Improving Expressivity of Graph Neural Networks

Stanisław J. Purgał
University of Innsbruck
Innsbruck, Austria
stanislaw.purgal@uibk.ac.at

***Abstract*—We propose a Graph Neural Network with greater expressive power than commonly used GNNs — not constrained to only differentiate between graphs that Weisfeiler–Lehman test recognizes to be non-isomorphic. We use a graph attention network with expanding attention window that aggregates information from nodes exponentially far away. We also use partially random initial embeddings, allowing differentiation between nodes that would otherwise look the same. This could cause problem with a traditional dropout mechanism, therefore we use a "head dropout", randomly ignoring some attention heads rather than some dimensions of the embedding.**

***Index Terms*—Graph Neural Networks, Graph Attention Networks, Deep Learning**

I. Introduction

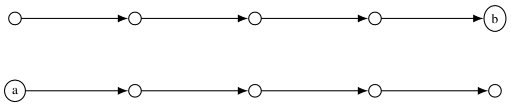
Recently there has been a great interest in neural network architectures capable of processing graphs [1]–[5]. They are applied for tasks of molecule properties prediction [6], premise selection in theorem proving [7], RNA sequence classification [8] etc.

Most Graph Neural Networks (GNNs) can recognize graphs only up to Weisfeiler–Lehman isomorphism test (WL-test) [9], [10], meaning that if the test says the graphs are isomorphic, the networks will process the graphs as if they were exactly the same — even if they are not.

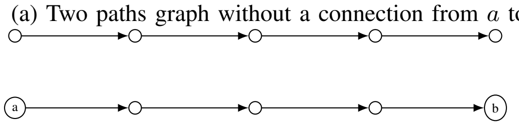
In our work we seek to overcome two types of failure of the WL-test. First is when the difference between graphs is only noticeable when considering long connections (eg. as in fig. 1). Another failure that we correct for is when we need to notice whether two indirect connections lead to one and the same node or to two similar nodes (as in fig. 2).

The first failure is addressed in our proposed model by aggregating nodes with an exponentially expanding window. This way we allow the network to notice a connection of exponential length. This operation could be seen as an attempt to imitate operations done in usual convolutions, such as pooling done in computer vision, which also aggregates information from exponentially far away, although in a more structured way. Another operation we can be said to imitate is an expanding dilated convolution used in WaveNet [11], which again aggregates information from far away. Both those approaches use intrinsic structure of the data to aggregate more information layer by layer rather than trying to process larger and larger sets. Unfortunately, in general, there is no such structure in graphs.

The second problem of the WL-test is solved by introducing a random identifier for every node present in the graph. This preserves invariance under node permutation while allowing



(a) Two paths graph without a connection from a to b



(b) Two paths graph with a connection from a to b

Fig. 1: Graphs consisting of two paths

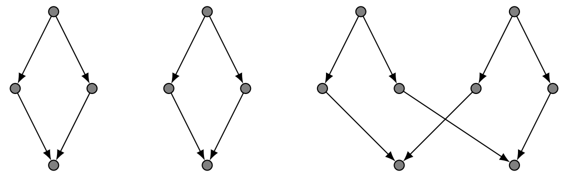


Fig. 2: "Diamond" graphs that common GNNs cannot differentiate

the network to differentiate between nodes even if they all look the same — thus allowing graph attention to be used even when no labels are present.

II. Preliminaries

We assume the reader to be familiar with self-attention mechanism [12] and its use in graph attention networks [13].

Graphs considered in this work are directed, with labelled nodes and edges. We allow all labels in a graph to be equal. Where we consider symmetric graphs, we model it with directed graphs where for every edge there exists a symmetric edge in the other direction. We do not consider multi-edges, though technically our model allows for edges with multiple labels.

When presenting formulas for calculations done in our model we mark parts with learnable parameters with subscript ϕ. We use $||$ to mark concatenation and $\odot$ to mark point-wise multiplication (or Hadamard product).

III. Proposed model

Our proposed model modifies standard graph attention [13] in three ways:

- random initial node embeddings — to facilitate attention mechanism recognizing different nodes, we add (by concatenation) a random vector to initial embedding of every node, with different random values every time the embeddings are evaluated.

- expanding attention window — we use multi-headed attention [12], with separate heads for different edge categories. Some of attention heads only see neighbours (as is standard), but some see exponentially expanding neighbourhoods (nodes in distance 2, 4 and so on).

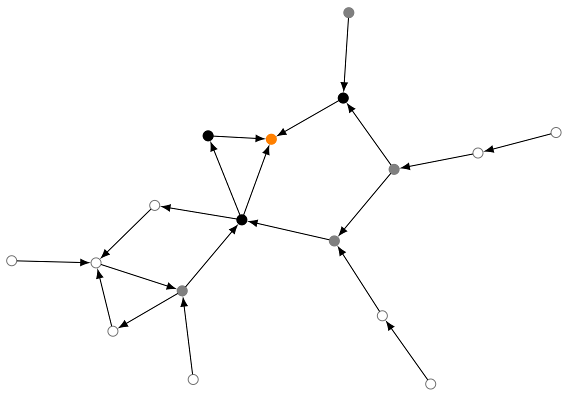
A. Partially random initial node embeddings

In our model (expGNN), initial embedding of a node is composed of two concatenated components of same length. One is a learnable embedding of a node label, the other a *random node identifier*, a random vector composed of 1s and 0s (each possible with probability $\frac{1}{2}$).

$$\mathbf{n}^0_{i\phi} = \textsc{Embed}_\phi(\textsc{Label}(n_i)) \,||\, \textsc{RandomSequence}(\{0 : \frac{1}{2}, 1 : \frac{1}{2}\}))$$

This identifier is different every time an embedding is being calculated, but stays the same withing one graph instance. This means that it is possible to differentiate between nodes, even if their label and neighbourhoods are the same.

B. Expanding attention window



To facilitate propagation of information within a graph (faster than one edge per one layer), we propose an *expanding attention window*. In each layer this window expands exponentially, aggregating information from nodes further away. So, in layer n we aggregate nodes that are within distance 2^n.

C. Multiple attention filters

Since it is not clear that this expanding window would be helpful for every task, we use different windows for different attention heads, with some aggregating only neighbours, some using this expanding window, and some aggregating from all nodes in the graph. Since we want information to spread both ways, not only in the direction of edges, we also use different heads where edges go in opposite direction.

All attention head types used in our model are:

- Neighbouring nodes (different edge types separately)
- Reversed neighbouring nodes (all edge types together)
- Expanding window (all edge types together)
- Reversed expanding window (all edge types together)
- All nodes in the graph

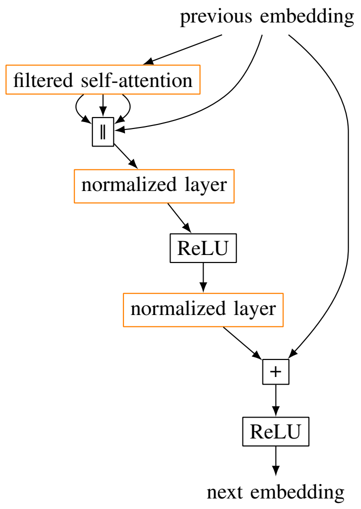
When working with an adjacency matrix, expanding the window can be done quite efficiently, by calculating a new adjacency matrix:

$$A_{n+1} = \min(1, A_n \cdot A_n + A_n)$$

Of course, when working with more optimized graph representations for sparse graphs, this operation is very costly, as it makes the graph much denser.

D. Single layer architecture

For a single layer in our model we use residual connection [14], similar to that used in Transformer [12], but also utilizing layer normalization [15].



$$\mathbf{n}^{n+1}_{i\phi} = \textsc{ReLU}(\mathbf{n}^n_{i\phi} + \textsc{FNN}_\phi(\mathbf{n}^n_{i\phi} || \textsc{FilteredMultiHead}^n_\phi(\mathbf{n}^n_{i\phi}, \mathbf{n}^n_{*\phi}, \mathbf{n}^n_{*\phi})))$$

$$\textsc{FNN}_\phi(x) = \textsc{NormalizedLayer}_\phi(\textsc{ReLU}(\textsc{NormalizedLayer}_\phi(x)))$$

Multi-headed dot-product attention works as in [12], only difference being using different masks for different heads.

$$\begin{aligned}\textsc{Attention}_\phi(Q, K, V, M) &= \alpha V W^V_\phi \\ \alpha &= \textsc{maskedSoftmax}(M, \beta) \\ \beta &= \frac{(QW^Q_\phi)(KW^K_\phi)^T}{\sqrt{d_k}} \\ \textsc{maskedSoftmax}(M, x) &= \frac{\exp x \odot M}{\sum \exp x \odot M}\end{aligned}$$

$$\begin{aligned}\textsc{FilteredMultiHead}^n_\phi(Q, K, V) = \textsc{concat}(&\\ &\textsc{Attention}_\phi(Q, K, V, M_1)\\ &\vdots\\ &\textsc{Attention}_\phi(Q, K, V, M_h))\end{aligned}$$

Normalized layer is defined in [15] as:

$$\begin{aligned}\textsc{NormalizedLayer}_\phi(x) &= \frac{g_\phi}{\sigma} \odot (a - \mu) + b_\phi \\ a &= xA_\phi \\ \mu &= \frac{1}{H}\sum_{i=1}^{H} a_i \\ \sigma &= \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i - \mu)^2}\end{aligned}$$

E. Final aggregation for graph classification

The node embeddings resulting from a few layers described above (in our experiments 3) are aggregated from all nodes in the graph using simple *maximum*. The resulting graph embedding is fed to a two-layer feed-forward network.

F. Head dropout

The standard dropout [16] mechanism may conflict with the random initial embeddings. The network is supposed to rely on random distribution of vector representations of the nodes in the graph. Using dropout changes this distribution, making it different during training and during evaluation. This could (and a few times did during the experiments) lead to a situation where loss goes down while the accuracy remains poor.

To counteract this problem, and to force learning of different useful properties, we use a "head dropout". Instead of removing some parts of vectors, we randomly ignore certain attention heads. During training, each type of attention window (immediate neighbours, expanding, reversed etc.) is ignored with some probability (in our experiments 0.1).

IV. Experiments

We test ability of our model to recognize properties that theoretically require overcoming the limitation of WL-test. To do that, we generate artificial datasets, with graph labels determined by the tested property. To better validate generalizing ability, we use more than one evaluation set, with a few different methods of generating random graphs (but using the same property for labels).

For training datasets we use uniform random graphs, where every edge exists with the same probability. This probability is chosen to be such that about half of the generated graphs have the property being tested.

Size of training datasets is 10^6 (one million), and sizes of random testing datasets are all 10^4 (ten thousand). The synthetic datasets used are available online[1] in a format compatible with [17].

A. Presence of a cycle in a symmetric graph

We generate symmetric graphs with 32 nodes, and classify them by checking whether there is a cycle in the graph.

Evaluation sets include:

- more random graphs from the same distribution as the training set
- uniform random graphs with 64 nodes (with lower edge-existence probability) and with 16 nodes (with higher edge-existence probability)
- random trees
- random trees with one additional edge (creating a cycle)
- line graphs of length between 3 and 64
- cycles of length between 3 and 64

Random trees are generated by adding nodes one by one, attaching each one to a random already existing node. Half of such generated trees also receive one additional edge between a random pair of not connected nodes.

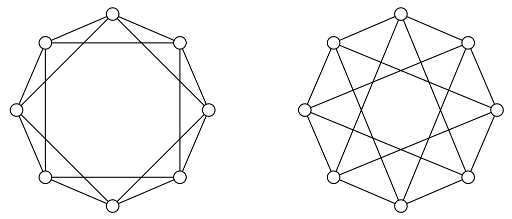


Fig. 3: Circulant skip links — $\mathcal{G}_{\text{skip}}(8,2)$ and $\mathcal{G}_{\text{skip}}(8,3)$

B. Presence of a clique 4

For training, again, we use random uniform graphs with 16 nodes. Evaluation sets include also bigger (and sparser) graphs than those used in training.

In each set the class on a graph depends on presence of a clique 4 (a subset of 4 nodes where is each node is connected to every other node).

C. Categorizing circulant skip links

We test our network on the dataset the most difficult dataset used in [18], [19]. This dataset has 10 categories, with only 1 graph each. Each graph is a $\mathcal{G}_{\text{skip}}(41, R)$ with R being one of $\{2, 3, 4, 5, 6, 9, 11, 12, 13, 16\}$. A graph $\mathcal{G}_{\text{skip}}(N, R)$ contains N nodes $\{1, ..., N\}$, such that a pair of nodes (a, b) is connected if (and only if) $|a - b| \equiv 1$ or $R \pmod N$ (see fig. 3 for an example).

In their tests [18], [19] use 15 randomly permuted instances of each graph (for a total of 150 graphs in the dataset). Since our network is invariant under permutations, that would be pointless here, and we only use 10 graphs. For evaluation however, since our model is non-deterministic, we do use 150 graphs to get a better evaluation of our accuracy.

Since no generalizing beyond the training set in necessary in this test, we do not use dropout here.

D. Presence of a path from one highlighted node to the other

As earlier, training set consists of uniformly random graphs, now with two nodes being given special labels (a and b). The class of a graph depends on the existence of a path from a to b. In the training set all graphs have 32 nodes.

As a special testing case we use graphs consisting of two paths. The highlighted nodes can be either on the ends on one path, or on two different paths (shown is figure 1). Those graphs are very similar, and hard for commonly used GNNs to differentiate between. We use paths of length from 2 to 32.

[1]http://cl-informatik.uibk.ac.at/cek/ijcnn2020/

TABLE I: Presence of a clique 4 results

model	accuracy			
	training	size 16	size 32	size 64
GFN	0.8076	0.8142	0.5068	0.5092
GCN	0.9005	0.9088	0.5118	0.4887
GraphStar	0.9983	**0.9772**	0.5331	0.5092
expGNN	0.9129	0.9108	0.7349	**0.5662**
expanding window only	0.5016	0.4924	0.4955	0.4908
random init only	0.9293	0.9279	**0.7401**	0.5427
basic graph attention	0.5016	0.4924	0.4955	0.4908

TABLE II: Presence of a cycle results

	accuracy						
	training	uniform 32	uniform 64	uniform 16	trees 64	trees 32	lines + cycles
GFN	0.9947	0.9961	0.9099	0.8737	0.7659	0.8141	0.0887
GCN	0.9995	**0.9996**	0.8914	0.8768	0.5649	0.8755	0.2419
GraphStar	0.9983	0.9729	**0.9994**	**1.0000**	0.8167	0.9005	0.1452
expGNN	0.9990	0.9993	0.9819	0.9999	**0.8227**	**0.9499**	0.5726
expanding window only	0.5294	0.5275	0.4424	0.5106	0.4951	0.4982	0.5000
random init only	0.9823	0.9836	0.9230	0.9972	0.7950	0.9226	**0.6129**
basic graph attention	0.5294	0.5275	0.4424	0.5106	0.4951	0.4982	0.5000

TABLE III: Circulant skip links results

model	accuracy			
	mean	std	max	min
RP-GIN [18]	0.376	0.129	0.533	0.100
16-CLIP [19]	0.908	0.068	0.987	0.760
Ring-GNN [20]	N/A	0.157	0.800	0.100
expGNN	**0.978**	0.015	**0.993**	**0.947**
expanding window only	0.100	0.000	0.100	0.100
random init only	0.687	0.030	0.740	0.647
basic graph attention	0.100	0.000	0.100	0.100

TABLE IV: Presence of node of degree 7 results

model	accuracy		
	training	size 16	size 32
GFN	1.0000	**1.0000**	**1.0000**
GCN	1.0000	**1.0000**	0.8300
GraphStar	1.0000	**1.0000**	**1.0000**
expGNN	0.9520	0.9534	0.5903
expanding window only	0.5863	0.5906	0.5385
random init only	0.9862	0.9873	0.6402
basic graph attention	0.5863	0.5906	0.5385

E. Presence of a node with 7 neighbours

In this dataset we simply generate uniform graphs and check whether there is a node with degree 7 or greater. For training we use graphs of size 16, for testing we use also bigger graphs of size 32.

F. Chemical datasets

We also test our model on a few chemical datasets from [17]. These were published on [21], collected from the PubChem website[2]. Each dataset belongs to a certain type of cancer screen with the outcome active or inactive.

[2]https://pubchem.ncbi.nlm.nih.gov/

G. Tested models

For comparison with our model we use several recently published graph neural architectures: Graph Feature Network [22], Graph Convolutional Network (using implementation from the same work [22]) and Graph Star Net [23].

The exception is the experiment with circulant skip list, where those networks mathematically can't differentiate between graphs. There we compare with results reported in other papers that also used this dataset [18]–[20]. Since [20] does not report mean of their results, we leave it as "N/A".

We also test variants of our model with only one of the two modifications, as well as without both (making it a Graph Attention Network [13]).

H. Hyperparameters

In our model we use 3 layers of graph message passing. In every layer each node is encoded in 128 dimensions. In dot-product attention the queries and keys have 32 dimensions. Each type of attention head is used thrice. During training each type has a 0.1 chance of being ignored. For optimization we use Adam optimizer [24] with default $\beta_1 = 0.9$, $\beta_2 = 0.999$, ϵ = 1e-7 and learning rate 1e-3.

V. Results and discussion

A. Presence of a clique and a cycle

These two observed graph properties are on one hand simple, on the other according to [10] cannot really be expressed by usual GNNs. Somewhat surprisingly, results in tables I and II show that GNNs still learn to recognize them with high accuracy given a graph of the same size as those in the training set. However, changing the size of the graph and the density of edges greatly lowers the accuracy, revealing that the learned property is not actually what we wanted.

Our proposed model seems to be able to generalize the property to graphs of different sizes much better.

TABLE V: Presence of a path results

model	training	accuracy			
		size 16	size 32	size 64	paths
GFN	0.8358	0.8453	0.8276	0.6775	0.5483
GCN	0.9706	0.7057	0.9696	0.6810	0.5161
GraphStar	0.9979	1.0000	0.9975	0.9925	0.5967
expGNN	1.0000	**1.0000**	**1.0000**	**1.0000**	0.8371
expanding window only	1.0000	**1.0000**	**1.0000**	0.9999	**0.8629**
random init only	0.9903	0.9984	0.9889	0.9853	0.5645
basic graph attention	0.9881	0.9977	0.9866	0.9859	0.5806

TABLE VI: Chemical datasets results

model	accuracy		
	SN12C	MOLT-4	Yeast
GFN	0.9639	0.9374	**0.8899**
GCN	0.9592	0.9336	0.8871
GraphStar	0.9640	**0.9394**	0.8884
expGNN	0.9633	0.9335	0.8870
expanding window only	**0.9656**	0.9365	0.8875
random init only	0.9626	0.9347	0.8865
basic graph attention	0.9648	0.9365	0.8870

B. Categorizing circulant skip links

The results in table III show that our model achieves better accuracy than reported in [18]–[20].

We see that categorizing long skip links is impossible in 3 layers when not using the expanding attention window. With it however, even 3 layers are enough.

We note that [20] also reports 100% accuracy with Ring-GNN-SVN, a variant of Ring-GNN that is given top eigenvalues of adjacency matrices, allowing for trivial classification.

C. Presence of a node with degree 7 and presence of a path

The last two graph properties are things that can be trivially learned by some GNNs. For most used models, the most basic property is the degree of a node (trivially extracted, or in GFN [22] just given as part of the initial embedding). In our model, learning to extract the degree a node is possible, but much harder (and, because it depends on random initial embeddings, remains not 100% accurate). Instead, the basic property is detecting a connection.

D. Chemical datasets

Experiments on chemical datasets (results shown in table VI) show that even though our model has higher theoretical expressive power, it does not improve accuracy on chemical benchmarks.

VI. Related work

This work seeks to improve Graph Neural Networks. The core idea of GNNs [25] is to generate new node embeddings by aggregating embeddings of neighbouring nodes. Initially, a recurrent networks would process the nodes until their embeddings converged to some value. Currently, most networks use some constant number of layers that aggregate nodes (as do we). GraphSAGE [26] experiments with aggregating embeddings using simple functions like mean and maximum.

Following spectral graph theory Kipf et al. [27] propose a Graph Convolution operator. It can be thought of as a sum aggregation, but with embeddings scaled by an inverse of a square root of a node degree ($\frac{1}{\sqrt{d_n}}$ or $d_n^{-\frac{1}{2}}$), both before and after aggregation. In [22] some features are added to the initial node embeddings.

Graph Attention Networks, a type of GNNs that we build on, were introduced in [13]. In this network attention mechanism [28] is used to aggregate the embeddings. Attention extracts information from a set of vectors (representations of things — in our case nodes) by first estimating importance of every element of the set and then calculating weighted average (with weights depending on importance). The importance calculation can be done using a smaller feedforward neural network that given the context and the element estimates importance of the element in the context, or (as in [12]) by calculating a dot-product of some projection of context representation with a projection of the element.

The attention mechanism allows for aggregating information of a *set*, rather than a *sequence* (that is, ignore ordering of elements), which fits exactly what we need in graph processing.

All of the above mentioned networks allow information to travel only one edge per network layer. To allow far information propagation a Graph Star Net was proposed [23], where a global state (a few ”star” nodes) is updated in every layer. This allows information to propagate globally and to neighbours, but not anything in-between. Thus, this network still suffers from the constraint of WL-test. Our model allows also far-but-not-global propagation, it is however much more computationally costly for large graphs.

A. On Graph Attention without node labels

The output of an attention mechanism with a multiset of exactly the same elements as input will always be equal to that one element. This is because the output of attention is essentially a weighted average, and if all elements are the same, the output will be the same regardless of what (and how many) the weights are.

Because of this, in a simple graph attention network, when all the nodes have the same embedding (eg. when no node labels are provided), they will remain the same after however many layers. The network can only differentiate between having any neighbours and having none.

The GraphStar network [23] gets around this problem by using attention across both neighbours and a few global stars.

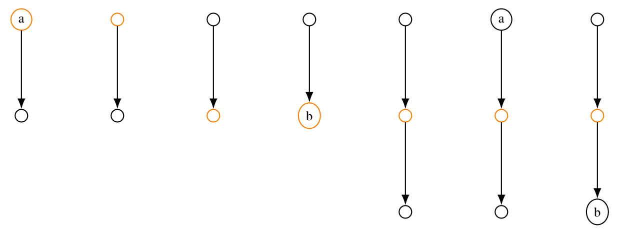


Fig. 4: All immediate neighbourhoods of a two paths graph (the same regardless of whether a and b are connected)

In this way, after one layer the embedding depends on the degree of a node (effectively nodes are given labels based on their degree).

Our proposed model uses random node identifiers, making a situation where all nodes have the same embeddings extremely unlikely (effectively impossible).

B. Perception limits of GNNs

Xu et al. [10] describe expressive power of message-passing GNNs as equivalent to Weisfeiler-Lehman isomorphism test [9]. This means that a node embedding can depend only on the node's subtree structure of certain depth (the depth being equal to the number of layers). The graph classification then depends on the multiset of subtree structures present in the graph. A network capable of distinguishing between all multisets of subtree structures (of certain depth) is referred to in [10] as a *maximally powerful GNN*. Yet even such networks cannot differentiate between graphs that WL-test deems isomorphic.

The work of [18] seeks to overcome this problem by using by using using a permutation sensitive aggregator and summing over all permutations. To make this computationally feasible, they propose k-ary Relational Pooling.

A different approach is proposed in [19] where they use one-hot encoded coloring of nodes added in way that allows for differentiating between nodes.

A mechanism similar to our expanding window was described in [29], where multiple powers of adjacency matrix were used during aggregation (in their experiments they were A^2 and A^4). Later in [20] they use a learnable mechanism to calculate consecutive powers of adjacency matrix, that can in particular learn to express the a property very similar to our exponentially expanding window (what the model in [20] can express is $\min(A^{2^n}, 1)$, a window containing all nodes that can be reached in *exactly* 2^n steps). In this variant, the "adjacency matrices" don't necessarily contain only 1s and 0s.

Our work seeks to expand the limit of WL-test in two places: for one, by utilizing expanding attention window we effectively increase depth of subtree structures exponentially. Since we also use direct neighbourhood in some attention heads we theoretically don't lose any expressive power. Another way our model is more expressive is its ability to recognize connection to one and the same node from a connection to two identical nodes. We achieve this by using random initial embeddings.

We should point out that because of use of randomness we lose the property of isomorphic graphs always having the same embedding — instead we have isomorphic graphs having the same *distribution* of embeddings.

VII. Conclusion

We present a Graph Neural Network with more expressive power than any model we have seen described. We show its ability to differentiate between graphs that other networks cannot. What our models seems to excel at is classifying synthetic datasets of graphs WL-test fails to recognize as non-isomorphic and generalization to previously unseen graph sizes (and edge densities).

Future work includes translating this improvement to accuracy on chemical datasets.

Acknowledgement

This research was supported by the ERC starting grant no. 714034 SMART.

References

[1] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 960–11 970.
[2] S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu, "Graph neural tangent kernel: Fusing graph neural networks with graph kernels," in *Advances in Neural Information Processing Systems*, 2019, pp. 5724–5734.
[3] W. Zhao, C. Xu, Z. Cui, T. Zhang, J. Jiang, Z. Zhang, and J. Yang, "When work matters: Transforming classical network structures to graph cnn," *arXiv preprint arXiv:1807.02653*, 2018.
[4] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 9240–9251.
[5] Q. Xuan, J. Wang, M. Zhao, J. Yuan, C. Fu, Z. Ruan, and G. Chen, "Subgraph networks with application to structural feature space expansion," *IEEE Transactions on Knowledge and Data Engineering*, 2019.
[6] C. Helma, R. King, S. Kramer, and A. Srinivasan, "The predictive toxicology challenge 2000-2001," *Bioinformatics*, vol. 17, 01 2001.
[7] M. Wang, Y. Tang, J. Wang, and J. Deng, "Premise selection for theorem proving by deep graph embedding," in *Advances in Neural Information Processing Systems*, 2017, pp. 2786–2796.
[8] E. Rossi, F. Monti, M. Bronstein, and P. Liò, "ncrna classification with graph convolutional networks," *arXiv preprint arXiv:1905.06515*, 2019.
[9] B. Weisfeiler and A. A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9, pp. 12–16, 1968.
[10] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.
[11] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.
[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.
[13] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.
[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[15] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel, and M. Neumann, "Benchmark data sets for graph kernels," 2016. [Online]. Available: http://graphkernels.cs.tu-dortmund.de

[18] R. L. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro, "Relational pooling for graph representations," *arXiv preprint arXiv:1903.02541*, 2019.

[19] G. Dasoulas, L. D. Santos, K. Scaman, and A. Virmaux, "Coloring graph neural networks for node disambiguation," *arXiv preprint arXiv:1912.06058*, 2019.

[20] Z. Chen, S. Villar, L. Chen, and J. Bruna, "On the equivalence between graph isomorphism testing and function approximation with gnns," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 868–15 876.

[21] X. Yan, "Graph datasets." [Online]. Available: https://sites.cs.ucsb.edu/~xyan/dataset.htm

[22] T. Chen, S. Bian, and Y. Sun, "Are powerful graph neural nets necessary? a dissection on graph classification," *arXiv preprint arXiv:1905.04579*, 2019.

[23] L. Haonan, S. H. Huang, T. Ye, and G. Xiuyan, "Graph star net for generalized multi-task learning," *arXiv preprint arXiv:1906.12330*, 2019.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[26] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[27] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[28] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[29] Z. Chen, X. Li, and J. Bruna, "Supervised community detection with line graph neural networks," *arXiv preprint arXiv:1705.08415*, 2017.