PiNet: Attention Pooling for Graph Classification

Peter Meltzer
Department of Computer Science
University College London
London, UK
p.meltzer@cs.ucl.ac.uk

Marcelo Daniel Gutierrez Mallea
Department of Computer Science
University College London
London, UK
marcelo.mallea.16@ucl.ac.uk

Peter J. Bentley
Department of Computer Science
University College London
London, UK
p.bentley@cs.ucl.ac.uk

Abstract

We propose PiNet, a generalised differentiable attention-based pooling mechanism for utilising graph convolution operations for graph level classification. We demonstrate high sample efficiency and superior performance over other graph neural networks in distinguishing isomorphic graph classes, as well as competitive results with state of the art methods on standard chemo-informatics datasets.

1 Introduction

Graph classification, the task of labeling each graph in a given set, has applications in many diverse domains ranging from chemo-informatics (1) and bio-informatics (2), to image classification (3) and cyber-security (4). In recent years, Convolutional Neural Networks (CNNs) have led the state of the art in many forms of pattern recognition, i.e. in images (5) and audio (6).

Essential to the success of CNNs in representation learning is the process of pooling (7), in which a set of related vectors are reduced to a single vector (or smaller set of vectors). An important property of a pooling operator is invariance to different orderings of the input vectors. In vertex level learning tasks such as link prediction and vertex classification, Graph Convolutional Networks (GCNs) achieve invariance by pooling neighbours' feature vectors with symmetric operators such as feature-weighted mean (8), max (9), and self-attention weighted means (10).

In this work we present PiNet[1], a differentiable pooling mechanism by which the vertex-level invariance to permutation achieved for vertex level tasks may be extended to the graph level. Inspired by the attention mechanisms of RNNs (11) and Graph Attention Networks (GAT) (10), we propose an attention-based aggregation method which weights the importance of each vertex in the final representation.

2 Related work

The idea of permutation invariant deep learning is not new. (12) consider the case of classification on sets, in which they propose that a permutation invariant function $f(\mathbb{X})$ on the set $\mathbb{X}$ may be learned

[1] Code available at http://github.com/meltzerpete/pinet

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

indirectly through decomposition in the form

$$f(\mathbb{X}) = \rho\left(\sum_{x\in\mathbb{X}} \phi(x)\right), \tag{1}$$

if suitable transformations ρ and ϕ can be found. This idea is specialized as Janossy Pooling in (13), where ρ is a normalisation function, and the summation occurs over the set of all possible permutations of the input set. They also propose the use of canonical input orderings and permutation sampling offering a trade-off between learnability and computational tractability.

The use of canonical orderings to tackle permutations in graph representation learning has been demonstrated to be effective in Patchy-SAN (14). Here canonical labellings are applied to provide an ordering over which nodes are sampled, aggregated and normalised to convert each graph to a fixed sized tensor which is then fed into a traditional CNN. DGCNN (15) also uses a sorting method to introduce permutation invariance, where vertex embeddings are first obtained with a GCN, and then sorted before being fed into a traditional CNN.

Considering the task of vertex classification, the GCN as introduced by (8) can in fact be formulated as a particular instance of Equation 1, where for each vertex i the output $\boldsymbol{x}_i^{(l+1)}$ of a single layer l with input features $\boldsymbol{x}^{(l)}$ is given by

$$\boldsymbol{x}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)\cup\{i\}} \frac{1}{c_{ij}} \boldsymbol{x}_j^{(l)} \boldsymbol{W}^{(l)}\right), \tag{2}$$

where σ is a non-linear activation function, $\mathcal{N}(i)$ is set of vertices in the immediate neighbourhood of vertex i, c_{ij} is a normalisation constant of edge (i, j), and $\boldsymbol{W}^{(l)}$ is the learned weights matrix for layer l. We also note that (8) and variants may also be expressed as an instance of the Weisfeiler-Lehmen graph isomorphism algorithm (16), thus providing the theoretical justification for which graph convolution operations are able to capture the structural information of graphs.

(10) extends (8) with the introduction of attention mechanisms, where a vertex's edges are weighted by a neural network with the vertex pair as input. Many other (in fact virtually all) variants of (8), i.e. (17; 9; 10; 18; 19; 20), may also be expressed as an instance of Equation 1, therefore indicating invariance to permutations at the vertex level (GraphSAGE with LSTM neighbourhood aggregator (9) is an example of one that is not). However, since the vertices have no natural ordering, the output matrix of a GCN is not inherently invariant to permutation and thus does not make a good graph representation.

A simple solution is to use a symmetric operator to combine vertex vectors to form a single graph vector, for example the mean. Again we can formulate this entire process as an instance of Equation 1, where ρ is the mean, and ϕ is the GCN's particular vertex function. A less naive method to aggregate GCN-learned vertex embeddings can be seen in DiffPool (21), where GCN-based vertex embeddings are used to cluster nodes to aggregate features hierarchically, thus considering the structural information of the graph as opposed to a flat, global aggregation. Other structural pooling methods include (22) which use attention-based guided walks to direct RNNs to select parts of the graph to inform the final representation.

3 PiNet

3.1 Model architecture

PiNet is a generalised end-to-end deep neural network architecture that utilizes the vertex-level permutation invariance of graph convolutions in order to learn graph representations that are also invariant to permutation.

Let $G = (\boldsymbol{A}, \boldsymbol{X})$ be a graph from a set $\mathcal{G}$ with adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ and vertex features matrix $\boldsymbol{X} \in \mathbb{R}^{N\times F}$, and $\psi : (\mathbb{R}^{N\times N}, \mathbb{R}^{N\times F}) \to \mathbb{R}^{N\times F'}$ be any message passing convolution network (i.e. the GCN (8)) (note ψ may contain an arbitrary number of layers). PiNet may then be defined by the output for a single graph,

$$z(G) = \sigma_S\Big[g\Big(\sigma_S\big([\psi_A(\boldsymbol{A}, \boldsymbol{X})]^\top\big) \cdot \psi_X(\boldsymbol{A}, \boldsymbol{X})\Big)\boldsymbol{W}_D\Big] \in \mathbb{R}^C, \tag{3}$$

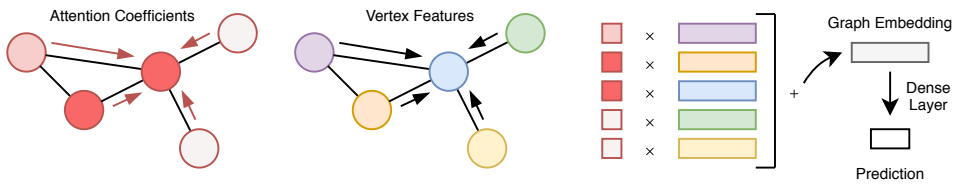

Figure 1: Overview of PiNet: One message passing network learns vertex features, the other learns attention coefficients. The final graph representation is a sum of the learned vertex features weighted by the attention coefficients. For multiple attention dimensions per vertex, the graph embedding becomes a matrix where the rows are concatenated to form a single vector.

where σ_s is the softmax activation function, g is a function that concatenates rows of a matrix to form a vector, ψ_A and ψ_X are separate message passing networks for learning attention coefficients and vertex features respectively, $\cdot$ is a matrix product, $\boldsymbol{W}_D$ is a weights matrix for a fully connected dense layer, and C is the number of target classes. The inner softmax constrains the attention coefficients to sum to 1 and prevents them from all falling to 0. The outer softmax may be replaced for multi-label classification tasks (i.e. sigmoid).

4 Experiments

All hyper-parameters are detailed in Appendix A.

4.1 Datasets

For the isomorphism test (4.2) we use a generated dataset available from our repository. The generation process is detailed in Appendix B. All other experiments are performed using a standard set of chemo-informatic benchmark datasets[2].

4.2 Experiment 1: Isomorphism test

For PiNet we use $\psi_A = \psi_X = \sigma_R(\tilde{\boldsymbol{A}} \cdot \sigma_R(\tilde{\boldsymbol{A}}\ \boldsymbol{X}\boldsymbol{W}^{(0)})\ \boldsymbol{W}^{(1)})$ (8), where $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\boldsymbol{D}^{-\frac{1}{2}}$, $\boldsymbol{D}$ is the diagonal degree matrix of $\hat{\boldsymbol{A}}$, $\hat{\boldsymbol{A}} = \boldsymbol{A}+\boldsymbol{I}$, $\boldsymbol{I}$ is the identity matrix, and σ_R is the ReLU activation function. We refer to this as PiNet (GCN). To evaluate the performance of our proposed architecture directly, we compare against a GCN with a dense layer applied to the concatenated vertex vectors and a GCN with a dense layer on the mean of its vertex vectors.

We also compare with three state of the art graph classifiers: DiffPool (21), DGCNN (15), and Patchy-SAN (14). We vary the number of training examples using stratified sampling and report the mean validation accuracy of 10 trials.

4.3 Experiment 2: Message passing mechanism

We extend the message passing matrix of (8) in which we add two additional trainable parameters, thus vector state is propagated by the matrix

$$\tilde{\boldsymbol{A}} = (p\boldsymbol{I} + (1-p)\boldsymbol{D})^{-\frac{1}{2}}(\boldsymbol{A} + q\boldsymbol{I})(p\boldsymbol{I} + (1-p)\boldsymbol{D})^{-\frac{1}{2}}, \tag{4}$$

where $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{D}$ is the diagonal degree matrix, and $\boldsymbol{A}$ is the graph adjacency matrix. p allows the model to optimise the extent to which to apply symmetric normalisation of the adjacency matrix, and q (as originally supposed for further work in (8)) allows the model to optimise the trade-off between keeping a vertex's own state and aggregating the states of its neighbours. Note that p and q are learned indirectly through optimising p' and q' with sigmoid to give $0 \leq p, q \leq 1$.

We compare the classification accuracy of the extreme cases of p and q ($\boldsymbol{A}$, $\boldsymbol{A}+\boldsymbol{I}$, $\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$, and $\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}$) against the learned p and q for each layer in each head. Following the methodology of (19) and (21) we perform 10-fold cross validation, reporting the mean validation accuracy for the single best epoch across the folds.

[2] Available at `https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets`.

4.4 Experiment 3: Benchmark

We benchmark the performance of PiNet with the original (8) and extended GCNs (4.3), on the benchmark datasets described above against the baseline and state of art methods used in 4.2, using the methodology as described in 4.2.

5 Results

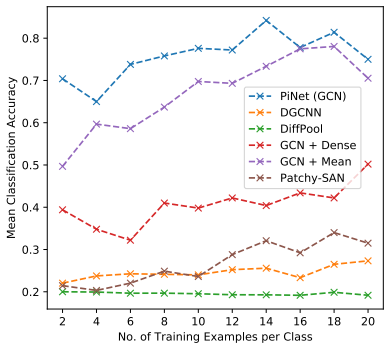

Figure 2: Mean classification accuracy over a range of training set sizes on the isomorphism dataset.

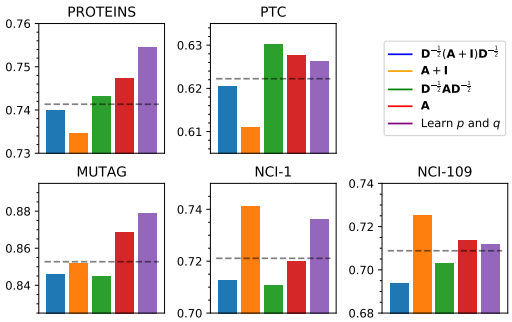

Figure 3: Mean classification accuracy for each message passing matrix within PiNet. Dashed lines indicate mean accuracy of manual search.

Table 1: Benchmark results. * indicates GCN with extended message passing with manual p and q, and ** with learned p and q.

	MUTAG	NCI-1	NCI-109	PROTEINS	PTC
GCN + Dense	0.86 ± 0.06	0.73 ± 0.03	0.72 ± 0.02	0.71 ± 0.04	0.63 ± 0.07
GCN + Mean	0.84 ± 0.07	0.68 ± 0.03	0.67 ± 0.03	0.74 ± 0.02	0.63 ± 0.04
Patchy-SAN	0.85 ± 0.06	0.58 ± 0.02	0.58 ± 0.03	0.70 ± 0.02	0.58 ± 0.02
DGCNN	0.86 ± 0.07	0.73 ± 0.03	0.72 ± 0.02	0.73 ± 0.05	0.61 ± 0.06
DiffPool	0.91 ± 0.08	0.73 ± 0.02	0.72 ± 0.03	0.80 ± 0.05	0.64 ± 0.07
PiNet (GCN)	0.85 ± 0.07	0.71 ± 0.03	0.69 ± 0.03	0.74 ± 0.05	0.62 ± 0.05
PiNet (GCN*)	0.87 ± 0.08	0.74 ± 0.03	0.73 ± 0.03	0.75 ± 0.06	0.63 ± 0.06
PiNet (GCN**)	0.88 ± 0.07	0.74 ± 0.02	0.71 ± 0.04	0.75 ± 0.06	0.63 ± 0.04

Experiment 1 (Figure 2) demonstrates the power of PiNet in capturing the most subtle differences between the test graphs, even with only 2 examples per class. Interestingly, this data presents a worst-case scenario for DiffPool and thus this method is unable to distinguish the different graph classes at all. In Experiment 2 (Figure 3) we see that while the optimal parameters p and q are not always found, the result of learning p and q offers better performance than the average of a manual search over the extreme values in all cases thus suggesting it is a suitable technique to reduce parameter searching. Finally, for the standard benchmark datasets we observe competitive performance with (within one standard deviation or better than) the state of the art methods for all datasets.

6 Conclusion

We have introduced PiNet, a generalised attention-based pooling mechanism for utilizing vertex-level convolution operators for graph level representations. We have demonstrated its ability to capture the finest subtleties in a graph isomorphism test and demonstrated results competitive with current state of the art methods on standard benchmark datasets. For further work we propose further study of PiNet with different convolution operators, as well as the use of skip connections to add great flexibility to the learned vertex representations prior to graph level pooling.

Acknowledgments

We thank Braintree Ltd. (`http://braintree.com`) for providing the full funding for this research.

References

[1] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," 2018. [Online]. Available: https://arxiv.org/pdf/1805.11973.pdfhttp://arxiv.org/abs/1805.11973

[2] M. Zitnik and J. Leskovec, "Predicting multicellular function through multi-layer tissue networks," in *Bioinformatics*, vol. 33, no. 14, 2017, pp. i190–i198. [Online]. Available: http://snap.stanford.

[3] Z. Harchaoui and F. Bach, "Image classification with segmentation graph kernels," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007. [Online]. Available: https://www.di.ens.fr/{~}fbach/harchaoui{_}bach{_}cvpr07.pdf

[4] D. H. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos, "Polonium: Tera-scale graph mining and inference for malware detection," in *Proceedings of the 11th SIAM International Conference on Data Mining, SDM 2011*, 2011.

[5] B. Graham, "Fractional Max-Pooling," 2014. [Online]. Available: https://arxiv.org/pdf/1412.6071.pdfhttp://arxiv.org/abs/1412.6071

[6] Y. Aytar, C. Vondrick, and A. Torralba, "SoundNet: Learning sound representations from unlabeled video," in *Advances in Neural Information Processing Systems*, 2016, pp. 892–900. [Online]. Available: http://www.cs.columbia.edu/{~}vondrick/soundnet.pdf

[7] Y. LeCun and Y. Bengio, "The Handbook of Brain Theory and Neural Networks," M. A. Arbib, Ed. Cambridge, MA, USA: MIT Press, 1998, ch. Convolutio, pp. 255–258. [Online]. Available: http://dl.acm.org/citation.cfm?id=303568.303704

[8] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations (ICLR)*, sep 2016. [Online]. Available: http://arxiv.org/abs/1609.02907

[9] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *NIPS*, 2017. [Online]. Available: https://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.pdfhttp://arxiv.org/abs/1706.02216

[10] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lì, and Y. Bengio, "Graph Attention Networks," in *ICLR*, 2018. [Online]. Available: https://arxiv.org/pdf/1710.10903.pdf

[11] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Advances in Neural Information Processing Systems*, 2014.

[12] M. Zaheer, S. Kottur, S. Ravanbhakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem, 2017, pp. 3392–3402. [Online]. Available: https://papers.nips.cc/paper/6931-deep-sets.pdf

[13] R. L. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro, "Janossy Pooling: Learning Deep Permutation-Invariant Functions for Variable-Size Inputs," in *ICLR*, 2019. [Online]. Available: https://arxiv.org/pdf/1811.01900.pdfhttp://arxiv.org/abs/1811.01900

[14] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning Convolutional Neural Networks for Graphs," vol. 1, 2016. [Online]. Available: http://arxiv.org/abs/1605.05273

[15] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018, pp. 4438–4445. [Online]. Available: www.aaai.org

[16] B. Y. Weisfeiler and A. A. Lehman, "Reduction of a graph to a canonical form and an algebra which appears in the process," *Nauchno-Technicheskaya Informatsiya, Ser. 2*, vol. 9, p. 12, 1968.

[17] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," in *Advances in neural information processing systems (NIPS)*, 2016. [Online]. Available: https://github.com/mdeff/cnn{_}graph

[18] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks," *Association for the Advancement of Artificial Intelligence*, 2019. [Online]. Available: www.aaai.orghttp://arxiv.org/abs/1810.02244

[19] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful Are Graph Neural Networks?" in *ICLR*, 2019. [Online]. Available: https://arxiv.org/pdf/1810.00826.pdf

[20] F. Wu, T. Zhang, A. H. de Souza, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying Graph Convolutional Networks," 2019. [Online]. Available: https://github.com/Tiiiger/SGChttp://arxiv.org/abs/1902.07153

[21] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical Graph Representation Learning with Differentiable Pooling," *32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.*, 2018. [Online]. Available: http://papers.nips.cc/paper/7729-hierarchical-graph-representation-learning-with-differentiable-pooling.pdfhttp://arxiv.org/abs/1806.08804

[22] J. B. Lee, R. Rossi, and X. Kong, "Graph classification using structural attention," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 1666–1674. [Online]. Available: https://doi.org/10.1145/3219819.3219980

[23] B. D. McKay and A. Piperno, "Practical graph isomorphism, II," *Journal of Symbolic Computation*, vol. 60, pp. 94–112, 2014. [Online]. Available: www.elsevier.com/locate/jsc

[24] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[25] P. Erdõs and A. Rényi, "On evolution of random graphs," *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, no. 5, pp. 17–61, 1960.

A Hyper-Parameters

In all experiments we use categorical cross-entropy for loss, and fix learning rate to 10^{-3}.

- PiNet (GCN): hidden sizes $\{32, 64\}$ for each layer in each head (two layers).
- GCN + Dense GCN + Mean: hidden sizes $\{32, 64\}$ for each layer (two layers).
- DiffPool: assign-ratio in $\{0.1, 0.2, 0.3\}$, hidden layer sizes in $\{30, 40, 50\}$ (for two layers)
- DGCNN: hidden sizes in $\{64, 96, 128\}$ and 3 sort pooling values selected according to the size of each dataset.
- Patchy-SAN: labelling procedures: NAUTY (23) and Betweenness Centrality (24).

B Isomorphism Dataset Generation

To generate the data we sample 5 unique Erdõs-Rényi graphs (25) with equal vertex degree distributions - this ensures a high level of challenge and prevents trivial classification. Each vertex is assigned one of two classes uniform randomly. The 5 unique graphs are then copied 99 times each and the vertex ids are permuted randomly on all of the graphs since we wish to test the ability to recognise isomorphic graphs even with different vertex orderings.