RELATIONAL GRAPH ATTENTION NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper we present Relational Graph Attention Networks, an extension of Graph Attention Networks to incorporate both node features and relational information into a masked attention mechanism, extending graph-based attention methods to a wider variety of problems, specifically, predicting the properties of molecules. We demonstrate that our attention mechanism gives competitive results on a molecular toxicity classification task (Tox21), enhancing the performance of its spectral-based convolutional equivalent. We also investigate the model on a series of transductive knowledge base completion tasks, where its performance is noticeably weaker. We provide insights as to why this may be, and suggest when it is appropriate to incorporate an attention layer into a graph architecture.

1 INTRODUCTION

Convolutional neural networks (CNNs) have been remarkably successful at a variety of tasks in Euclidean domains, such as image captioning (Donahue et al., 2017), human pose estimation (Toshev and Szegedy, 2014) and classifying videos (Karpathy et al., 2014). CNNs are successful because they make two important assumptions, which are valid for a variety of real-world data (particularly image and sound) (Henaff et al., 2015). Specifically, they assume the data domain is (Defferrard et al., 2016; Henaff et al., 2015; Bruna et al., 2013):

1. **Locally stationary.** Local features, such as corners, lines or textures, appear across the data domain. This is exploited in the architecture by the sharing of weights among filters. It follows that there is a significant reduction in the number of parameters and thus the model is less prone to over-fitting. More precisely: for input size n, k filters and p parameters per filter, there are $\mathcal{O}(kp)$ parameters, and hence the number of parameters is $\mathcal{O}(1)$ in the size of the input n.
2. **Compositional.** Naturally occurring complex features can be decomposed into simpler features. This is exploited by the architecture by stacking convolutional and pooling layers.

However, data is often in non-Euclidean domains, such as molecules or social networks. It is not clear how to exploit data with the above properties in non-Euclidean domains, since many operations we take for granted in Euclidean domains are ill-defined.

The study of generalising neural networks to non-Euclidean domains is called *geometric deep learning*. Current approaches may be roughly divided into three families: spectral, spatial and hybrid approaches (Bronstein et al., 2017).

Spectral approaches have a fundamental limitation in that they are basis-dependent. That is, when a spectral filter is learnt with respect to a basis on a domain, then re-used with a different basis on another domain, there is no guarantee the result is the same (Bronstein et al., 2017). Spatial approaches on non-Euclidean domains also have two major challenges: there is the lack of shift invariance, and there is no inherent coordinate system. Hybrid approaches combine these two approaches often to trade-off between spatial and spectral localisation.

Meanwhile, there has been recent interest in applying attention mechanisms to graphs, which have been shown to improve existing models on node classification, link prediction and graph

classification tasks (Lee et al., 2018). Further, there has also been interest in extending both attention mechanisms and the advancements in geometric deep learning to relational graphs. Tasks on relational graphs include molecule synthesis (Segler et al., 2018) and drug discovery (Feinberg et al., 2018). Recently, Schlichtkrull et al. (2018) proposed the Relational Graph Convolutional Network (RGCN); an extension of Kipf and Welling (2016) to relational graphs which achieved impressive performance on node classification and link prediction tasks.

We present two attention-based variants of RGCN to perform function approximation on graphs: Within-Relation Graph Attention (WIRGAT) and Across-Relation Graph Attention (ARGAT). We evaluate our proposed models on challenging transductive node classification and inductive graph classification tasks and show that Relational Graph Attention Network (RGAT) is extremely competitive on inductive graph classification yet performs relatively poorly on transductive node classification tasks, at least in the absence of node features. In hope to aid further investigation in this direction, we also provide a fully-vectorised, sparse, batched implementation of RGAT in `TensorFlow` which is fully compatible with `eager` execution mode.

2 RGAT ARCHITECTURE

2.1 RELATIONAL GRAPH ATTENTION LAYER

We follow the construction of the Graph Attention Network (GAT) layer in Veličković et al. (2017), extending to the relational setting, using ideas from Schlichtkrull et al. (2018) where appropriate.

The input to the layer is a graph with $R = |\mathcal{R}|$ relation types and N nodes. The i^{th}node is represented by a feature vector $\boldsymbol{h}_i \in \mathbb{R}^F$, and the features of all nodes are sumarised in the feature matrix $\boldsymbol{H} = [\boldsymbol{h}_1\, \boldsymbol{h}_2\, \dots\, \boldsymbol{h}_N] \in \mathbb{R}^{N \times F}$. The output of the layer is the transformed feature matrix $\boldsymbol{H'} = [\boldsymbol{h}'_1\, \boldsymbol{h}'_2\, \dots\, \boldsymbol{h}'_N] \in \mathbb{R}^{N \times F'}$, where $\boldsymbol{h}'_i \in \mathbb{R}^{F'}$ is the transformed feature vector of the i^{th}node.

Different relations convey distinct pieces of information. The update rule of Schlichtkrull et al. (2018) made this manifest by assigning each node i a distinct intermediate representation $\boldsymbol{g}_i^{(r)} \in \mathbb{R}^{F'}$ under relation r

$$\boldsymbol{G}^{(r)} = \boldsymbol{H}\, \boldsymbol{W}^{(r)} \in \mathbb{R}^{N \times F'}, \tag{1}$$

where $\boldsymbol{G}^{(r)} = \left[\boldsymbol{g}_1^{(r)}\, \boldsymbol{g}_2^{(r)}\, \dots\, \boldsymbol{g}_N^{(r)}\right]$ is the intermediate representation feature matrix under relation r, and $\boldsymbol{W}^{(r)} \in \mathbb{R}^{F \times F'}$ are the learnable parameters of a shared linear transformation.

Following Veličković et al. (2017), we assume the attention coefficient between two nodes is based only on the features of those nodes up to some neighborhood-level normalisation. To keep computational complexity linear in R, we assume that, given linear transformations $\boldsymbol{W}^{(r)}$, the logits $E_{i,j}^{(r)}$ of each relation r are independent of each other

$$E_{i,j}^{(r)} = a\left(\boldsymbol{g}_i^{(r)}, \boldsymbol{g}_j^{(r)}\right), \tag{2}$$

and indicate the importance of node j's intermediate representation to that of node i under relation r. The attention is masked so that, for node i, coefficients $\alpha_{i,j}^{(r)}$ exist only for $j \in \mathcal{N}_i^{(r)}$, where $\mathcal{N}_i^{(r)}$ denotes the set of neighbor indices of node i under relation $r \in \mathcal{R}$. We choose the specific realisation of a in Equation (2) given in Veličković et al. (2017)

$$E_{i,j}^{(r)} = \text{LeakyReLu}\left[(\boldsymbol{a}^{(r)})^T(\boldsymbol{g}_i^{(r)} \oplus \boldsymbol{g}_j^{(r)})\right], \tag{3}$$

where $\boldsymbol{x}^T$ denotes the transpose of $\boldsymbol{x}$, $\boldsymbol{a}^{(r)} \in \mathbb{R}^{2F'}$ are learnable parameters of the attention mechanism under relation r, and $\boldsymbol{x} \oplus \boldsymbol{y}$ denotes the vector concatenation of $\boldsymbol{x}$ and $\boldsymbol{y}$.

For comparison, the coefficients of RGCN are given by $\alpha_{i,j}^{(r)} = |\mathcal{N}_i^{(r)}|^{-1}$. This encodes the prior that the intermediate representations of nodes $j \in \mathcal{N}_i^{(r)}$ to node i under relation r are equally important.

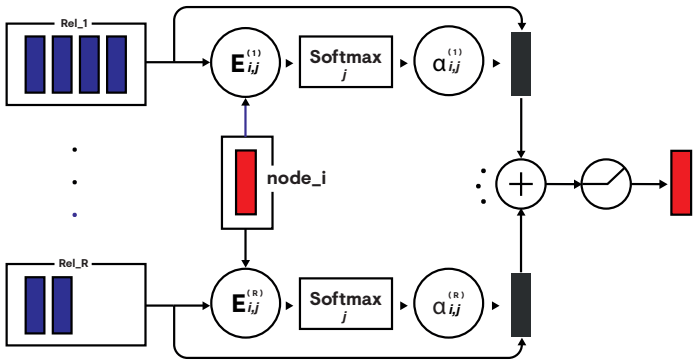


Figure 1: WIRGAT. The intermediate representations for node i (left red rectange) are combined with the intermediate representations for nodes in its neighborhood (blue rectangles) under each relation r, to form each logit $E_{i,j}^{(r)}$. A softmax is taken over each logit matrix for each relation type to form the attention coefficients $\alpha_{i,j}^{(r)}$. These attention coefficients construct a weighted sum over the nodes in the neighborhood for each relation (black rectangle). These are then aggregated and passed through a nonlinearity to produce the updated representation for node i (right red rectangle).

The attention coefficients should be comparable across nodes. This can be achieved by applying softmax appropriately. We investigate two candidates.

WIRGAT The simplest way to take the softmax over the logits $E_{i,j}^{(r)}$ of Equation (3) replicates the softmax taken in Veličković et al. (2017) for each relation r

$$\alpha_{i,j}^{(r)} = \underset{j}{\text{softmax}}\left(E_{i,j}^{(r)}\right) = \frac{\exp\left(E_{i,j}^{(r)}\right)}{\sum_{k\in\mathcal{n}_i^{(r)}}\exp\left(E_{i,k}^{(r)}\right)}, \qquad \forall\, i, r: \sum_{j\in\mathcal{n}_i^{(r)}} \alpha_{i,j}^{(r)} = 1. \tag{4}$$

We call the attention in Equation (4) WIRGAT and it is shown in Figure 1. This mechanism encodes the prior that relation importance is a purely global property of the graph by implementing an independent probability distribution over nodes in the neighborhood of i for each relation r. Explicitly, for any node i and relation r, nodes $j, k \in \mathcal{n}_i^{(r)}$ yield competing attention coefficients $\alpha_{i,j}^{(r)}$ and $\alpha_{i,k}^{(r)}$ with sizes depending on their corresponding representations $\boldsymbol{g}_j^{(r)}$ and $\boldsymbol{g}_k^{(r)}$. There is no competition between any attention coefficients $\alpha_{i,j}^{(r)}$ and $\alpha_{i,k}^{(r')}$ for all nodes i and nodes $j \in \mathcal{n}_i^{(r)}, j' \in \mathcal{n}^{(r')}$ where $r' \neq r$ irrespective of node representations.

ARGAT An alternative way to take the softmax over the logits $E_{i,j}^{(r)}$ of Equation (3) is across node neighborhoods irrespective of relation r

$$\alpha_{i,j}^{(r)} = \underset{j,r}{\text{softmax}}\left(E_{i,j}^{(r)}\right) = \frac{\exp\left(E_{i,j}^{(r)}\right)}{\sum_{r'\in\mathcal{R}}\sum_{k\in\mathcal{n}_i^{(r')}}\exp\left(E_{i,k}^{(r')}\right)}, \qquad \forall\, i: \sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{n}_i^{(r)}} \alpha_{i,j}^{(r)} = 1. \tag{5}$$

We call the attention in Equation (5) ARGAT and it is shown in Figure 2. This mechanism encodes the prior that relation importance is a local property of the graph by implementing a single probability distribution over the different representations $\boldsymbol{g}_j^{(r)}$ for nodes j in the neighborhood of node i. Explicitly, for any node i and all $r, r' \in \mathcal{R}$, all nodes $j \in \mathcal{n}_i^{(r)}$ and $k \in \mathcal{n}_i^{(r')}$ yield competing attention coefficients $\alpha_{i,j}^{(r)}$ and $\alpha_{i,k}^{(r')}$ with sizes depending on their corresponding representations $\boldsymbol{g}_j^{(r)}$ and $\boldsymbol{g}_k^{(r')}$.

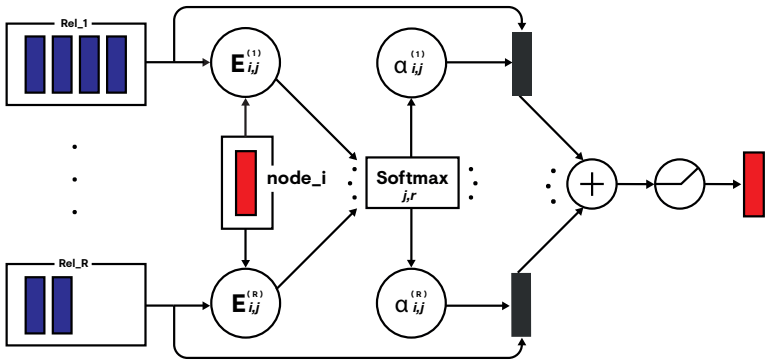


Figure 2: ARGAT. The logits are produced identically to those in Figure 1. A softmax is taken across all logits independent of relation type to form the attention coefficients $\alpha_{i,j}^{(r)}$. The remaining weighting and aggregation steps are the same as those in Figure 1.

Propagation rule Combining the attention mechanism of either Equation (4) or Equation (5) with the neighborhood aggregation step of Schlichtkrull et al. (2018) gives

$$\boldsymbol{h}_i' = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^{(r)}} \alpha_{i,j}^{(r)}\,\boldsymbol{g}_j^{(r)}\right) \in \mathbb{R}^{N\times F'}, \tag{6}$$

where σ represents an optional nonlinearity. Similar to Vaswani et al. (2017); Veličković et al. (2017), we also find that using multiple heads in the attention mechanism can enhance performance

$$\boldsymbol{h}_i' = \bigoplus_{k=1}^{K}\sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^{(r)}} \alpha_{i,j}^{(r,k)}\,\boldsymbol{g}_j^{(r,k)}\right) \in \mathbb{R}^{N\times K\,F'}, \tag{7}$$

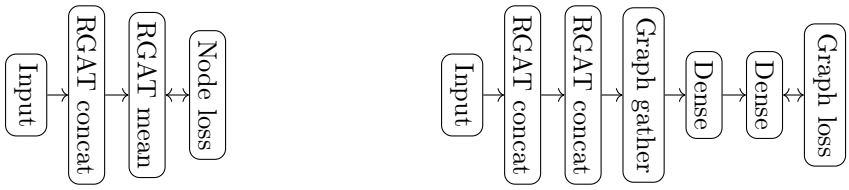
where $\oplus$ denotes vector concatenation, $\alpha_{i,j}^{(r,k)}$ are the normalised attention coefficients under relation r computed by either WIRGAT or ARGAT, and $\boldsymbol{g}_i^{(r,k)} = \boldsymbol{h}_i\,\left(\boldsymbol{W}^{(r,k)}\right)^T$ is the head specific intermediate representation of node i under relation r.

It might be interesting to consider cases where there are a different number of heads for different relationship types, as well as when a mixture of ARGAT and WIRGAT produce the attention coefficients, however, we leave that subject for future investigation and will not consider it further.

Basis decomposition The number of parameters in the RGAT layer increases linearly with the number of relations R and heads K, and can lead quickly to overparameterization. In RGCNs it was found that decomposing the kernels was beneficial for generalisation, although it comes at the cost of increased model bias (Schlichtkrull et al., 2018). We follow this approach, decomposing both the kernels $\boldsymbol{W}^{(r,k)}$ as well as the kernels of attention mechanism $\boldsymbol{a}^{(r,k)}$ into B_V basis matrices $\boldsymbol{V}^{(b)} \in \mathbb{R}^{F\times F'}$ and B_v basis vectors $\boldsymbol{v}^{(b)} \in \mathbb{R}^{2F'}$

$$\boldsymbol{W}^{(r,k)} = \sum_{b=1}^{B_W} c_b^{(r,k)}\,\boldsymbol{V}^{(b)}, \qquad\qquad \boldsymbol{a}^{(r,k)} = \sum_{b=1}^{B_v} d_b^{(r,k)}\,\boldsymbol{v}^{(b)}, \tag{8}$$

where $c_b^{(r,k)}, d_b^{(r,k)} \in \mathbb{R}$ are basis coefficients. We consider models using full and decomposed $\boldsymbol{W}$ and $\boldsymbol{a}$.



(a) Node classification (b) Multi-task graph classification.

Figure 3: (a) The network architecture used for node classification on AIFB and MUTAG. This architecture is the same as in Schlichtkrull et al. (2018) except with RGCNs replaced with RGATs. (b) The network architecture used for multi-task graph classification on Tox21. This architecture is the same as the Graph Convolutional Networks (GCNs) architecture in Altae-Tran et al. (2016) except with RGCNs replaces with GATs and we do not use graph pooling.

2.2 Node classification

For the transductive task of semi-supervised node classification, we employ a two-layer RGAT architecture shown in Figure 3a. We use a Rectified Linear Unit (ReLU) activation after the RGAT concat layer, and a node-wise softmax on the final layer to produce an estimate for the probability that the i^{th}label is in the class α

$$P(\text{class}_i = \alpha) \approx \hat{y}_{i,\alpha} = \text{softmax}(\boldsymbol{h}_i^{(2)})_\alpha. \tag{9}$$

We then employ a masked cross-entropy loss $\mathcal{L}$ to constrain the network updates to the subset of nodes $\mathcal{Y}$ whose labels are known

$$\mathcal{L} = -\sum_{i \in \mathcal{Y}} \sum_{\alpha=1}^{n_{\text{classes}}} y_{i,\alpha} \ln\left(\hat{y}_{i,\alpha}\right), \tag{10}$$

where $\boldsymbol{y}_i$ is the one-hot representation of the true label for node i.

2.3 Graph classification

For inductive graph classification, we employ a two-layer RGAT followed by a graph gather and dense network architecture shown in Figure 3b. We use ReLU activations after each RGAT layer and the first dense layer. We use a tanh activation after the GraphGather : $\mathbb{R}^{N \times F} \to \mathbb{R}^{2F}$, which is given by

$$\boldsymbol{H}' = \text{GraphGather}(\boldsymbol{H}) = \left(\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{h}_i\right) \oplus \left[\bigoplus_{f=1}^{F} \max_i h_{i,f}\right], \tag{11}$$

and is the vector concatenation of the mean of the node representations with the feature-wise max of the node representations.

The final dense layer then produces logits of the size $n_{\text{classes}} \times n_{\text{tasks}}$, and we apply a task-wise softmax to its output to produce an estimate $\hat{y}_{t,\alpha}$ for the probability that the graph is in class α for a given task t, analogous to Equation (9). Weighted cross-entropy loss $\mathcal{L}$ is then used to form the learning objective

$$\mathcal{L}(w, y, \hat{y}) = -\sum_{t=1}^{n_{\text{tasks}}} \sum_{\alpha=1}^{n_{\text{classes}}} w_{t,\alpha}\, y_{t,\alpha} \ln\left(\hat{y}_{t,\alpha}\right), \tag{12}$$

where $w_{t,\alpha}$ and $y_{t,\alpha}$ are the weights and one-hot true labels for task t and class α respectively.

3 Evaluation

3.1 Datasets

We evaluate our models on both transductive and inductive node classification tasks. Following the experimental setup of Schlichtkrull et al. (2018) for the transductive tasks, we

evaluate our model on the Resource Description Framework (RDF) datasets AIFB and MUTAG. We also evaluate our model for an inductive task on the molecular dataset, Tox21. Details of these data sets are given in Table 1. The reader is reffered to Ristoski and Paulheim (2016) and Wu et al. (2018) for further details on the transductive and inductive datasets respectively.

Datasets	AIFB	MUTAG	Tox21
Task	Transductive	Transductive	Inductive
Nodes	8,285 (1 graph)	23,644 (1 graph)	145,459 (8014 graphs)
Edges	29,043	74,227	151,095
Relations	45	23	4
Labelled	176	340	96,168 (12 per graph)
Classes	4	2	12 (multi-label)
Train nodes	112	218	(6411 graphs)
Validation nodes	28	54	(801 graphs)
Test nodes	28	54	(802 graphs)

Table 1: A summary of the datasets used in our experiments and how they're partitioned.

Transductive baselines We consider as a baseline the recent state-of-the-art results from Schlichtkrull et al. (2018) obtained with a two-layer RGCN model with 16 hidden units and basis function decomposition. We also include the same challenging baselines of FEAT (Paulheim and Fümkranz, 2012), WL (Shervashidze et al., 2011; de Vries and de Rooij, 2015) and RDF2Vec (Ristoski and Paulheim, 2016). In depth details of these baselines are given by Ristoski and Paulheim (2016).

Inductive baselines As baselines for Tox21, we compare against the most competetive methods on Tox21 reported in Wu et al. (2018). Specficially, we compare against deep multi-task networks Ramsundar et al. (2015) whose inputs are Extended Connectivity Fingerprintss (ECFP4s) generated by RDKit, deep bypass multitask networks, which add in connections that bypass task-shared representations to overcome the difficulties multitask networks have in accounting for unrelated sample variance across tasks Wu et al. (2018), Weave, which is a GCN-style model incorporating connections between all atoms in a molecule and leveraging the properties between these molecules as edge features Kearnes et al. (2016), and a RGCN model whose relational structure is determined by the degree of the node to be updated Altae-Tran et al. (2016). Specficially, up to some maximum degree D,

$$\boldsymbol{h}_i' = \sigma\left[(\boldsymbol{W}^{\deg(\mathrm{i})})^T\boldsymbol{h}_i + \sum_{j\in n_i}(\boldsymbol{U}^{\deg(\mathrm{i})})^T\boldsymbol{h}_j + \boldsymbol{b}^{\deg(i)}\right], \tag{13}$$

where $\boldsymbol{W}^{\deg(i)} \in \mathbb{R}^{F\times F'}$ is a degree-specific linear transformation for self-connections, $\boldsymbol{U}^{\deg(\mathrm{i})} \in \mathbb{R}^{F\times F'}$ is a degree-specific linear transformation for neighbours into their intermediate representations $\boldsymbol{g}_i \in \mathbb{R}^{F'}$, and $\boldsymbol{b}^{\deg(i)}$ is a degree-specific bias. Any update for any degree $d(i) > D$ gets assigned to the update for the maximum degree D.

3.2 Experimental setup

Transductive learning For the transductive learning tasks we apply the architecture discussed in Section 2.2. Its hyperparameters were optimised for both AIFB and MUTAG on their respective training/validation sets defined in Ristoski and Paulheim (2016), using 5-fold cross validation. Using the found hyperparameters, we retrain on the full training set and report results on the test set across 200 seeds. We employ early stopping on the validation set during cross-validation to determine the number of epochs we will run on the final training set.

Inductive learning For the inductive learning tasks we apply the architecture discussed in Section 2.3. In order to hyperparameter optimise once, ensure no data leakage, but also

Model	AIFB	MUTAG
Feat	55.55 ± 0.00	77.94 ± 0.00
WL	80.55 ± 0.00	$\mathbf{80.88} \pm 0.00$
RDF2Vec	88.88 ± 0.00	67.20 ± 1.24
R-GCN	95.83 ± 0.62	73.23 ± 0.48
C-WIRGAT	$\mathbf{96.85} \pm 0.01$	69.37 ± 0.03
WIRGAT	96.83 ± 0.01	69.83 ± 0.01
C-ARGAT	93.05 ± 0.03	63.69 ± 0.08
ARGAT	94.01 ± 0.03	65.54 ± 0.06

(a) Transductive

Model	Tox21
Multitask	0.803 ± 0.012
Bypass	0.810 ± 0.013
Weave	0.820 ± 0.010
R-GCN	0.829 ± 0.006
C-WIRGAT	0.829 ± 0.010
WIRGAT	$\mathbf{0.835} \pm 0.006$
C-ARGAT	0.832 ± 0.009
ARGAT	$\mathbf{0.835} \pm 0.006$

(b) Inductive

Table 2: (a) Entity classification results in accuracy (mean and standard deviation over 10 runs) for FEAT (Paulheim and Fümkranz, 2012), WL (Shervashidze et al., 2011; de Vries and de Rooij, 2015), RDF2Vec (Ristoski and Paulheim, 2016) and RGCN (Schlichtkrull et al., 2018), and (mean and standard deviation over 100 runs) for (C-)WIRGAT and (C-)WIRGAT (this work, mean and standard deviation over 200 runs). Test performance is reported on the splits provided in Ristoski and Paulheim (2016). (b) Graph classification mean Area Under the Curve (AUC) across all 12 tasks (mean and standard deviation over 3 splits) for Multitask (Ramsundar et al., 2015), Bypass (Wu et al., 2018), Weave (Kearnes et al., 2016), RGCN (Altae-Tran et al., 2016), (C-)WIRGAT and (C-)ARGAT (this work, 2 seeds per split). Test performance is reported on the splits provided in Wu et al. (2018). For completeness, we present the training and validation mean AUCs alongside the test AUCs in Appendix A.

provide comparable benchmarks to those presented in Wu et al. (2018), we took the three benchmark splits from the `MolNet benchmarks`[1], isolated graphs belonging to any of the test sets. Using the remaining graphs we formed a hyperparameter search using 2 folds of 10-fold cross validation. Using the found hyperparameters, we then retrained on the three benchmark splits provided with 2 seeds each, giving an unbiased estimate of model performance. We employ early stopping during both the cross-validation and final run (the validation set of the inductive task is available for the final benchmark, in contrast to the transductive tasks) to determine the number of training epochs.

Details of hyperparameter optimisation are given in Appendix B.

In all experiments, we train with the attention mechanism turned on. At evaluation time, however, we report results with and without the attention mechanism to provide insight into whether the attention mechanism helps. ARGAT (WIRGAT) without the attention is called C-ARGAT (C-WIRGAT). In all experiments we use the Adam optimiser (Kingma and Ba, 2014).

3.3 Results

Validation of setup Before enabling the full relational attention mechanism, we performed several

checks of our setup and implementation. First, noting that by setting the number of relations to one, both ARGAT and WIRGAT reduce to GAT. We train on Cora with the setup described in Veličković et al. (2017), and reproduce their results, averaging over 100 runs. Similarly, by setting the logits of the attention mechanism to zero, the WIRGAT mechanism reduces to the RGCN of Schlichtkrull et al. (2018). We train on both AIFB and MUTAG, and reproduce their results, averaging over 100 runs.

Transductive learning In Table 2a we evaluate the RGAT model on MUTAG and AIFB. We see that WIRGAT outperforms ARGAT on all tasks. This is in line with the finding of

[1]Retrieved from `http://deepchem.io.s3-website-us-west-1.amazonaws.com/trained_models/Hyperparameter_MoleculeNetv3.tar.gz`.

Schlichtkrull et al. (2018) where normalisation done within relation type was found to give the best performance. For node classification tasks on RDF data, this strongly indicates that the importance of a relation type does not alter much (if at all) across the graph.

When compared to constant attention mechanisms, we see a relative performance improvement on AIFB for ARGAT of 1.03%, and on MUTAG 0.6% and 2.9% for WIRGAT and ARGAT respectively. In the case of WIRGAT on AIFB, there is a performance drop when using attention compared with constant attention, however, this effect is within observed model variance.

Although we report a state-of-the art result on AIFB with (C-)WIRGAT, since the performance with and without attention are identical up to model variance, it is unlikely that this is due to the attention mechanism itself. The performance of this result is more likely attributed to a thorough hyperparameter search that includes a larger variety of regularisation options. We suspect the same performance can be obtained by RGCN.

We note that RGCN consistently outperforms RGAT on MUTAG, contrary to what might be expected (Schlichtkrull et al., 2018). The result is initially surprising given that RGCN lies within the parameter space of RGAT (where the attention kernel is zero), and we explicitly check that performance point with C-WIRGAT. In our experiments we have observed that both RGCN and RGAT can memorise the MUTAG training set with 100% accuracy without much difficulty (this is not the case for AIFB). The performance gap between RGCN and RGAT could then be explained by the following:

- During the training phase, the RGAT layer uses its attention mechanism to solve the learning objective. Once the objective is solved, the model is not encouraged by the loss function to seek a point in the parameter space that would also behave well when attention is set to a normalising constant within neighbourhoods (i.e. the parameter space point that would be found by RGCN).

- The RDF tasks are transductive, meaning that a basis-dependent spectral approach is sufficient to solve them. As RGCN already memorises the MUTAG training set, a model more complex[2] than RGCN, for example RGAT, that can also memorise the training set is unlikely to generalise as well (although this is a hotly debated topic (Zhang et al., 2016)).

We employed a suite of regularisation techniques to get RGAT to generalise on MUTAG, including L2-norm penalties, dropout in multiple places, batch normalisation, parameter reduction and early stopping, however, no evaluated harshly regularised points for RGAT generalise well on MUTAG.

Our final observation is that the attention mechanism presented in Section 2.1 relies on node features. The input node features on AIFB and MUTAG have to be learned from scratch (the input feature matrix is a one-hot index for nodes in the graph) as part of the task, and so it is possible that in the semi-supervised setup of MUTAG there is insufficient signal in the data to learn both the input node embeddings as well as the attention mechanism that acts upon them.

Inductive learning In Table 2b we evaluate the RGAT model on Tox21. Both WIRGAT and ARGAT outperform RGCN by a relative margin of 0.7% on mean AUC, yielding competetive results for Tox21. Compared to their constant attention versions, WIRGAT sees a relative improvement of 0.7% and ARGAT sees a relative improvement of 0.3%. The competetive nature of RGAT compared to RGCN on Tox21 is indicative of the gains that be made by the appropriate choice of non-spectral models on inductive tasks.

[2]Measured in terms of Minimum Description Length (MDL), for example.

4 Related Work

4.1 Spectral approaches

Spectral approaches define the convolutional operator using the spectral representation of a graph. Notably, Bruna et al. (2013) was the first to generalise CNNs to graphs based on the spectrum of the graph Laplacian. However, their approach involves the costly operation of eigendecomposition, the filters in the spectral domain are not guaranteed to be localised in the spatial domain, and the number of free parameters per filter are $\mathcal{O}(n)$ in the size of the input (Bronstein et al., 2017). Henaff et al. (2015) proposed a variant with smooth spectral multipliers, which guarantee spatially-localised filters and are $\mathcal{O}(1)$ in the size of the input. Subsequently, Defferrard et al. (2016) used Chebyshev polynomials as a basis for the filters which does not require eigendecomposition. Using this insight, Kipf and Welling (2016) restricted filters to a 1-hop neighbourhood for a fast and scalable propagation rule. Schlichtkrull et al. (2018) extended this approach to relational graphs by computing a weighted average of the hidden representations of each relationship type.

4.2 Spatial approaches

Spatial approaches define the convolutional operator in the spatial domain, i.e. directly on the graph. There are two major difficulties with this approach: there is the lack of shift invariance, and there is no inherent coordinate system in graphs. Duvenaud et al. (2015) learns a weight matrix per node degree to solve these problems, but is prohibitively slow for large graphs. A more scalable approach was proposed by Atwood and Towsley (2016), who define a convolution operation using random walks by considering a power series of a transition matrix with a separate weight matrix learnt for each neighbourhood size and graph feature. Monti et al. (2017) proposed using a mixture of Gaussian kernels defined on pseudo-coordinate systems local to nodes as convolutional operators. Further, they show previous approaches are special cases of their approach (Masci et al., 2015; Boscaini et al., 2016).

4.3 Hybrid approaches

Hybrid approaches combine spectral and non-spectral approaches. Shuman et al. (2016) generalises the windowed Fourier transform; a classic technique allowing for an explicit trade-off between spatial and spectral localisation through the choice of a window function (Bronstein et al., 2017). Rustamov and Guibas (2013), Szlam et al. (2005) and Gavish et al. (2010) are among those that have also extended wavelets to non-Euclidean domains to achieve a similar effect.

4.4 Attention mechanisms in graphs

Attention mechanisms aid models to focus on relevant parts of the input. Veličković et al. (2017) proposes a shared attention mechanism where attention coefficients between nodes depend on their respective features. A similar architecture is given by Thekumparampil et al. (2018), which uses a weighted cosine similarity between node features instead of training an MLP to compute the attention coefficients. Further, Shang et al. (2018) proposed attention coefficients which solely depend on the edge types, but do not leverage (often available) node features.

5 Conclusion

We have presented two attention mechanisms within the context of Relational Graph Attention Networks (RGATs), Within-Relation Graph Attention (WIRGAT) and Across-Relation Graph Attention (ARGAT). These mechanisms act upon graph structures, inducing a masked self-attention that takes account of local relational structure as well as node features. This allows both nodes and their corresponding properties under specific relations to be

dynamically assigned an importance for different nodes in the graph, and opens up spectral graph-based approaches to a wider variety of problems.

The models based on RGAT perform competitively or poorly on established baselines. This behaviour appears strongly task-dependant. Specifically, relational inductive tasks, such as graph classification, benefit the most from the RGAT, whereas transductive relational tasks, such as knowledge base completion, at least in the absense of node features, seem better tackled using spectral methods like Relational Graph Convolutional Networks (RGCNs) or other graph feature extraction methods like Weisfeiler-Lehman (WL) graph kernels. Future work should establish whether the deficiencies of RGAT compared to RGCN on relational transductive tasks is a general property of the model, or specific to the tasks we have investigated.

A Tox21 Results

For completeness, we present the training, validation and test set performance of our models in addition to those in Wu et al. (2018) in Table 3.

Model	Training	Validation	Test
Multitask	0.884 ± 0.001	0.795 ± 0.017	0.803 ± 0.012
Bypass	0.938 ± 0.001	0.800 ± 0.008	0.810 ± 0.013
Weave	0.875 ± 0.004	0.828 ± 0.008	0.820 ± 0.010
R-GCN	$\mathbf{0.905} \pm 0.004$	0.825 ± 0.013	0.829 ± 0.006
C-WIRGAT	0.897 ± 0.022	0.842 ± 0.04	0.829 ± 0.010
WIRGAT	0.902 ± 0.02	0.845 ± 0.005	$\mathbf{0.835} \pm 0.006$
C-ARGAT	0.884 ± 0.01	0.848 ± 0.003	0.832 ± 0.009
ARGAT	0.896 ± 0.02	$\mathbf{0.851} \pm 0.004$	$\mathbf{0.835} \pm 0.006$

Table 3: Graph classification mean Area Under the Curve (AUC) across all 12 tasks (mean and standard deviation over 3 splits) for Multitask (Ramsundar et al., 2015), Bypass (Wu et al., 2018), Weave (Kearnes et al., 2016), RGCN (Altae-Tran et al., 2016), WIRGAT and ARGAT (this work). Training, validation and test performance is reported on the splits provided in Wu et al. (2018).

B Hyperparameters

We perform hyperparameter optimisation using `hyperopt` Bergstra et al. (2013) over the space specified in Table 4

References

Altae-Tran, H., B. Ramsundar, A. S. Pappu, and V. Pande
2016. Low Data Drug Discovery with One-shot Learning. Pp. 1–20.

Atwood, J. and D. Towsley
2016. Diffusion-Convolutional Neural Networks. (Nips).

Bergstra, J., D. Yamins, and D. Cox
2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Machine Learning*, S. Dasgupta and D. McAllester, eds., volume 28 of *Proceedings of Machine Learning Research*, Pp. 115–123, Atlanta, Georgia, USA. PMLR.

Boscaini, D., J. Masci, E. Rodol, and M. Bronstein
2016. Learning shape correspondence with anisotropic convolutional neural networks. (Nips):1–9.

Hyperparameter	Distributions
Units (graph)	$\text{MultiplesOfFour}(4, 20)^\dagger$, $\text{MultiplesOfEight}(32, 128)^*$
Units (dense)	$\text{MultiplesOfEight}(32, 128)^*$
Heads	$\text{OneOf}(1, 2, 4)^\dagger$, $\text{OneOf}(1, 2, 4, 8)^*$
Feature dropout	$\text{Uniform}(0.0, 0.8)$
Edge dropout	$\text{Uniform}(0.0, 0.8)$
$\boldsymbol{W}$ basis size	$\text{OneOf}(\text{Full}, 5, 10, 20, 30)^\dagger$, $\text{Full}^\dagger$
$\boldsymbol{W}$ L2 coef (1)	$\text{LogUniform}(10^{-6}, 10^{-1})$
$\boldsymbol{W}$ L2 coef (2)	$\text{LogUniform}(10^{-6}, 10^{-1})$
$\boldsymbol{a}$ basis size	$\text{OneOf}(\text{Full}, 5, 10, 20, 30)^\dagger$, $\text{Full}^\dagger$
$\boldsymbol{a}$ L2 coef (1)	$\text{LogUniform}(10^{-6}, 10^{-1})$
$\boldsymbol{a}$ L2 coef (2)	$\text{LogUniform}(10^{-6}, 10^{-1})$
Learning rate	$\text{LogUniform}(10^{-5}, 10^{-1})$
Use bias	$\text{OneOf}(\text{Yes}, \text{No})$
Use batch normalisation	$\text{OneOf}(\text{Yes}, \text{No})$

Table 4: Hyperparameter search space for the transductive and inductive relational tasks. Hyperparameters indicated with a † are for transductive tasks only, hyperparameters indicated with a * are for inductive tasks only. The remaining hyperparameters are common to all tasks. For inductive tasks, the batch size was held at 64. When multihead attention is used, the number of units per head is appropriately reduced in order to keep the total number of output units of an RGAT layer independent of the number of heads.

Bronstein, M. M., J. Bruna, Y. Lecun, A. Szlam, and P. Vandergheynst
2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

Bruna, J., W. Zaremba, A. Szlam, and Y. LeCun
2013. Spectral Networks and Locally Connected Networks on Graphs. Pp. 1–14.

de Vries, G. K. D. and S. de Rooij
2015. Substructure counting graph kernels for machine learning from rdf data. *Web Semant.*, 35(P2):71–84.

Defferrard, M., X. Bresson, and P. Vandergheynst
2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. (Nips).

Donahue, J., L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell
2017. Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691.

Duvenaud, D., D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams
2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints.

Feinberg, E. N., D. Sur, B. E. Husic, D. Mai, Y. Li, J. Yang, B. Ramsundar, and V. S. Pande
2018. Spatial Graph Convolutions for Drug Discovery. Pp. 1–14.

Gavish, M., B. Nadler, R. R. Coifman, and N. Haven
2010. Multiscale Wavelets on Trees, Graphs and High Dimensional Data: Theory and Applications to Semi-Supervised Learning. *Icml*, (i):367–374.

Henaff, M., J. Bruna, and Y. LeCun
2015. Deep Convolutional Networks on Graph-Structured Data. Pp. 1–10.

Karpathy, A., G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li
2014. Large-scale video classification with convolutional neural networks. *Proc. IEEE CVPR.*

Kearnes, S., K. McCloskey, M. Berndl, V. Pande, and P. Riley
2016. Molecular graph convolutions: moving beyond fingerprints. *J. Comput. Aided. Mol. Des.*, 30(8):595–608.

Kingma, D. P. and J. Ba
2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kipf, T. N. and M. Welling
2016. Semi-Supervised Classification with Graph Convolutional Networks.

Lee, J. B., R. A. Rossi, S. Kim, N. K. Ahmed, and E. Koh
2018. Attention Models in Graphs: A Survey. 0(1).

Masci, J., D. Boscaini, M. M. Bronstein, and P. Vandergheynst
2015. Geodesic convolutional neural networks on Riemannian manifolds.

Monti, F., D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein
2017. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-January, Pp. 5425–5434.

Paulheim, H. and J. Fümkranz
2012. Unsupervised generation of data mining features from linked open data. In *Proc. 2nd Int. Conf. Web Intell. Min. Semant. - WIMS '12*, P. 1, New York, New York, USA. ACM Press.

Ramsundar, B., S. Kearnes, P. Riley, D. Webster, D. Konerding, and V. Pande
2015. Massively Multitask Networks for Drug Discovery. (Icml).

Ristoski, P. and H. Paulheim
2016. RDF2Vec: RDF graph embeddings for data mining. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9981 LNCS, Pp. 498–514.

Rustamov, R. and L. Guibas
2013. Wavelets on graphs via deep learning. *Nips*, Pp. 1–9.

Schlichtkrull, M., T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling
2018. Modeling Relational Data with Graph Convolutional Networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10843 LNCS, Pp. 593–607.

Segler, M. H., M. Preuss, and M. P. Waller
2018. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*.

Shang, C., Q. Liu, K.-S. Chen, J. Sun, J. Lu, J. Yi, and J. Bi
2018. Edge Attention-based Multi-Relational Graph Convolutional Networks.

Shervashidze, N., P. Schweitzer, E. Jan van Leeuwen, K. Mehlhorn, and K. Borgwardt
2011. Weisfeiler-Lehman Graph Kernels. *J. Mach. Learn. Res.*, 12:2539–2561.

Shuman, D. I., B. Ricaud, and P. Vandergheynst
2016. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291.

Szlam, A. D., M. Maggioni, R. R. Coifman, and J. C. BremerJr.
2005. Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions. P. 59141D.

Thekumparampil, K. K., C. Wang, S. Oh, and L.-J. Li
2018. Attention-based Graph Neural Network for Semi-supervised Learning. Pp. 1–15.

Toshev, A. and C. Szegedy
2014. DeepPose: Human Pose Estimation via Deep Neural Networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 1653–1660.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin
2017. Attention is all you need. *CoRR*, abs/1706.03762.

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio
2017. Graph Attention Networks.

Wu, Z., B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande
2018. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530.

Zhang, C., S. Bengio, M. Hardt, B. Recht, and O. Vinyals
2016. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530.