification task for the test set of CoMA [36]. The input for network was 3D position data. Note that selection-based methods outperform GAT while using fewer parameters and are competitive with a network such as EdgeConv that was designed specifically for 3D data.

Network	Selection	F1 Score
GCNConv [23]	-	0.282
SelGCNConv	26 Directions	**0.770**
SAGEConv [19]	-	0.150
SelSAGEConv	26 Directions	**0.742**
GraphConv [31]	-	0.522
SelGraphConv	6 Directions	0.615
SelGraphConv	26 Directions	**0.713**
SelGraphConv	6 Directions + 4 Distances	0.682
GINConv [51]	-	0.682
SelGINConv	26 Directions	**0.779**
GAT [44]	-	0.600
EdgeConv [45]	-	**0.860**

5.2. Traffic Prediction

We show the performance of selection-based GCNs on the METR-LA dataset [27] which contains traffic data for highways in Los Angeles County. For this tasks, we use the distance-based selection function described in Sec. 4.2.

First, we show the ability of selection to improve basic spatiotemporal networks. We start with the recurrent network of TGCN [54] as our temporal backbone. We use the implementation provided by [37] which includes a basic attention-mechanism across time as introduced in [1]. In our experiment, we compare replacing the standard spatial graph convolution (GCNConv) in the network with our selection-based one (SelGCNConv). We also compare with using GAT as the spatial aggregator with four heads since SelGCNConv uses four distance-based weights. These results can be seen in Table 3.

We note three important findings from these results. First, selection-based graph convolution significantly outperforms standard convolution, even when the convolution uses distance-based edge weights. Second, we note that SelGCNConv out performs GAT while using fewer parameters. Last, we note that both standard convolution and attention-based convolution do not scale in performance as more layers are added to the spatial network. In comparison, our network continues to improve in performance when additional layers are used.

As another experiment, we modify the MSTGCN network proposed in [18] to include selection. In [18], ASTGCN implements an attention-based Chebyshev convolution for the spatial aggregator in the traffic network. MSTGCN is also described as its non-attention counterpart. By simply augmenting MSTGCN to include selection, we achieve superior performance to ASTGCN while having fewer parameters. This is shown in Table 4.

Finally, we specifically find that a five-layer deep version of the simple TGCN with SelGCNConv performs comparably with the state-of-the-art methods that rely on complex algorithms such as pretraining schemes and changing multi-layer graph structures. These results are shown in Table 5.

Table 3. Mean absolute error in prediction of traffic speed (in mph) after different amounts of time for the METR-LA dataset. The designation (EW) means that distance edge weights were used.

Network	Layers	# Params	15 Min	30 Min	60 Min
GCNConv [23]	1	26.1K	6.186	6.356	6.654
GCNConv (EW)	1	26.1K	5.265	5.591	6.100
GAT [44]	1	78.7K	3.206	3.788	4.632
SelGCNConv	1	75.3K	3.010	3.603	4.472
GCNConv	2	38.6K	6.366	6.479	6.674
GCNConv (EW)	2	38.6K	5.837	6.021	6.326
GAT	2	277K	3.332	3.860	4.670
SelGCNConv	2	136K	2.901	3.389	4.073
GCNConv	3	51.1K	6.225	6.331	6.521
GCNConv (EW)	3	51.1K	5.993	6.140	6.389
GAT	3	476K	3.272	3.780	4.512
SelGCNConv	3	198K	**2.814**	**3.212**	**3.707**

Table 4. Mean absolute error in prediction of traffic speed (in mph) after different amounts of time for the METR-LA dataset. Results are given for non-attention-based MSTGCN and attention-based ASTGCN as proposed in [18]. We compare this to our single layer and multi-layer selection-based SelMSTGCN which have superior performance.

Network	# Params	15 Min	30 Min	60 Min
MSTGCN	80.3K	2.908	3.432	4.142
ASTGCN	367K	2.869	3.382	4.052
SelMSTGCN-1L	112K	2.850	3.329	3.936
SelMSTGCN-3L	236K	**2.804**	**3.253**	**3.813**

5.3. Molecular Property Prediction

As another example task that benefits from selection, we use the QM9 dataset [50], which contains over 130,000 organic molecules descriptions with the target of estimating quantum properties such as Atomization Energy. We modify DimeNet [16], a state-of-the-art network designed for molecular data. This network utilizes both learned features on edges as well as angles between triplets of atoms. Thus, we experiment with an angle-based selection that employs a binning strategy as described in Sec. 4.2. For the angles, we select in 3 groups between 0° and 180° (60° increments). In the supplemental material, we also provide results for distance-based selections, as well as a combined selection function with both distance and angle using the technique described in Sec. 4.4.

We compare the original DimeNet to our modified network on multiple targets. In the original work, the authors trained DimeNet for over 1500 epochs with exponential learning rate decay and an exponential moving average on the weights of the network. We also use an exponential moving average, but found we could achieve similar perfor-

Table 5. Performance comparison on the METR-LA dateset using multiple error metrics for various state-of-the-art traffic prediction networks. (*) Numbers taken from updated appendix D of [41].

	15 Min			30 Min			60 Min		
Models	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
ARIMA [27]	3.99	8.21	9.60%	5.15	10.45	12.70%	6.90	13.23	17.40%
FC-LSTM [27]	3.44	6.30	9.60%	3.77	7.23	10.90%	4.37	8.69	13.20%
DCRNN [27]	2.77	5.38	7.30%	3.15	6.45	8.80%	3.60	7.60	10.50%
GMAN [55]	2.80	5.55	7.41%	3.12	6.49	8.73%	3.44	7.35	10.07%
SelGCN (ours)	2.80	5.15	7.57%	3.16	5.95	8.90%	3.59	6.81	10.58%
Graph WaveNet [49]	2.69	5.15	6.90%	3.07	6.22	8.37%	3.53	7.37	10.01%
MTGNN [48]	2.69	5.18	6.88%	3.05	6.17	8.19%	3.49	7.23	9.87%
TwoResNet [26]	2.67	5.13	6.83%	3.05	6.18	8.13%	3.49	7.32	9.67%
GTS* [41]	2.64	**4.95**	6.8-%	3.01	**5.85**	8.2-%	3.41	**6.74**	9.9-%
STEP [42]	**2.61**	4.98	**6.60%**	**2.96**	5.97	**7.96%**	**3.37**	6.99	**9.61%**

Table 6. The mean absolute error of DimeNet with various selection functions on multiple QM9 molecular targets. Results for SchNet are also provided as a baseline. Selection over angle lowers the error rate of DimeNet on many tasks when using a full-sized network. Also, our smaller selection networks can achieve similar results to the original network with far fewer parameters and less training time.

Target	Unit	SchNet [39]	DimeNet-small	SelDimeNet-small	DimeNet [16]	SelDimeNet
	# of params	-	534 K	608 K	2.1 M	2.4 M
	training time	-	21 h	24 h	32 h	48 h
μ	D	0.033	0.037	0.031	0.029	**0.028**
α	${a_0}^3$	0.235	0.059	0.051	0.047	**0.044**
ϵ_{HOMO}	meV	41	32.4	29.2	27.8	**23.1**
ϵ_{LUMO}	meV	34	23.0	21.0	19.7	**18.0**
$\langle R^2 \rangle$	${a_0}^2$	**0.073**	0.739	0.597	0.331	0.451
ZPVE	meV	1.7	1.37	**1.25**	1.29	**1.25**
U_0	meV	14	9.40	8.30	8.02	**7.88**
U	meV	19	9.95	8.29	7.89	**7.87**
H	meV	14	10.53	8.40	**8.11**	8.71
G	meV	14	10.97	**8.71**	8.98	9.11
c_v	$\frac{\text{cal}}{\text{mol K}}$	0.033	0.027	0.025	0.025	**0.023**

mance using 200 epochs with a faster exponential learning rate decay (from 10^{-3} to 10^{-5} over the 200 epochs). This shorter training scheme is necessary to conduct an effective ablation study over many targets and variations of the selection function. In all other ways, we match the original configuration and hyperparameters of the original work for all networks tested. Additional details are given in the accompanying supplemental materials. Lastly, we also experiment with small versions of the networks, where the hidden size at each layer is 64 instead of 128, which drastically reduces the number of parameters and training time for the network.

The mean absolute error of each network for each target is given in Table 6. As is shown, selection improves performance of the complex DimeNet network on many of the desired targets. Additionally, a smaller version of our selection-based network can achieve similar performance to the original DimeNet while using fewer than 1/3 as many parameters. We also note that selection increases error on some targets. This reinforces the idea that selection functions should be designed to match our intuition of the problem. A specific selection function is needed to improve performance, not an arbitrary one. For molecule datasets, it is possible that better selection functions could be designed by specialists to match scientific principles of chemistry.

6. Conclusion

We have presented a technique for adding intrinsic graph relationships into Graph Convolutional Networks, especially spatial data. We have demonstrated that incorporating selection can improve a variety of networks and even outperform attention-based networks. We have further demonstrated that our approach can improve the performance of state-of-the-art graph networks or perform comparably with smaller networks. Our framework is general and can continue to be used and extended by others to incorporate intuitive relational information into graph networks.

References

[1] Jiandong Bai, Jiawei Zhu, Yujiao Song, Ling Zhao, Zhixiang Hou, Ronghua Du, and Haifeng Li. A3T-GCN: Attention temporal graph convolutional network for traffic forecasting. *ISPRS International Journal of Geo-Information*, 10(7), 2021. 2, 7

[2] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. In *Advances in Neural Information Processing Systems*, volume 33, pages 17804–17815. Curran Associates, Inc., 2020. 2

[3] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1):2453, 2022. 3

[4] Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. In *International Conference on Machine Learning*, pages 748–758. PMLR, 2021. 1, 2

[5] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. 1, 2

[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations*, 2014. 1

[7] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. Relational graph attention networks. *arXiv preprint 1904.05811*, 2019. 2

[8] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020. 1, 2

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, 2016. 1

[10] Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 3

[11] Yan Ding, Xiaoqian Jiang, and Yejin Kim. Relational graph convolutional networks for predicting blood–brain barrier penetration of drug molecules. *Bioinformatics*, 38(10):2826–2831, 04 2022. 2

[12] Moshe Eliasof and Eran Treister. Diffgcn: Graph convolutional networks via differential operators and algebraic multigrid pooling. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. 1, 2

[13] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 4, 6

[14] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. In *Conference on Neural Information Processing Systems*, 2021. 3

[15] Johannes Gasteiger, Shankari Giri, Johannes T. Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. In *Machine Learning for Molecules Workshop, NeurIPS*, 2020. 3

[16] Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020. 2, 3, 5, 7, 8

[17] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017. 3

[18] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):922–929, Jul. 2019. 2, 7

[19] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017. 1, 2, 5, 6, 7

[20] David Hart, Michael Whitney, and Bryan Morse. SelectionConv: convolutional neural networks for non-rectilinear image data. In *European Conference on Computer Vision*, October 2022. 1, 2, 3, 5

[21] David Hart, Michael Whitney, and Bryan Morse. Interpolated SelectionConv for spherical images and surfaces. In *Winter Conference on Applications of Computer Vision*, pages 321–330. IEEE, January 2023. 2

[22] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207:117921, 2022. 3

[23] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. 1, 2, 4, 6, 7

[24] Eitan Kosman and Dotan Di Castro. GraphVid: It only takes a few nodes to understand a video. In *European Conference on Computer Vision*, page 195–212, Berlin, Heidelberg, 2022. Springer-Verlag. 1, 2, 6

[25] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 8500–8509. IEEE, June 2022. 2

[26] Danya Li, Semin Kwak, and Nikolas Geroliminis. Tworesnet: Two-level resolution neural network for traffic forecasting on freeway networks. In *International Conference on Intelligent Transportation Systems*, pages 3963–3969. IEEE, 2022. 8

[27] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018. 2, 5, 7, 8

[28] Yi Liu, Limei Wang, Meng Liu, Yuchao Lin, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical message passing for 3d molecular graphs. In *International Conference on Learning Representations*, 2022. 3

[29] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2014. 2

[30] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Conference on Computer Vision and Pattern Recognition*, pages 5115–5124. IEEE, 2017. 3, 4, 6

[31] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. *AAAI*, 33(01):4602–4609, 2019. 1, 2, 3, 6, 7

[32] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020. 1, 2

[33] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 652–660. IEEE, 2017. 2, 5

[34] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2

[35] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, and Kai Xu. Geometric transformer for fast and robust point cloud registration. In *Conference on Computer Vision and Pattern Recognition*, pages 11143–11152. IEEE, June 2022. 2

[36] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. Generating 3D faces using convolutional mesh autoencoders. In *European Conference on Computer Vision*, pages 725–741, 2018. 6, 7

[37] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzmán López, Nicolas Collignon, and Rik Sarkar. Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, page 4564–4573, New York, NY, USA, 2021. Association for Computing Machinery. 7

[38] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018. 2

[39] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in Neural Information Processing Systems*, 30, 2017. 3, 8

[40] Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, pages 9377–9388. PMLR, 2021. 3

[41] Chao Shang, Jie Chen, and Jinbo Bi. Discrete graph structure learning for forecasting multiple time series. In *International Conference on Learning Representations*, 2021. 8

[42] Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 1567–1577, New York, NY, USA, 2022. Association for Computing Machinery. 8

[43] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1548–1554. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track. 1, 2

[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. 1, 2, 4, 6, 7

[45] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph CNN for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 1, 2, 4, 6, 7

[46] Yuankai Wu and Huachun Tan. Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *arXiv preprint 1612.01022*, 2016. 2

[47] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–21, 2020. 2

[48] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 753–763, 2020. 8

[49] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, page 1907–1913. AAAI Press, 2019. 8

[50] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018. 1, 3, 7

[51] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International*

Conference on Learning Representations, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019. 1, 2, 6, 7

[52] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018. 2

[53] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *International Conference on Computer Vision*, pages 16259–16268. IEEE, 2021. 2

[54] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-GCN: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858, 2019. 2, 7

[55] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 1234–1241, 2020. 8

[56] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017. 3