UNDERSTANDING ATTENTION IN GRAPH NEURAL NETWORKS

Boris Knyazev [1,2] **& Graham W. Taylor**[1,2,3]
[1]University of Guelph, Canada
[2]Vector Institute for Artificial Intelligence, Canada
[3]Canada CIFAR AI Chair
{bknyazev,gwtaylor}@uoguelph.ca

Mohamed R. Amer
SRI International
201 Washington Rd
Princeton, NJ08540, USA
mohamed.amer@sri.com

ABSTRACT

We aim to better understand attention over nodes in graph neural networks and identify factors influencing its effectiveness. Motivated by insights from the work on Graph Isomorphism Networks (Xu et al., 2019), we design simple graph reasoning tasks that allow us to study attention in a controlled environment. We find that under typical conditions the effect of attention is negligible or even harmful, but under certain conditions it provides an exceptional gain in performance of more than 40% in some of our classification tasks. However, we have yet to satisfy these conditions in practice.

1 ATTENTION MEETS POOLING IN GRAPH NEURAL NETWORKS

The practical importance of attention in deep learning is well-established and there are many arguments in its favor (Vaswani et al., 2017), including interpretability (Park et al., 2016; Deac et al., 2018). In graph neural networks (GNNs), attention can be defined over edges (Velickovic et al., 2018; Zhang et al., 2018) or over nodes (Lee et al., 2018a). In this work, we focus on the latter, because, despite being equally important in certain tasks, it is not thoroughly studied in previous work (Lee et al., 2018b). To start our description, we first establish a connection between attention and pooling methods. In convolutional neural networks (CNNs), pooling methods are generally based on uniformly dividing the regular grid (such as one-dimensional temporal grid in audio) into local regions and taking a single value from that region (average, weighted average, max, stochastic, etc.), while attention in CNNs is typically a separate mechanism that weights input $X \in \mathbb{R}^{N \times C}$:

$$Z = \alpha \odot X, \tag{1}$$

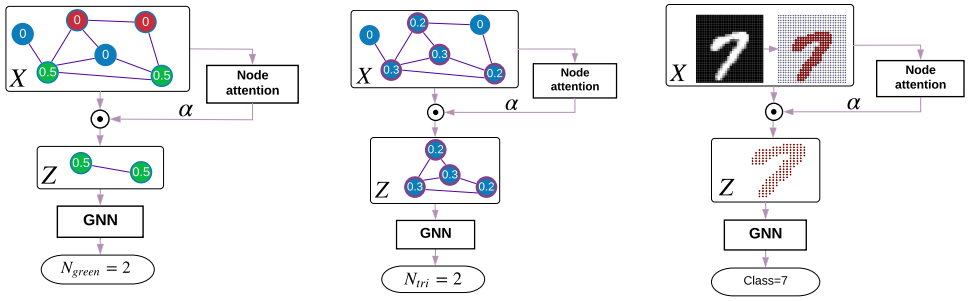
where $Z_i = \alpha_i X_i$ - output for unit (node in a graph) i, $\sum_i^N \alpha_i = 1$, $\odot$ - element-wise multiplication, N - the number of units in the input (i.e. number of nodes in a graph), C - its dimensionality.

In GNNs, pooling methods generally follow the same pattern as in CNNs, but the pooling regions (sets of nodes) are found based on clustering (Defferrard et al., 2016; Ying et al., 2018), since there is no grid that can be uniformly divided into regions in the same way across all examples (graphs) in the dataset. Recently, top-k pooling (Gao & Ji, 2018) was proposed, diverging from other methods: instead of clustering "similar" nodes, it propagates only part of the input and this part is not uniformly sampled from the input. Top-k pooling can thus select some local part of the input graph, completely ignoring the rest. For this reason at first glance it does not appear to be logical. However, we can notice that pooled feature maps in Gao & Ji (2018, Eq. 2) are computed in the same way as attention outputs Z in Eq. 1 above, if we rewrite their Eq. 2 in the following way:

$$Z_i = \begin{cases} \alpha_i X_i, & \forall i \in P \\ \emptyset, & \text{otherwise} \end{cases} \tag{2}$$

where P is a set of indices of pooled nodes, $|P| \leq N$, and $\emptyset$ denotes the unit is absent in the output.

The only difference between Eq. 2 and Eq. 1 is that $Z \in \mathbb{R}^{|P| \times C}$, i.e. the number of units in the output is smaller or, formally, there exists a ratio $r = |P|/N \leq 1$ of preserved nodes. Accordingly, we integrate attention and pooling into a single computational block of a GNN. In contrast, in CNNs,



(a) COLORS (b) TRIANGLES (c) MNIST

Figure 1: Three tasks we consider in this work. The values inside the nodes are ground truth attention coefficients, $\alpha_i^{(GT)}$, which we find heuristically as described in the Appendix.

it is challenging to achieve this, because the input is defined on a regular grid, so we need to keep resolution the same for all examples in the dataset after each pooling layer. In GNNs, we can remove any number of nodes, so that the next layer will receive a smaller graph. When applied on the input layer, this form of attention-based pooling also brings us interpretability of predictions, since the network makes a decision only based on pooled nodes.

Despite the appealing nature of attention, it is often unstable to train and conditions under which it fails or succeedes are unclear. Motivated by insights of Xu et al. (2019) recently proposed Graph Isomorphism Networks (GIN), we design two simple graph reasoning tasks that allow us to study attention in a controlled environment where we know ground truth attention. The first task is counting colors in a graph (COLORS), where a color is a unique discrete feature. The second task is counting the number of triangles in a graph (TRIANGLES). Finally, we confirm our observations on a standrad benchmark, MNIST (LeCun et al., 1998) (Figure 1).

2 MODEL

We study two variants of GNNs: Graph Convolutional Networks (GCN) (Kipf & Welling, 2016) and Graph Isomorphism Networks (GIN) (Xu et al., 2019). One of the main ideas of GIN is to replace the MEAN aggregator over nodes, such as the one in GCN, with a SUM aggregator, and add more fully connected layers after aggregating neigboring node features. The resulting model can distinguish a wider range of graph structures than previous models (Xu et al., 2019, Figure 3).

2.1 THRESHOLDING BY ATTENTION COEFFICIENTS

To pool the nodes in a graph using the method from Gao & Ji (2018) a predefined ratio $r = |P|/N$ (Eq. 2) must be chosen for the entire dataset. For instance, for $r = 0.8$ only 80% of nodes are left after each pooling layer. From our illustration in Figure 1, it is clear that this ratio should be different from graph to graph. Therefore, we propose choosing a threshold value $\tilde{\alpha}$ such that only nodes with attention coefficients $\alpha_i > \tilde{\alpha}$ will be propagated:

$$Z_i = \begin{cases} \alpha_i X_i, & \forall i : \alpha_i > \tilde{\alpha} \\ \emptyset, & \text{otherwise} \end{cases} \tag{3}$$

2.2 ATTENTION SUBNETWORK

To train an attention model that predicts the coefficients for nodes, we consider two approaches: (1) Gao & Ji (2018), where a single layer projection $\mathbf{p} \in \mathbb{R}^C$ is trained: $\alpha_{pre} = X\mathbf{p}$; (2) DiffPool (Ying et al., 2018), where a separate GNN is trained:

$$\alpha_{pre} = \text{GNN}(A, X), \tag{4}$$

where A - adjacency matrix of a graph. In all cases, we use a softmax activation (Vaswani et al., 2017; Park et al., 2016) instead of tanh in Gao & Ji (2018), because it provides more interpretable results and ecourages sparse outputs: $\alpha = \text{softmax}(\alpha_{pre})$. To train attention in a supervised way, we use the Kullback-Leibler divergence loss (see details in the Appendix).

2.3 CHEBYGIN

In some of our experiments, the performance of both GCNs and GINs is quite poor and, consequently, it is also hard for the attention subnetwork to learn. By combining GIN with ChebyNet (Defferrard et al., 2016), we propose a stronger model, ChebyGIN. ChebyNet is a multiscale extension of GCN (Kipf & Welling, 2016), so that for the first scale, $K = 1$, node features are node features themselves, for $K = 2$ features are averaged over one-hop neighbors, for $K = 3$ - over two-hop neighbors and so forth. To implement the SUM aggregator in ChebyGIN, we multiply features by node degrees starting from $K = 2$. We also add more fully connected layers as in GIN.

3 EXPERIMENTS

We introduce the color counting task (COLORS) and the triangle counting task (TRIANGLES) in which we generate synthetic training and test graphs. We also experiment with MNIST images (LeCun et al., 1998) (Figure 1). In all tasks, we assume to know ground truth attention, i.e. for each node i we heuristically define its importance in solving the task correctly, $\alpha_i^{(GT)} \in [0, 1]$, which is necessary to train (in the supervised case) and evaluate our attention models. See detailed description of tasks and model hyperparameters in the Appendix.

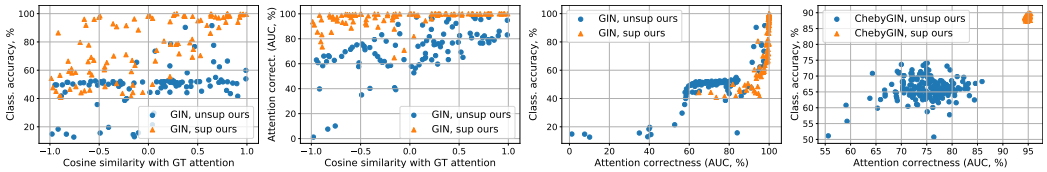
Generalization to larger graphs. One of the core strengths of attention is that it makes it easier to generalize to unseen, potentially more complex, inputs by reducing their complexity to similar inputs in the training set. To examine this phenomenon, for COLORS and TRIANGLES tasks we add test graphs that can be several times larger than the training ones and in Table 1 we report results on two test subsets. In all other experiments we report an average accuracy on the combined test set.
Attention correctness. We evaluate attention correctness using an area under the ROC curve (AUC) as an alternative to other methods, such as Liu et al. (2017), which can be overoptimistic in some extreme cases, such as when all attention is concentrated in a single node or attention is uniformly spread over all nodes. AUC allows to evaluate ranking of α instead of their absolute values. To evaluate attention correctness of models with global pooling, after training them we removed each of the nodes and compared the changes in predictions (Zeiler & Fergus, 2014). While this method shows surprisingly good results in some tasks, it is not built-in in training and thus only implicitly interprets a model's prediction (see Figure 5 in the Appendix for examples).

Table 1: **Results on three different tasks.** $Acc_{N \leq 25}$ denotes accuracy on the graphs with number of nodes ≤ 25, i.e. same as in the training set. $Acc_{N>25}$ - accuracy on large graphs. $\pm$ denotes we run experiments 10 times and report mean and standard deviation (high std is explained in Section 3.1). The best result in each column is boldfaced.

		COLORS			TRIANGLES			MNIST	
		$Acc_{N \leq 25}$	$Acc_{N>25}$	Attn AUC	$Acc_{N \leq 25}$	$Acc_{N>25}$	Attn AUC	Acc	Attn AUC
Baselines	GCN, global pool	93.3±1	45.3±25	99.9±0	46.0±1	23.2±1	79.2±0	84.32	69.2
	GIN, global pool	94.7±2	8.5±11	98.0±0	49.9±1	22.1±1	77.0±0	95.78	81.6
	GIN, unsup top-k	96.1±2	7.1±3	71.5±13	47.0±2	18.3±1	51.7±6	98.33	66.9
	ChebyGIN, global pool	**99.9±0**	44.0±24	**100.0±0**	66.1±1	29.5±1	79.3±0	98.75	76.4
	ChebyGIN, unsup top-k	98.8±2	12.5±10	76.3±20	64.0±5	25.2±2	75.8±6	98.60	75.8
Ours	GIN, unsup	86.4±20	11.4±17	71.3±18	47.8±2	19.7±2	61.6±7	97.95	73.4
	GIN, sup	89.2±10	**64.7±34**	96.5±7	48.9±1	22.2±1	75.9±1	98.44	96.5
	ChebyGIN, unsup	96.2±14	14.4±20	67.6±22	67.9±3	25.2±2	75.7±4	98.63	73.9
	ChebyGIN, sup	97.2±10	58.6±29	96.0±7	**88.5±1**	**48.7±1**	**94.8±0**	**99.03**	**97.9**
Upper bound	GIN + GT attn	100	100	100	86.0	65.8	100	98.09	100
	ChebyGIN + GT attn				99.6	100	100	98.92	100

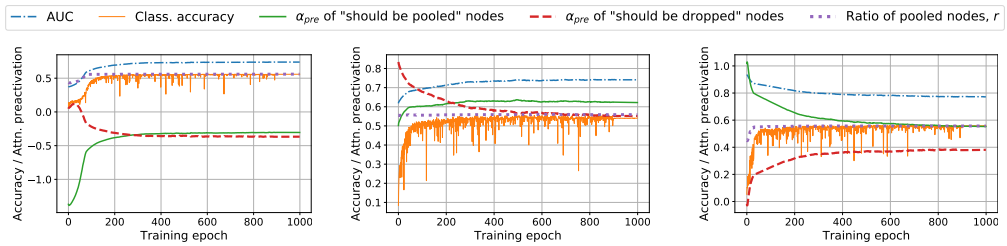
3.1 DISCUSSION OF RESULTS

GNNs operating in the spatial domain were designed to learn from graphs of arbitrary shape (Bronstein et al., 2017), so we expect them to be insensitive to graph size, N, both during training and testing. Our results in Table 1 (compare columns $Acc_{N \leq 25}$ and $Acc_{N>25}$) suggest that this assumption is only partially valid and GNNs generalize poorly to unseen graphs of larger size, which remains an open issue that can be solved by attention as discussed further. Our results in Table 1 confirm that GIN can better distinguish certain graph structures due to its injective SUM aggregator and more fully connected layers, however, it comes at significant downfall in generalization performance, i.e. on larger graphs in the test set. One possible explanation is that the SUM can quickly



Corr. coef.: (a) unsup: 0.40, sup: 0.67 (b) unsup: 0.60, sup: 0.42 (c) unsup: 0.79, sup: 0.70 (d) unsup: 0.12, sup: 0.28

Figure 2: Disentangling factors influencing classification accuracy for COLORS (a)-(c) and TRIANGLES (d). Notice the exponential growth of class. acc. depending on attention correctness (c).



(a) bad initialization (b) good initialization (c) optimal initialization

Figure 3: Influence of initialization on training dynamics for COLORS using GIN trained in an unsupervised way with an optimal threshold $\tilde{\alpha}$.

explode for large graphs thereby confusing the model. The MEAN aggregator in GCN is less sensitive to graph size and shows better performance on larger graphs, although is weaker on the test subset with small graphs. Thus, developing a neighborhood aggregator that is best in both test subsets is challenging. One way to alleviate this issue both for GCNs and GINs can be attention, which reduces the complexity of unseen samples to samples already seen in training.

However, our results show that training attention in an unsupervised way - a widely accepted approach - does not yield expected results. Moreover, we experience a high variation on the COLORS dataset. There are several reasons for that. First, we show that initialization of the attention model is critical (Figure 2). We ran 100 experiments with random initializations of the projection vector $\mathbf{p}$ and measured how performance of both attention and classification is affected depending on how close (in terms of cosine similarity) the initialized vector was to the optimal one, $\mathbf{p} = [0, 1, 0]$. While the direct relationship between the classification accuracy and cosine similarity is noisy (Figure 2, (a)), disentangling this dependency into two functions makes the relationship clearer (Figure 2, (b, c)). We first observe that initialization has a strong impact on attention, especially in the unsupervised case. Then, interestingly, we found that classification accuracy depends *exponentially* on attention correctness and becomes close to 100% only when attention is also very close to being perfect. In case of slightly worse attention, even starting from 99%, classification performance drops significantly. This is an important finding that can also be valid for other more realistic applications. In the TRIANGLES task we only partially confirm this finding, because our attention models could not achieve performance high enough to boost classification. However, by observing upper bound results obtained by training with ground truth attention, we assume that this boost potentially should happen once attention becomes accurate enough. But, once the attention model is trained on high dimensional inputs, it is very unlikely to initialize it in a way close to optimal (Figure 2, (d)), thereby making the effect of the attention model neglible or even harmful (see GIN results in Table 1)

Training attention in a supervised way only partially solves the initialization problem. In case of a good enough initialization, the network quickly converges to very good results. But in case of a bad initialization, it takes a very long time to reach similar performance (Figure 4 in the Appendix). In the unsupervised case, if the initialization is bad, the attention model is stuck in a suboptimal state (Figure 3, (a)) and even if it recovers after 200 epochs, it never reaches the performance of better initialized attention (Figure 3, (b,c)).

4 CONCLUSION

We show that learned attention can be extremely powerful in graph neural networks, but only if it is close to optimal. This is difficult to achieve due to the sensitivity of initialization, especially in the unsupervised setting where we do not know ground truth attention. Thus, we have identified initialization of attention models for high dimensional inputs as an important open issue.

ACKNOWLEDGMENTS

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. The authors also acknowledge support from the Canadian Institute for Advanced Research and the Canada Foundation for Innovation. We are also thankful to useful feedback from Angus Galloway.

REFERENCES

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Andreea Deac, Petar VeliČković, and Pietro Sormanni. Attentive cross-modal paratope prediction. *Journal of Computational Biology*, 2018.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

Hongyang Gao and Shuiwang Ji. Graph U-Net. In *Submitted to the Seventh International Conference on Learning Representations (ICLR)*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

John Boaz Lee, Ryan Rossi, and Xiangnan Kong. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1666–1674. ACM, 2018a.

John Boaz Lee, Ryan A Rossi, Sungchul Kim, Nesreen K Ahmed, and Eunyee Koh. Attention models in graphs: A survey. *arXiv preprint arXiv:1807.07984*, 2018b.

Chenxi Liu, Junhua Mao, Fei Sha, and Alan L Yuille. Attention correctness in neural image captioning. *arXiv preprint arXiv:1605.09553*, 2017.

Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. Attentive explanations: Justifying decisions and pointing to the evidence. *arXiv preprint arXiv:1612.04757*, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pp. 4805–4815, 2018.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*, 2018.

APPENDIX

4.1 DATASETS

COLORS. We introduce the color counting task. We generate random graphs where features for each node are set to one of the three one-hot values (colors): [1,0,0] (red), [0,1,0] (green), [0,0,1] (blue). The task is to count the number of green nodes, N_{green}. This is a trivial task, but it lets us study the influence of initialization of the attention model $\mathbf{p} \in \mathbb{R}^3$ on the training dynamics. In this task, graph structure is unimportant and edges of graphs act like a source of noise. Ground truth attention is $\alpha_i^{(GT)} = 1/N_{green}$, where i - indices of green nodes and $\alpha_i^{(GT)} = 0$ otherwise.

TRIANGLES. Counting the number of triangles in a graph is a well-known task which can be solved analytically by computing trace$(A^3)/6$, where A - adjacency matrix. This task turned out to be hard for GNNs, so we add node degree features as one-hot vectors to all graphs, so that the model can exploit both graph structure and features. Compared to the COLORS task, here it is more challenging to study the effect of initializing $\mathbf{p}$, but we can still calculate ground truth attention as $\alpha_i^{(GT)} = T_i / \sum_i T_i$, where T_i is the number of triangles that include node i, so that $\alpha_i^{(GT)} = 0$ for nodes that are not part of triangles.

MNIST(LeCun et al., 1998). MNIST contains 70k grayscale images of size 28×28 pixels. We represent each pixel as a node with features being pixel intensity values and coordinates. Hence we have $N = 784$ nodes in each graph. Edges are formed based on spatial distance between pixels as in Defferrard et al. (2016, Eq. 8). Each image depicts a handwritten digit from 0 to 9 and the task is to classify the image. Ground truth attention is considered to be $\alpha_i^{(GT)} = 1/N_{nonzero}$, where i - indices of pixels with nonzero intensity, $N_{nonzero}$ - total number of such pixels. The idea is that only nonzero pixels are important to determine the digit class. More sophisticated strategies to define $\alpha_i^{(GT)}$ can be applied.

4.2 NETWORK ARCHITECTURES AND TRAINING.

We build 2 layer GNNs for COLORS and 3 layer GNNs for other tasks with 64 filters in each layer (except for MNIST where we have more filters). We train them with Adam (Kingma & Ba, 2014), learning rate 0.001, batch size 32 (see Table 2 for details). For COLORS and TRIANGLES we minimize the regression (MSE) loss and cross entropy for MNIST. For experiments with supervised attention, we additionally minimize the Kullback-Leibler divergence loss between ground truth attention $\alpha^{(GT)}$ and predicted coefficients α:

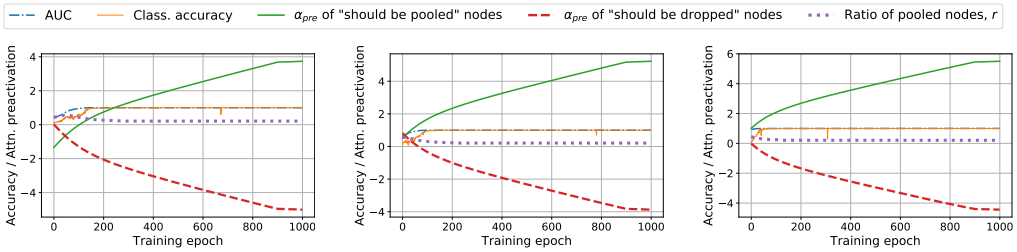
$$D = \frac{1}{N} \sum_i \alpha_i^{(GT)} \log\left(\frac{\alpha_i^{(GT)}}{\alpha_i}\right). \tag{5}$$

Table 2: Dataset statistics and model hyperparameters.

	COLORS	TRIANGLES	MNIST
# train graphs	500	30,000	60,000
# test graphs	$N \leq 25$: 2,500 $N > 25$: 2,500	$N \leq 25$: 5,000 $N > 25$: 5,000	10,000
# classes	10	10	10
# nodes (N) train	4-25	4-25	784
# nodes (N) test	4-200	4-100	784
# layers and filters	2 layers, 64 filters in each	3 layers, 64 filters in each	3 layers: 32, 64, 512 filters
Dropout	0	0	0.5
Nonlinearity	ReLU	ReLU	ReLU
# pooling layers	1	2	2
READOUT layer	global sum	global max	global max
GIN aggregator	SUM MLP with 256 hidden units	SUM MLP with 64 hidden units	SUM MLP with 64 hidden units
ChebyGIN # scales, K	2	7	2
ChebyGIN aggregator	SUM MLP with 256 hidden units	SUM MLP with 64 hidden units	SUM MLP with 64 hidden units
Attention model	$\mathbf{p}$ applied to input layer	Same arch. as the class. GNN, but $K = 2$ for ChebyGIN, applied to hidden layer (Eq. 4)	$\mathbf{p}$ applied to input layer[1]
Optimal threshold, $\tilde{\alpha}$	0.03	0.001	0.0001
Training params	50 epochs (lr decay after 35, 45)[2] Atnn. models: 100 epochs (lr decay after 85, 95)	100 epochs (lr decay after 85, 95)	30 epochs (lr decay after 20, 25)

[1]On MNIST, we found that training GNN for the attention model (Eq. 4) is unstable and often does not converge, so we replaced it with the trainable projection vector $\mathbf{p}$.

[2]Fewer than for attention models due to severe overfitting.



(a) bad initialization (b) good initialization (c) optimal initialization

Figure 4: Influence of initialization on training dynamics for the COLORS dataset using the GIN model trained in a supervised way with an optimal threshold $\tilde{\alpha}$. In case of a bad initialization, it still converges to perfect accuracy, but it takes longer compared to good and optimal initializations. In case of unsupervised learning of attention, the model can be stuck in a suboptimal state if it was initialized badly (Figure 3, (a)), which remains an open issue and requires further investigation.

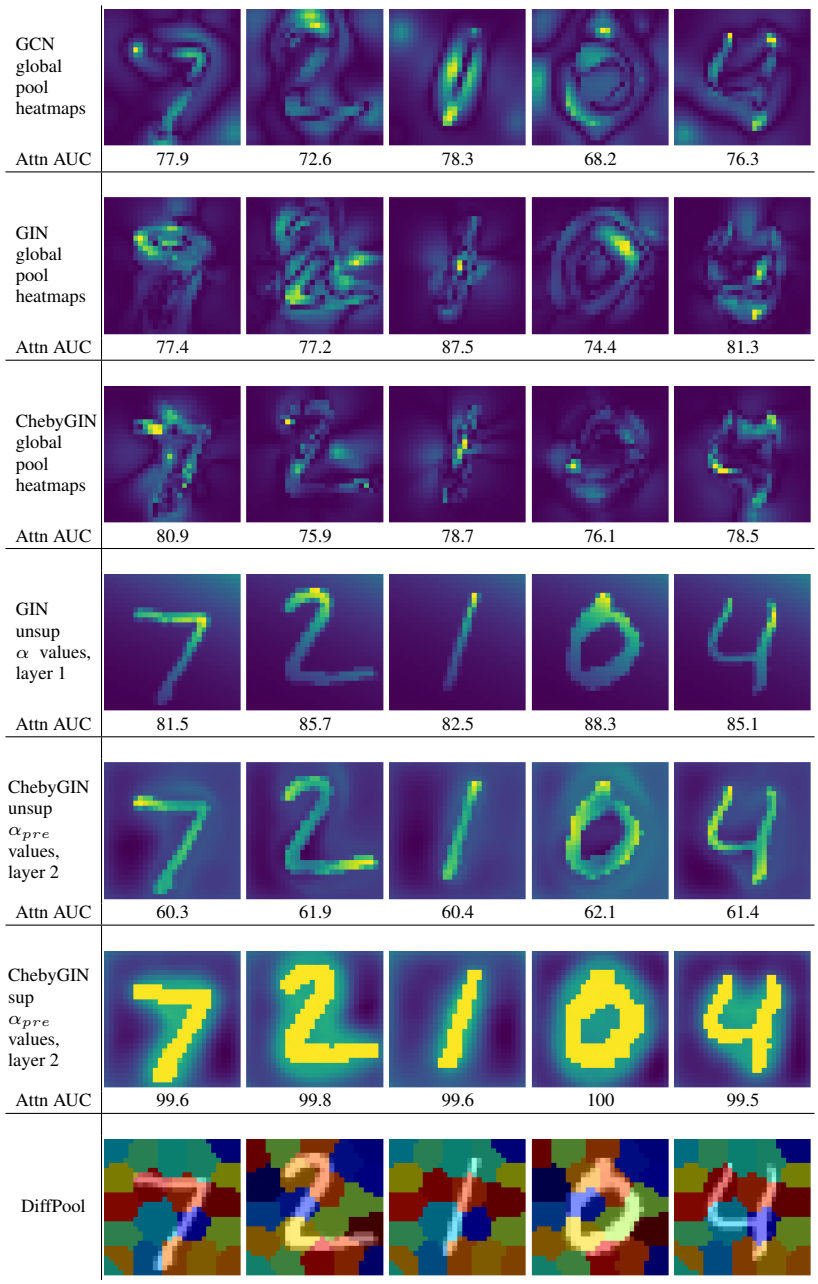


Figure 5: Visualizations of attention coefficients predicted by different models on five test images of MNIST. Attention correctness (AUC, %) for each image is shown under each image. We also trained a DiffPool (Ying et al., 2018) (bottom row) model to show the key differences between attention-based and clustering-based pooling. We believe both methods are strong and interpretable depending on the task. In the tasks we considered, attention based pooling is more effective.