CAUSALLY-GUIDED REGULARIZATION OF GRAPH ATTENTION IMPROVES GENERALIZABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph attention networks estimate the relational importance of node neighbors to aggregate relevant information over local neighborhoods for a prediction task. However, the inferred attentions are vulnerable to spurious correlations and connectivity in the training data, hampering the generalizability of the model. We introduce CAR, a general-purpose regularization framework for graph attention networks. Embodying a causal inference approach, CAR aligns the attention mechanism with the causal effects of active interventions on graph connectivity in a scalable manner. CAR is compatible with a variety of graph attention architectures, and we show that it systematically improves generalizability on various node classification tasks. Our ablation studies indicate that CAR hones in on the aspects of graph structure most pertinent to the prediction (e.g., homophily), and does so more effectively than alternative approaches. Finally, we also show that CAR enhances interpretability of attention weights by accentuating node-neighbor relations that point to causal hypotheses. For social media network-sized graphs, a CAR-guided graph rewiring approach could allow us to combine the scalability of graph convolutional methods with the higher performance of graph attention.

1 INTRODUCTION

Graphs encode rich relational information that can be leveraged in learning tasks across a wide variety of domains. Graph neural networks (GNNs) can learn powerful node, edge or graph-level representations by aggregating a node's representations with that of its neighbors. The specifics of a GNN's neighborhood aggregation scheme are critical to its effectiveness on a prediction task. For instance, graph convolutional networks (GCNs) aggregate information via a simple averaging or max-pooling of neighbor features. While GCNs scales well, they are also indiscriminate. GCNs are prone to suffer in many real-world scenarios where uninformative or noisy connections exist between nodes (Kipf & Welling, 2017; Hamilton et al., 2017). GCNs may also over-smooth, with nodes in a graph taking on indistinguishable representations due to the indiscriminate aggregation operation (Li et al., 2018; Chen et al., 2020; Topping et al., 2022).

Graph-based attention mechanisms combat these issues by quantifying the relevance of node-neighbor relations and softly selecting neighbors in the aggregation step accordingly (Velickovic et al., 2018; Brody et al., 2022; Shi et al., 2021). This process of attending to select neighbors has contributed to significant performance gains for GNNs across a variety of tasks (Zhou et al., 2018; Veličković, 2022). Similar to the use of attention in natural language processing and computer vision, attention in graph settings also enables the interpretability of model predictions via the examination of attention weights (Serrano & Smith, 2019; Knyazev et al., 2019).

Paramount to the effectiveness of attention mechanisms in GNNs is their ability to discern the specific graph connectivity patterns that underlie accurate, generalizable predictions. Attention mechanisms in GNNs can be sensitive to spurious edges and correlations that mislead them in how they attend to node neighbors, which manifests as a failure to generalize to unseen data (Knyazev et al., 2019). One approach to improve the GNN's generalizability is to regularize the attention weights so they become more robust to spurious correlations/connections in the training data. Previous work along this direction has focused on L_0 regularization of attention weights to enforce sparsity (Ye & Ji, 2021) or has co-optimized a link prediction side-task using the attention mechanism(Kim & Oh, 2021). Since these regularization strategies are formulated independently of the actual prediction

task, they align the attention mechanism with some intrinsic property of the input graph without regard for the training objective.

We take a different approach. We begin our study with the question: "What is the importance of a specific edge to the prediction task?" Our answer comes from the perspective of regularization: we introduce CAR, a causal attention regularization framework that is suitable for any graph attention network architecture (**Figure** 1). Intuitively, an edge in the input graph is important to a prediction task if removing it leads to substantial degradation in the prediction performance of the GNN. The key conceptual advance of this work is to scalably leverage active interventions on node neighborhoods (i.e., deletion of specific edges) to align graph attention training with the causal impact of these interventions on task performance. While some efforts have previously been made to infuse notions of causality into GNNs, these causal approaches have been largely limited to using causal effects from pre-trained models as features for a separate model (Feng et al., 2021; Knyazev et al., 2019) or decoupling causal from non-causal effects (Sui et al., 2021). Causality is a frequently misused term, and CAR may not uncover rigorously causal mechanisms. Nonetheless, CAR is strongly influenced by the invariant prediction (Peters et al., 2016) framework for causal inference.

Our work builds on previous findings (Knyazev et al., 2019) that an explicit focus on improving attention weights can pay rich dividends. We apply CAR on three graph attention architectures across eight node classification tasks each, finding that it consistently improves test loss and accuracy. Our causal regularization fine-tunes attention weights by aligning them with task-specific homophily. In contrast, a non-causal label-propagation approach can also pick up on homophily but is task-agnostic and generalizes less well. On the ogbn-arxiv network, we investigate the citations up/down-weighted by CAR and found them to broadly group into three intuitive themes. Our causal approach can thus enhance the interpretability of attention weights, and we provide a qualitative analysis of this improved interpretability. We also present preliminary results demonstrating the applicability of CAR to graph property prediction and graph pruning tasks. Due to the size of industrially relevant graphs, it is common to use GCNs or sampling-based approaches on them. There, using attention weights learned by CAR on sampled subnetworks may guide graph rewiring of the full network to improve the results obtained with convolutional techniques.

2 Methods

2.1 Graph Attention Networks

Attention mechanisms have been effectively used across many domains by enabling models to dynamically attend to the specific parts of an input that are most relevant to a prediction task (Chaudhari et al., 2021). In graph settings, attention mechanisms compute the relevance of node-neighbor relations, as denoted by the edges of the graph. The neighbor aggregation operator in GNNs then uses this information to weight the contribution of each edge in the prediction task.

Across the multitude of graph attention mechanisms that have been proposed (Lee et al., 2019), the procedures for computing attention share a general form (Velickovic et al., 2018). A graph attention layer takes as input a set of node features $\mathbf{h} = \{\vec{h}_1, ..., \vec{h}_N\}$, $\vec{h}_i \in \mathbb{R}^F$, where N is the number of nodes and F is the dimensionality of the node features. From these node features, the graph attention layer in a GAT computes unnormalized attention weights for each edge (i, j)

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j), \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{F' \times F}$ linearly projects the input features into a new feature space and the attention mechanism function $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$ uses these new node features to compute an attention weight for each edge. These attention weights are typically normalized with respect to the 1-hop neighborhood of each node using the softmax function, yielding normalized attention coefficients α_{ij} that indicate the importance of node i's input features to node j. For a node j, these normalized attention weights are then used to compute a linear combination of its neighbors' features.

$$\vec{h}'_j = \sum_{i \in N_j} \alpha_{ij} \mathbf{W}\vec{h}_i, \;\; s.t. \sum_{i \in N_j} \alpha_{ij} = 1 \tag{2}$$

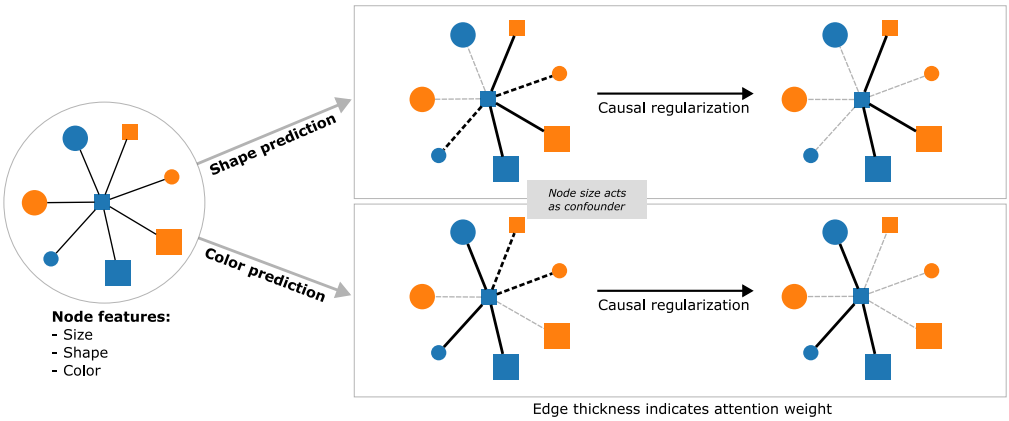


Figure 1: Schematic of CAR: Graph attention networks learn the relative importance of each node-neighbor for a given prediction task. However, their inferred attention weights can be miscalibrated due to noise, spurious correlations, or confounders (e.g., node size here). Our causal approach directly intervenes on a sampled subset of edges and supervises an auxiliary task that aligns an edge's causal importance to the task with its attention weight.

To employ multi-headed attention with K heads, each of the K attention mechanisms first independently calculates its respective normalized weights $\alpha_{i,j}^{(k)}$ using its own attention mechanism function $a^{(k)}(\cdot)$ and linear projection weights $\mathbf{W}^{(k)}$. The head-specific node outputs are then averaged.

In this paper, we focus on three widely used graph attention architectures: the original graph attention network (GAT) (Velickovic et al., 2018), a modified version of this original network (GATv2) (Brody et al., 2022), and the Graph Transformer network (Shi et al., 2021). The three architectures and their equations for computing attention are presented in **Table** 1. Here, $||$ denotes concatenation.

Table 1: Attention weight calculation across graph attention architecture variants

GAT (Velickovic et al., 2018)	$e_{ij} = \text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j)]\right)$
GATv2 (Brody et al., 2022)	$e_{ij} = \vec{\mathbf{a}}^T \text{LeakyReLU}\left(\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j\right)$
Graph Transformer (Shi et al., 2021)	$e_{ij} = \dfrac{(\mathbf{W}\vec{h}_i)^T(\mathbf{W}\vec{h}_j)}{\sqrt{F'}}$

2.2 Causal Attention Regularization Scheme

Causal attention objective We leverage the intuition that relevant node-neighbor relations for a particular task should not only be assigned high attention weights but also be important to the predictive accuracy of the model (**Figure** 1). When removed from the graph, these relevant edges should cause the predictions that rely on them to substantially worsen. We formalize this intuition by devising an auxiliary supervised prediction task that aims to align the attention weight for an edge with the causal effect of removing that edge as per the following loss function:

$$\mathcal{L}^c = \frac{1}{R}\sum_{r=1}^{R}\left(\frac{1}{\left|S^{(r)}\right|}\sum_{(n,i,j)\in S^{(r)}} \ell^c\left(\alpha_{ij}^{(n)}, c_{ij}^{(n)}\right)\right) \tag{3}$$

Here, n represents a single entity for which we aim to make a prediction. For a node prediction task, the entity n corresponds to a node, and in a graph prediction task, n corresponds to a graph. In each batch, we perform R separate rounds of causal effect calculations, each of which consists of a set of edge interventions $S^{(r)}$, $r = 1, \ldots, R$. Throughout this paper, we assume that all edges are directed and, if necessary, decompose an undirected edge into two directed edges. Each edge intervention in $S^{(r)}$ is represented by a set of tuples (n, i, j) which denote a selected edge (i, j) for an entity n in round r. Note that in a node classification task, n is the same as j (i.e., the node with the incoming edge). The causal effect $c_{ij}^{(n)}$ (see below) scores the impact on predictive performance of deleting

the edge (i, j). This is compared to the edge's attention weight $\alpha_{ij}^{(n)}$ via the loss function $\ell^c(\cdot, \cdot)$. For models with multiple attention heads, $\alpha_{ij}^{(n)}$ is the mean attention weight across the heads.

We regularize the attention mechanism in accordance with the learned causal effects from edge interventions. Thus, our full objective function $\mathcal{L}$ includes the loss associated with the prediction $\mathcal{L}^p$, the loss associated with causal attention task $\mathcal{L}^c$, and a causal regularization strength hyperparameter λ that mediates the contribution of the causal attention task to the objective.

$$\mathcal{L} = \mathcal{L}^p + \lambda \mathcal{L}^c \tag{4}$$

Edge intervention procedure We sample a set of edges in each round such that the prediction for each entity will strictly be affected by at most one edge intervention in that round. For example, in a node classification task for a model with one GNN layer, a round of edge interventions entails removing only a single incoming edge for each node being classified. In the graph property prediction case, only one edge will be removed from each graph per round. Because a model with one GNN layer only aggregates information over a 1-hop neighborhood, the removal of each edge will only affect the model's prediction for that edge's target node. To select a set of edges in the L-layer GNN case, edges are sampled from the 1-hop neighborhood of each node being classified, and sampled edges that lie in more than one target nodes' L-hop neighborhood are removed from consideration as intervention candidates. We note that in practice GNNs usually have $L \leq 2$ layers since greater depth increases the computational cost and the risk of oversmoothing (Li et al., 2018; Chen et al., 2020; Topping et al., 2022).

This edge intervention selection procedure is crucial, as it enables the causal effect of each selected intervention $c_{ij}^{(n)}$ on an entity n to be calculated independently of other interventions. By selecting only one intervention per entity within each round, we can parallelize the computation of these causal effects across all entities per round, aiding scalability.

Calculating task-specific causal effects We quantify the causal effect of an intervention at edge (i, j) on entity n as the relative change in the prediction loss for entity n upon removal of that edge.

$$c_{ij}^{(n)} = \sigma\left(\left(\rho_{ij}^{(n)}\right)^{d(n)} - 1\right) \quad \text{where} \quad \rho_{ij}^{(n)} = \frac{\ell^p\left(\hat{y}_{\setminus ij}^{(n)}, y^{(n)}\right)}{\ell^p\left(\hat{y}^{(n)}, y^{(n)}\right)} \tag{5}$$

Here, $\ell^p(\cdot, \cdot)$ corresponds to the loss function for the given prediction task, $\hat{y}^{(n)}$ is the prediction for entity n, $\hat{y}_{\setminus ij}^{(n)}$ is the prediction for entity n upon removal of edge (i, j), and $y^{(n)}$ is the ground truth value for entity n. Here, $d(n)$ represents the node degree for node classification, while it represents the number of edges in a graph for graph classification. Since attention weights for each node sum to 1, per-edge attention weights are expected to be smaller for higher-degree nodes. Exponentiating the relative intervention effect $\rho_{ij}^{(n)}$ by $d(n)$ is intended to adjust for this. The link function $\sigma(\cdot)$ maps its input to the support of the distribution of attention weights. The predictions $\hat{y}^{(n)}$ and $\hat{y}_{\setminus ij}^{(n)}$ are generated from the graph attention network being trained with CAR. We emphasize that they are both computed during the same training run, rather than in two separate runs.

Scalability CAR increases the computational cost of training in two ways: (i) additional evaluations of the loss function, and (ii) searches to ensure independent interventions. CAR performs $\mathcal{O}(R|V|)$ interventions, where R is the number of interventions per vertex, and $|V|$ is the cardinality of the vertex set. Each intervention requires a L-layer deep BFS, with complexity $\mathcal{O}(b^L)$, where L is the depth of the GNN, and b is the mean in-degree. While this carries a worst case time-complexity of $\mathcal{O}(R|V|b^L)$, in practice, we are able to mitigate much of the computational cost by parallelizing the intervention-overlap checks using the edge selection procedure above. Furthermore, the identification of non-overlapping interventions can also be done as a preprocessing step. In summary, we have found CAR to have a modest effect on the scalability of graph attention methods, with training runtimes that are only increased by 1.5-2 fold (**Appendix** A.3).

3 RELATED WORK

The performance gains associated with graph attention networks have led to a number of efforts to enhance and better understand graph attention mechanisms (Lee et al., 2019). One category of methods aims to improve the expressive power of graph attention mechanisms by supplementing a given prediction objective with a supervised attention mechanism (Feng et al., 2021), or a self-supervised connectivity prediction approach (Kim & Oh, 2021). A related set of methods leverage signals from interventions on graphs from a causal perspective to aid in training GNNs. One class of techniques performs interventions on the nodes (Knyazev et al., 2019; Feng et al., 2021). Meanwhile, other methods such as CAL (Sui et al., 2021) or CFLP (Zhao et al., 2021) augment the loss by adding additional tasks intended to separate causal and non-causal effects.

Our intervention-oriented framework can also be understood as a graph structure perturbation method. Perturbation methods can be broadly split into three different categories: graph data augmentation (Zhao et al., 2022), structural graph rewiring (Rong et al., 2020), and geometric graph rewiring (Topping et al., 2022). Inspired by the success of data augmentation approaches in computer vision, graph data augmentation methods seek to generate new training samples through different augmentation techniques. One of the earliest methods is DropEdge (Rong et al., 2020) reduces overfitting by randomly selecting edges from a uniform distribution to delete . Other methods build on DropEdge and select edges according to additional constraints including geometrical invariants (Gao et al., 2021), target probability distributions (Park et al., 2021), and information criteria (Suresh et al., 2021). Structural graph rewiring instead seeks to enforce structural priors such as sparsity or homophily during the graph alteration phase. Examples of these rewiring priors include fairness (Kose & Shen, 2022; Spinelli et al., 2022), temporal structure (Wang et al., 2021), predicted homophily (Chen et al., 2020), sparsity(Jin et al., 2020; Zheng et al., 2020), or information transfer efficiency (Klicpera et al., 2019). Geometric approaches, instead, choose to view the graph as a discrete geometry and alter the connectivity according to the balanced Forman curvature (Topping et al., 2022), the stochastic discrete Ricci flows (Bober et al., 2022), commute times (Arnaiz-Rodríguez et al., 2022), or algebraic connectivity(Arnaiz-Rodríguez et al., 2022). All of these approaches are designed either for use either in self-supervised learning or in a task-agnostic fashion, and consider the input graph independently of the task at hand.

To summarize, CAR introduces a combination of advances not previously reported: applicability to diverse attention architectures; task-based supervised regularization (rather than task-agnostic or self-supervised regularization) that leads to improved generalization; and a causal approach that directly relates an edge's importance to its attention weight, enhancing interpretability.

4 RESULTS

4.1 EXPERIMENTAL SETUP

We assessed the effectiveness of CAR across a diverse set of prediction tasks, datasets, graph attention mechanisms, and hyperparameter choices. We evaluated CAR on 8 node classification tasks (also see additional applications in §4.3). We used a simple network architecture, choosing $L = 1$ or $L = 2$ graph attention layers positioned between single-layer feed-forward networks (**Appendix** A.2). For each model, the PyTorch Geometric (Fey & Lenssen, 2019) implementation of one of GAT, GATv2, or Graph Transformer was used as the graph attention mechanism (Velickovic et al., 2018; Brody et al., 2022; Shi et al., 2021).

To establish CAR's robustness to hyperparameter choices, we evaluated CAR across a broad range of hyperparameter settings, rather than selecting specific hyperparameters for each model configuration. The number of attention heads was selected from $k = \{1, 3, 5\}$; the regularization strength was from $\lambda = \{0.1, 0.5, 1, 5\}$; and the number of hidden features used across layers were from $F' = \{10, 25, 100, 200\}$. See **Appendix** A.2 for details on the network architecture, hyperparameters, and training configurations.

We used cross-entropy loss for the prediction loss $\ell^p(\cdot, \cdot)$ and binary cross-entropy loss for the causal regularization loss $\ell^c(\cdot, \cdot)$. The link function $\sigma(\cdot)$ was chosen to be the sigmoid function with temperature $T = 0.1$. Unless otherwise specified, we performed $R = 5$ rounds of edge interventions per mini-batch when training with CAR.

4.2 NODE CLASSIFICATION

Datasets and evaluation: We used a total of 8 real-world node classification datasets of varying sizes and degrees of homophily: Cora, CiteSeer, PubMed, ogbn-arxiv, Chameleon, Squirrel, Cornell and Wisconsin. Each model was evaluated according to its accuracy on a held-out test set. We also evaluated the test cross-entropy loss, as it accounts for the full distribution of per-class predictions rather than just the highest-valued prediction considered in accuracy calculations. See **Appendix** A.1 for details on dataset statistics, splits, and references. Runtimes for training models with CAR ranged from roughly 2 seconds to 5 minutes on the evaluated datasets, with CAR's runtime only 1.5-2 times that of standard training on average (see **Appendix** A.3 for details).

Generalization performance: We compared both the test accuracy and the test loss of each model when trained with and without CAR. Across all model architecture and hyperparameter choices for a dataset, we applied the one-tailed paired Wilcoxon signed-rank test to quantify the overall outperformance of models trained with CAR against models trained without it. CAR resulted in higher test accuracy in 7 of the 8 node classification datasets and a lower test loss in all 8 datasets (**Figure** 2, $p < 0.05$). We also report the relative performance when averaging over all hyperparameter choices in **Table** 2 (test loss) and **Table** 11 (test accuracy, in Appendix A.6). These results highlight the consistency of performance gains achieved with CAR across the space of graph attention mechanisms and hyperparameters evaluated.

Table 2: Test loss on 8 node classification datasets

	Cora	CiteSeer	PubMed	ogbn-arxiv	Chameleon	Squirrel	Cornell	Wisconsin
GAT	2.33 ± 0.75	3.68 ± 2.64	1.05 ± 0.24	1.49 ± 0.02	1.29 ± 0.03	1.48 ± 0.02	1.19 ± 0.23	0.88 ± 0.11
GAT + CAR	$\mathbf{1.82 \pm 0.77}$	$\mathbf{3.06 \pm 2.34}$	$\mathbf{0.89 \pm 0.18}$	1.49 ± 0.02	$\mathbf{1.26 \pm 0.03}$	$\mathbf{1.47 \pm 0.02}$	$\mathbf{1.14 \pm 0.17}$	$\mathbf{0.84 \pm 0.13}$
GATv2	2.26 ± 0.54	3.08 ± 1.06	1.07 ± 0.29	1.48 ± 0.02	1.28 ± 0.03	1.48 ± 0.02	1.09 ± 0.11	0.88 ± 0.15
GATv2 + CAR	$\mathbf{1.63 \pm 0.59}$	$\mathbf{3.04 \pm 2.07}$	$\mathbf{0.90 \pm 0.17}$	$\mathbf{1.47 \pm 0.03}$	$\mathbf{1.24 \pm 0.03}$	$\mathbf{1.46 \pm 0.02}$	$\mathbf{1.04 \pm 0.10}$	$\mathbf{0.81 \pm 0.09}$
Transformer	3.43 ± 1.61	9.78 ± 8.27	1.79 ± 0.88	1.50 ± 0.03	1.38 ± 0.06	1.50 ± 0.02	1.22 ± 0.25	1.03 ± 0.28
Transf. + CAR	$\mathbf{1.71 \pm 0.50}$	$\mathbf{3.92 \pm 2.25}$	$\mathbf{1.60 \pm 0.59}$	1.50 ± 0.03	$\mathbf{1.33 \pm 0.04}$	$\mathbf{1.48 \pm 0.03}$	$\mathbf{1.07 \pm 0.12}$	$\mathbf{0.86 \pm 0.13}$

4.3 MODEL INVESTIGATION AND ABLATION STUDIES

Impact of regularization strength We explored 4 settings of λ: $\{0.1, 0.5, 1, 5\}$. For 6 of the 8 node classification datasets, CAR models trained with the higher causal regularization strengths ($\lambda = 1,\ 5$) demonstrated significantly larger reductions in test loss ($p < 0.05$, one-tailed Welch's t-test) compared to those trained with weaker regularization ($\lambda = 0.1,\ 0.5$). Notably, all four of the datasets with lower homophily (Chamelon, Squirrel, Cornell and Wisconsin) displayed significantly larger reductions with the higher regularization strengths, suggesting that stronger regularization may contribute to improved generalization in such settings (**Appendix** A.5).

Connection to homophily For the node classification task, the relevant neighbors to a particular node are expected to be those that share the same label as that node. We assessed if the graph attention mechanisms in models trained with CAR more effectively prioritized such relevant edges. For each edge, we used the label agreement between nodes connected by it as a proxy for the edge's ground-truth relevance: specifically, we assigned a reference attention weight ℓ_{ij} to an edge based on the label agreement in the target node's neighborhood (Kim & Oh, 2021).

$$\ell_{ij} = \hat{\ell}_{ij} / \sum_{k \in N_j} \hat{\ell}_{kj} \tag{6}$$

Here, $\hat{\ell}_{ij} = 1$ if nodes i and j have the same label $\hat{\ell}_{ij} = 0$ otherwise. N_j denotes the in-neighbors of node j. We then calculate the KL divergence of an edge's attention weight $\alpha_{i,j}$ from its reference attention weight ℓ_{ij} and summarize a model's ability to identify relevant edges as the mean of these KL divergence values across the edges in the held-out test set.

We compared these mean KL divergences between baseline models trained without CAR and models trained with CAR across the same broad range of model architecture and hyperparameter choices described above. We found that CAR-trained models consistently yielded lower mean KL divergence values than models trained without CAR for 6 of 8 node classification datasets (**Figure** 3,$p < 0.05$, one-tailed paired Wilcoxon signed-rank test).

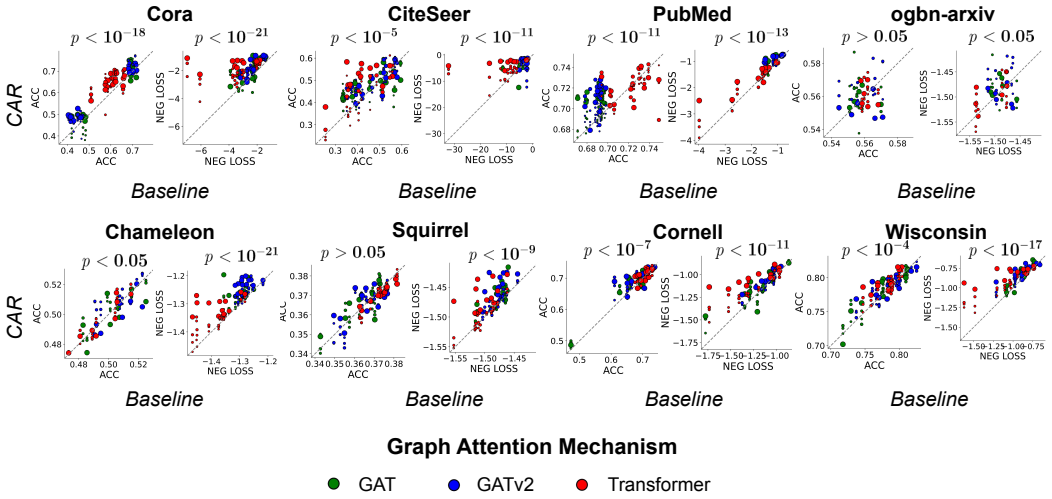


Figure 2: Test accuracy and negative loss on 8 node classification datasets. Each point corresponds to a comparison between a baseline model trained without CAR and an identical model trained with CAR. The point size represents the magnitude of the λ value chosen for the CAR-trained model. p-values are computed from one-tailed paired Wilcoxon signed-rank tests evaluating the improvement of CAR-trained models over the baseline models.

Notably, this enhanced prioritization of relevant edges was achieved without explicitly optimizing label agreement during training and is an inherent manifestation of aligning attention with the node classification tasks' causal effects. Thus, these results demonstrate that CAR not only improves performance on node classification tasks but also more accurately prioritizes the edges that are most relevant to the task at hand.

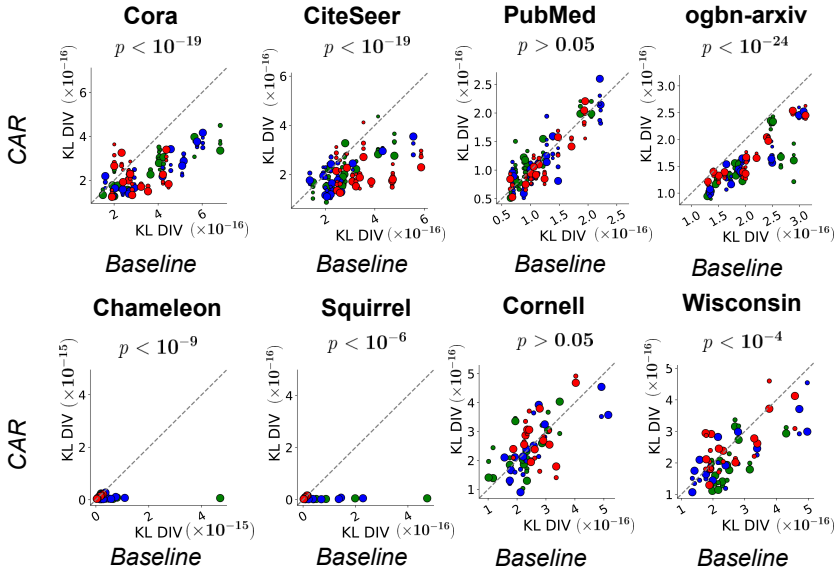


Figure 3: Coherence between attention weights and label agreement for CAR-trained models, compared to baseline. Lower KL divergence implies greater coherence. Point colors and sizes have the same meaning as in Figure 2. p-values are computed from one-tailed paired Wilcoxon signed-rank tests evaluating the improvement of CAR-trained models over the baseline models.

Comparison to neighbor voting-based regularization We wondered if the performance gains by our causal approach could be achieved by a non-causal approach that simply aligns the attention weights with a generic measure of homophily. To evaluate this, we employed a similar regularization scheme to that of CAR (**Equation** 5) but decoupled the causal effect calculation from the specific

network being trained. Instead of using predictions from the network to determine the causal effect, we used predictions from a neighbor voting model, in which each node's classification prediction is a histogram of class labels from the node's neighbors. In this neighbor voting regularization scheme, the attention mechanism for the network being trained is thus guided during training by the effects of edge interventions to this neighbor voting model's predictions rather than by the effects of edge interventions to the network's own predictions.

We compared CAR-trained models with those trained with neighbor voting regularization by evaluating the consistency of their test loss improvements relative to the baseline models without regularization. We used the one-tailed paired Wilcoxon signed-rank test to evaluate the significance of the regularized models' test loss improvements across the set of model architecture and hyperparameter choices, focusing on models trained with the higher $\lambda = 1$ or $\lambda = 5$ regularization strengths. Models trained when regularizing attention weights with the neighbor voting scheme underperformed those trained with CAR in 7 of the 8 datasets (**Table** 3). Interestingly, the neighbor voting-based regularization showed an overall gain in performance relative to the baseline models (i.e., those trained without any regularization), suggesting that even non-specific regularization can be somewhat useful for training attention. Overall, these results demonstrate the effectiveness of using CAR to improve generalization performance for many node classification tasks.

Table 3: Generalization performance of CAR-trained models with ablations compared to baseline ($-\log_{10}(p)$, one-tailed paired Wilcoxon signed-rank test)

	Cora	**CiteSeer**	**PubMed**	**ogbn-arxiv**	**Chameleon**	**Squirrel**	**Cornell**	**Wisconsin**
CAR	**12.36**	**5.92**	**10.51**	**1.81**	**12.77**	9.72	**6.72**	**9.34**
Neighbor Voting	10.04	5.59	7.30	0.53	12.34	**10.97**	4.15	8.20

Additional applications To explore the utility of CAR as a general-purpose framework, we employed CAR in two additional tasks: graph property prediction and graph pruning. For graph property prediction, we evaluated CAR on ogbg-molbace, a dataset for binary classification, and ogbg-molfreesolv, which features a regression task (Hu et al., 2020). We trained models with CAR across a broad range of graph attention mechanisms and hyperparameter choices. For both datasets, we found that CAR-trained models consistently outperformed those that were trained without CAR ($p < 0.05$, one-tailed paired Wilcoxon signed-rank test, see **Appendix** A.8 for details).

We also explored a graph pruning approach that uses attention weights from CAR-trained models to remove task-irrelevant edges. In particular, we used this edge pruning procedure as a pre-processing step for training and inference with GCNs, which are more scalable than graph attention networks (Rossi et al., 2020) but indiscriminate in how they aggregate information over node neighborhoods. We found that using CAR-guided pruning was able to improve the test accuracy of GCNs not only relative to GCNs trained and evaluated over the full graph but also outperformed GCNs trained and evaluated with non-regularized (i.e. baseline) graph attention mechanisms (**Appendix** A.9). These preliminary results open the door for further exploration of CAR's applicability on these tasks.

4.4 Interpreting Attention Weights: Qualitative Analysis

In addition to providing a robust way to increase both model quality and generalization, CAR learns attention weights that are more interpretable in a *post hoc* analysis. Here, we explored the edge-wise difference of the attention values between our method and a baseline GAT applied to the ogbn-arxiv dataset. In this dataset, nodes represent computer science arXiv papers, edges are citation links between these papers, and the prediction task is to classify papers into their subject areas. We manually reviewed citations that were up/down-weighted by CAR-trained models and observed that these citations broadly fit into one of three categories: (i) down-weighting self-citations, (ii) down-weighting popular "anchor" papers, or (iii) upweighting topically-narrow papers with few citations.

In the first case, we found that CAR down-weights edges associated with self-citations (see **Table** 4 for examples). The causal story here is quite clear— machine learning is a fast-moving field with authors moving into the field and changing specialties as those specialties are born. Because of this, the narrative arc that a set of authors constructs to present this idea can include citations to their own previous work from different subfields. While these citations do help situate the work within the

broader literature and can provide background that readers might find valuable, they are not relevant to the subject area prediction task.

Table 4: Down-weighted self-citations.

Paper Title	Cited Title (from a different subject area)	Δ
Viterbinet a Deep Learning Based Viterbi Algorithm for Symbol Detection	*Frequency Shift Filtering for Ofdm Signal Recovery in Narrowband Power Line Communications*	-89.91 %
Solving Underdetermined Systems with Error Correcting Codes	*Systems of MDS Codes from Units and Idempotents*	-89.80 %
Performance Analysis for Multichannel Reception of OOFSK Signaling	*On-Off Frequency-Shift Keying for Wideband Fading Channels*	-81.03 %

In the second case, we found that CAR down-weights edges directed towards popular or otherwise seminal "anchor" papers (**Table** 5). These papers tends to be included in introductions to provide references for common concepts or methods, such as Adam, ResNet, and ImageNet. They are also widely cited across subject areas and hence have little bearing on the subject area prediction task.

Notably, CAR does not simply learn to ignore edges from high-degree nodes. In the example of the Word2Vec paper, we observed notable increases in attention weights for edges connecting it to multiple highly related papers, including a 2.5×10^6 % increase for *Efficient Graph Computation for Node2Vec* and a 2.0×10^6 % increase for *Multi-Dimensional Explanation of Reviews* .

Table 5: Down-weighted citations of "anchor" papers. Each of the cited works that is down-weighted is a well known ML paper.

Paper Title	Cited Title	Δ
Reservoir Computing Hardware with Cellular Automata	*Adam: A Method for Stochastic Optimization*	-99.87%
Quantization Networks	*Deep Residual Learning for Image Recognition*	-99.86%
Compressive Hyperspherical Energy Minimization	*ImageNet Large Scale Visual Recognition Challenge*	-99.75%

In the final case, we observed that CAR up-weighted edges directed towards deeply related but largely unnoticed papers (**Table** 12). In our manual exploration of the data, we observed that these papers are those that are close to the proposed method. These papers are the type that are often found only after a thorough literature review. Clearly, such edges should play a key role in predicting a paper's topic and should be up-weighted.

5 Conclusion

We introduced CAR, a causally-inspired regularization scheme that can be applied to any attention based graph architecture. Applying it to both homophilic and heterophilic node-classification tasks, we found accuracy improvements and loss reductions in almost all circumstances. We performed ablation studies for a deeper understanding, and found that CAR aligns attention with per-task homophily and does so better than a neighbor voting-based regularizer. A qualitative review also suggested that the attention-weight changes produced by CAR are intuitive and interpretable.

Understanding how, and improving what, GNNs learn remains a major open problem and an active area of research. Towards this, our work bridges two families of techniques: attention regularization and causal interventions. We believe this synthesis could be a promising direction of future research. Also, while our work is causally inspired, we do not currently formalize the causal foundations of our method. Future work may benefit from utilizing the tools of causal inference theory, like do-calculus (Pearl, 2012), for selecting and analyzing edge interventions. Lastly, while our graph pruning results are preliminary, they also suggest a promising direction for future work on scaling CAR-based insights to web-scale graphs.

6 REPRODUCIBILITY STATEMENT

To ensure the reproducibility of the results in this paper, we have included the source code for our method as supplementary materials. The datasets used in this paper are all publicly available, and we also use the publicly available train/validation/test splits for these datasets. We provide details on these datasets in the Appendix and have provided references to them in both the main text and the Appendix. In addition, we have provided detailed descriptions of the experimental setup, model training schemes, model architecture design choices, and hyperparameter choices in the "Experimental Setup" section as well as in Appendix A.2.

REFERENCES

Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver. Diffwire: Inductive graph rewiring via the lovász bound, 2022.

Jakub Bober, Anthea Monod, Emil Saucan, and Kevin N. Webster. Rewiring networks for graph neural network training using discrete geometry, 2022.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=F72ximsx7C1.

Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An attentive survey of attention models. *ACM Trans. Intell. Syst. Technol.*, 12(5):53:1–53:32, 2021. doi: 10.1145/3465055. URL https://doi.org/10.1145/3465055.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 3438–3445. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/5747.

Fuli Feng, Weiran Huang, Xiangnan He, Xin Xin, Qifan Wang, and Tat-Seng Chua. Should graph convolution trust neighbors? A simple causal inference method. In Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (eds.), *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pp. 1208–1218. ACM, 2021. doi: 10.1145/3404835.3462971. URL https://doi.org/10.1145/3404835.3462971.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *CoRR*, abs/1903.02428, 2019. URL http://arxiv.org/abs/1903.02428.

Zhan Gao, Subhrajit Bhattacharya, Leiming Zhang, Rick S Blum, Alejandro Ribeiro, and Brian M Sadler. Training robust graph neural networks with topology adaptive edge dropping. *arXiv preprint arXiv:2106.02892*, 2021.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual*

Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '20, pp. 66–74, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403049. URL https://doi.org/10.1145/3394486.3403049.

Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Wi5KUNlqWty.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. *Diffusion Improves Graph Learning*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. Understanding attention and generalization in graph neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 4204–4214, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

O. Deniz Kose and Yanning Shen. Fair node representation learning via adaptive data augmentation, 2022.

John Boaz Lee, Ryan A. Rossi, Sungchul Kim, Nesreen K. Ahmed, and Eunyee Koh. Attention models in graphs: A survey. *ACM Trans. Knowl. Discov. Data*, 13(6):62:1–62:25, 2019. doi: 10.1145/3363574. URL https://doi.org/10.1145/3363574.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3538–3545. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16098.

Hyeonjin Park, Seunghun Lee, Sihyeon Kim, Jinyoung Park, Jisu Jeong, Kyung-Min Kim, Jung-Woo Ha, and Hyunwoo J Kim. Metropolis-hastings data augmentation for graph neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 19010–19020. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/9e7ba617ad9e69b39bd0c29335b79629-Paper.pdf.

Judea Pearl. The do-calculus revisited, 2012. URL https://arxiv.org/abs/1210.4852.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society Series B*, 78(5):947–1012, November 2016. URL https://ideas.repec.org/a/bla/jorssb/v78y2016i5p947-1012.html.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Hkx1qkrKPr.

Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael M. Bronstein, and Federico Monti. SIGN: scalable inception graph neural networks. *CoRR*, abs/2004.11198, 2020. URL https://arxiv.org/abs/2004.11198.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *J. Complex Networks*, 9(2), 2021. doi: 10.1093/comnet/cnab014. URL https://doi.org/10.1093/comnet/cnab014.

Sofia Serrano and Noah A. Smith. Is attention interpretable? In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 2931–2951. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1282. URL https://doi.org/10.18653/v1/p19-1282.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pp. 1548–1554. ijcai.org, 2021. doi: 10.24963/ijcai.2021/214. URL https://doi.org/10.24963/ijcai.2021/214.

Indro Spinelli, Simone Scardapane, Amir Hussain, and Aurelio Uncini. Fairdrop: Biased edge dropout for enhancing fairness in graph representation learning. *IEEE Transactions on Artificial Intelligence*, 3(3):344–354, 2022. doi: 10.1109/TAI.2021.3133818.

Yongduo Sui, Xiang Wang, Jiancan Wu, Min Lin, Xiangnan He, and Tat-Seng Chua. Causal attention for interpretable and generalizable graph classification. 2021. doi: 10.1145/3534678.3539366.

Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=ioyq7NsR1KJ.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7UmjRGzp-A.

Petar Veličković. Message passing all the way up. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022. URL https://openreview.net/forum?id=Bc8GiEZkTe5.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Yiwei Wang, Yujun Cai, Yuxuan Liang, Henghui Ding, Changhu Wang, Siddharth Bhatia, and Bryan Hooi. Adaptive data augmentation on temporal graphs. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 1440–1452. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/0b0b0994d12ad343511adfbfc364256e-Paper.pdf.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay S. Pande. Moleculenet: A benchmark for molecular machine learning. *CoRR*, abs/1703.00564, 2017. URL http://arxiv.org/abs/1703.00564.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 40–48. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/yanga16.html.

Yang Ye and Shihao Ji. Sparse graph attention networks. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Tong Zhao, Gang Liu, Daheng Wang, Wenhao Yu, and Meng Jiang. Learning from counterfactual links for link prediction, 2021.

Tong Zhao, Gang Liu, Stephan Günnemann, and Meng Jiang. Graph data augmentation for graph machine learning: A survey, 2022.

Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11458–11468. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/zheng20d.html`.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018. URL `http://arxiv.org/abs/1812.08434`.

A Appendix

A.1 Datasets

A.1.1 Node Classification Datasets

We provide overviews of the various node classification datasets along with accompanying statistics in Table 7. For all datasets, we use the publically available train/validation/test splits that accompany these datasets.

Planetoid: The Cora, CiteSeer, and PubMed datasets are citation networks from Yang et al. (2016). Nodes represent documents and directed edges represent citation links. Nodes are featurized as bag-of-word representations of their respective documents. The prediction task for this dataset is to classify a given paper into its respective subject area.

ogbn-arxiv: The ogbn-arxiv dataset is a citation network between computer science arXiv paper indexed by MAG (Hu et al., 2020). Nodes represent papers and a node's features are the mean embeddings of words in its corresponding paper's title and abstract. Edges are directed and represent a citation by one paper of another. The prediction task for this dataset is to predict the subject area of a given arXiv paper.

Wikipedia: The Chameleon and Squirrel datasets are Wikipedia networks from Rozemberczki et al. (2021), in which nodes represent web pages and edges represent hyperlinks between them. Nodes are featurized as bag-of-word representations of important nouns in their respective Wikipedia pages. Average monthly traffic of web pages are converted into categories, and the prediction task is to assign a given page to its corresponding category.

WebKB: The Cornell and Wisconsin datasets are networks of web pages from various computer science departments, in which nodes represent web pages and edges are hyperlinks between them. Node features are bag-of-word representations of their respective web pages, and the prediction task is to assign a given web page to the category that describes its content.

Table 6: Node Classification Dataset Statistics

Dataset	# classes	# nodes	# edges	# splits	Mean degree	Homophily
Cora	7	2,708	10,556	1	3.9	0.81
CiteSeer	6	3,327	9,104	1	2.7	0.74
PubMed	3	19,717	88,648	1	4.5	0.80
ogbn-arxiv	40	169,343	1,166,243	1	6.9	0.66
Chameleon	5	2,277	36,051	10	15.8	0.23
Squirrel	5	5,201	216,933	10	41.7	0.22
Cornell	5	183	295	10	1.6	0.12
Wisconsin	5	251	499	10	2.0	0.17

A.1.2 Graph Property Prediction Datasets

OGB: The ogbg-molbace and ogbg-molfreesolv datasets consist of molecular graphs from the MoleculeNet benchmark collection (Wu et al., 2017). The prediction task for the ogbg-molbace dataset is binary classification, for which the objective is to identify inhibitors of human β-secretase 1 (BACE-1). The ogbg-molfreesolv dataset features a regression task, whose objective is to predict the hydration free energy of small molecules in water. These datasets were obtained from OGB (Hu et al., 2020).

Table 7: Dataset Statistics

Dataset	# graphs	Mean # nodes	Mean # edges
ogbn-molbace	1,513	34.1	73.7
ogbn-molfreesolv	642	8.7	16.8

A.2 Graph Attention Network Architectures and Training

A.2.1 Node Classification Model Architecture

The GNN model used for node classification tasks takes as input the original node features $\vec{x}_i \in \mathbb{R}^d$ and applies a non-linear projection to these features to yield a set of hidden features $\vec{h}_i = \text{LeakyReLU}\big(\mathbf{W}_1\vec{x}_i + \mathbf{b}_1\big)$, where $\mathbf{W}_1 \in \mathbb{R}^{F\times d}$ and $\mathbf{b}_1 \in \mathbb{R}^F$. These hidden features are then passed through L graph attentional layers of the same chosen architecture, yielding new hidden features per node of the same dimensionality $\vec{h}'_i \in \mathbb{R}^F$. The pre- and post-graph attention layer hidden features are then concatenated $[\vec{h}_i||\vec{h}'_i]$, after which a final linear layer and softmax transformation $\sigma_{\text{softmax}}(\cdot)$ are applied to produce the prediction output $\hat{y}_i = \sigma_{\text{softmax}}\big(\mathbf{W}_2\text{LeakyReLU}\big([\vec{h}_i||\vec{h}'_i]\big) + \mathbf{b}_2\big)$. Here, $\hat{y}_i \in \mathbb{R}^C$, $\mathbf{W}_2 \in \mathbb{R}^{2F\times C}$, and $\mathbf{b}_2 \in \mathbb{R}^F$, where C is the number of classes in the classification task. Models were implemented in PyTorch and PyTorch Geometric (Fey & Lenssen, 2019). Self-loops were not included in the graph attention layers; otherwise, default PyTorch Geometric parameter settings were used for the graph attention layers.

A.2.2 Graph Property Prediction Model Architecture

The GNN model used for graph property prediction tasks uses the same architecture as the node classification model above to generate hidden node features. The original node features $\vec{x}_i \in \mathbb{R}^d$ are transformed into the hidden node features $[\vec{h}_i||\vec{h}'_i]$, composed of the pre- and post-graph attention layer hidden features. The hidden node features within each graph are then added together, after which the sum is passed through a LeakyReLU activation function and a final linear layer is applied. For binary classification tasks, a sigmoid function is applied to the output of this linear layer. For the evaluated datasets, the original node features here refer to trainable embeddings generated by the AtomEncoder embedding model implemented in Hu et al. (2020).

A.2.3 Training Details

All models were trained using the Adam optimizer with a learning rate of 0.01 and mini-batch size of 10,000. Each dataset was partitioned into training, validation, and test splits in line with previous work (**Appendix** A.1), and early stopping was applied during training with respect to the validation loss. Training was performed on a single NVIDIA Tesla T4 GPU.

A.3 RUNTIME STATISTICS

Table 8: Training times for node classification datasets

Dataset	# nodes (train set)	# edges (train set)	Train time w/o CAR (sec)	Train time w/ CAR (sec)
Cora	140	638	1.1 ± 0.4	2.2 ± 2.1
CiteSeer	120	364	1.3 ± 0.5	2.1 ± 1.7
PubMed	60	297	1.4 ± 0.7	2.3 ± 2.6
ogbn-arxiv	32,970	93,942	165 ± 78	286 ± 166
Chameleon	1,092	$17{,}157 \pm 1{,}013$	1.6 ± 0.3	2.6 ± 1.1
Squirrel	2,496	$105{,}517 \pm 3{,}062$	2.7 ± 0.6	4.0 ± 1.9
Cornell	87	148 ± 18	1.3 ± 1.2	1.4 ± 0.9
Wisconsin	120	239 ± 10	1.1 ± 0.4	1.1 ± 0.6

A.4 CHANGES IN TEST ACCURACY AND LOSS BY GRAPH ATTENTION MECHANISM AND REGULARIZATION STRENGTH

Table 9: Average percent change in test accuracy

λ Value	Cora			CiteSeer			PubMed		
	GAT	GATv2	Transformer	GAT	GATv2	Transformer	GAT	GATv2	Transformer
$\lambda \in \{1.0, 5.0\}$	**+3.8%**	**+7.4%**	+7.0%	**+7.4%**	**+7.1%**	**+12.5%**	**+2.7%**	+0.3%	0.0%
$\lambda \in \{0.1, 0.5\}$	+2.3%	+6.6%	**+7.6%**	+4.2%	+3.0%	+1.5%	+0.4%	+0.3%	0.0%

λ Value	ogbn-arxiv			Chameleon			Squirrel		
	GAT	GATv2	Transformer	GAT	GATv2	Transformer	GAT	GATv2	Transformer
$\lambda \in \{1.0, 5.0\}$	-0.9%	-0.6%	**-0.3%**	-0.2%	-1.1%	-0.2%	+0.3%	**+0.4%**	-0.5%
$\lambda \in \{0.1, 0.5\}$	**0%**	**-0.3%**	-0.7%	**+0.3%**	**+0.2%**	**0.0%**	**+0.5%**	-0.1%	**-0.4%**

λ Value	Cornell			Wisconsin		
	GAT	GATv2	Transformer	GAT	GATv2	Transformer
$\lambda \in \{1.0, 5.0\}$	**+1.8%**	**+3.1%**	+0.6%	**+0.6%**	**+0.4%**	**+1.2%**
$\lambda \in \{0.1, 0.5\}$	+1.4%	+1.6%	**+0.7%**	+0.4%	0.0%	+0.9%

Table 10: Average percent change in test loss

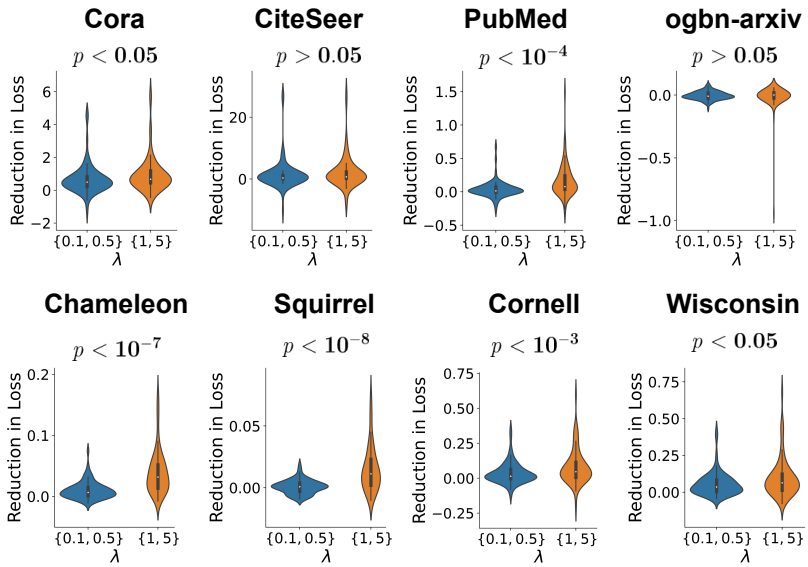
λ Value	Cora			CiteSeer			PubMed		
	GAT	GATv2	Transformer	GAT	GATv2	Transformer	GAT	GATv2	Transformer
$\lambda \in \{1.0, 5.0\}$	**-22.6%**	**-29.0%**	**-43.6%**	-11.7%	**+7.6%**	**-45.0%**	**-12.8%**	**-13.9%**	**-7.0%**
$\lambda \in \{0.1, 0.5\}$	-15.3%	-19.8%	-28.8%	**-12.9%**	+11.2%	-20.9%	-2.8%	-2.9%	-1.1%

λ Value	ogbn-arxiv			Chameleon			Squirrel		
	GAT	GATv2	Transformer	GAT	GATv2	Transformer	GAT	GATv2	Transformer
$\lambda \in \{1.0, 5.0\}$	+0.6%	+1.9%	-0.2%	**-2.4%**	**-2.7%**	**-3.8%**	**-1.1%**	**-1.5%**	**-2.0%**
$\lambda \in \{0.1, 0.5\}$	**-0.1%**	**-0.2%**	**-0.8%**	-0.7%	-0.7%	-0.8%	-0.2%	0.0%	0.0%

λ Value	Cornell			Wisconsin		
	GAT	GATv2	Transformer	GAT	GATv2	Transformer
$\lambda \in \{1.0, 5.0\}$	**-1.8%**	**-3.1%**	**-0.6%**	-0.6%	**-0.4%**	**-1.2%**
$\lambda \in \{0.1, 0.5\}$	-1.4%	-1.6%	**-0.7%**	-0.4%	0.0%	-0.9%

A.5 REGULARIZATION STRENGTH AND GENERALIZATION PERFORMANCE

Figure 4: Test loss reduction for CAR-trained models across regularization strengths. p-values are computed from one-tailed t-tests evaluating the significance of the test loss reductions for the $\lambda \in \{1, 5\}$ CAR-trained models being greater than those of the $\lambda \in \{0.1, 0.5\}$ CAR-trained models.



A.6 Test Accuracy on Node Classification Datasets

Table 11: Test accuracy on 8 node classification datasets

	Cora	CiteSeer	PubMed	ogbn-arxiv	Chameleon	Squirrel	Cornell	Wisconsin
GAT	0.58 ± 0.13	0.46 ± 0.09	0.69 ± 0.01	0.56 ± 0.01	0.50 ± 0.01	0.37 ± 0.01	0.66 ± 0.06	0.78 ± 0.03
GAT + CAR	$\mathbf{0.60 \pm 0.12}$	$\mathbf{0.49 \pm 0.06}$	$\mathbf{0.71 \pm 0.01}$	0.56 ± 0.01	0.50 ± 0.01	0.37 ± 0.01	$\mathbf{0.67 \pm 0.06}$	0.78 ± 0.03
GATv2	0.60 ± 0.14	0.47 ± 0.08	0.69 ± 0.01	0.56 ± 0.01	0.51 ± 0.01	0.37 ± 0.01	0.68 ± 0.04	0.79 ± 0.02
GATv2 + CAR	$\mathbf{0.61 \pm 0.12}$	$\mathbf{0.50 \pm 0.06}$	$\mathbf{0.71 \pm 0.01}$	0.56 ± 0.01	0.51 ± 0.01	0.37 ± 0.01	$\mathbf{0.70 \pm 0.03}$	0.79 ± 0.02
Transformer	0.60 ± 0.04	0.43 ± 0.09	0.72 ± 0.02	0.56 ± 0.01	0.50 ± 0.01	0.37 ± 0.01	0.70 ± 0.02	0.78 ± 0.02
Transformer + CAR	$\mathbf{0.65 \pm 0.04}$	$\mathbf{0.47 \pm 0.09}$	0.72 ± 0.01	0.56 ± 0.01	0.50 ± 0.01	0.37 ± 0.01	$\mathbf{0.71 \pm 0.02}$	$\mathbf{0.79 \pm 0.01}$

A.7 Explanability Tables

Table 12: Upweighted highly relevant citations.

Paper Title	Cited Title	Δ
Generalized Random Gilbert Varshamov Codes	*Expurgated Random Coding Ensembles Exponents Refinements and Connections*	7700 %
Sign Language Recognition Generation and Translation: an Interdisciplinary Perspective	*Swift a SignWriting Improved Fast Transcriber*	2470 %
Random Beamforming over Quasi-Static and Fading Channels: A Deterministic Equivalent Approach	*Optimal Selective Feedback Policies for Opportunistic Beamforming*	536 %

A.8 Graph Property Prediction

We evaluated CAR on the ogbg-molbace and ogbg-molfreesolv datasets, which feature a binary classification and a regression task, respectively. CAR was assessed across the same set of graph attention architectures and hyperparameters as in the node classification task. The dimensionality

of the AtomEncoder node embeddings was set to the chosen number of hidden features F' for each model. We used binary cross entropy and mean-square error as the prediction losses for ogbg-molbace and ogbg-molfreesolv, respectively. The remaining training configurations are identical to those used in the node classification tasks. To assess models, we used the standard evaluation metrics provided in OGB (Hu et al., 2020) for these datasets. The ogbg-molbace dataset was evaluated using the area under the receiver operating characteristic curve (ROCAUC), and the ogbg-molfreesolv was evaluated using the root-mean-square error (RMSE). These evaluation metrics were computed with respect to each dataset's held-out test set. We also assessed each model's performance by computing its prediction loss with respect to this held-out test set.

Figure 5: Evaluation metrics for graph property prediction datasets

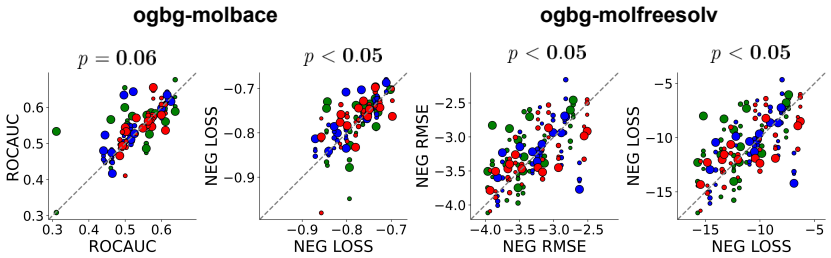


Table 13: Test loss on graph classification datasets

	ogbg-molbace (Eval: ROCAUC)	**ogbg-molbace (Loss: Cross Entropy)**	**ogbg-molfreesolv (Eval: RMSE)**	**ogbg-molfreesolv (Loss: MSE)**
GAT	0.53 ± 0.09	0.77 ± 0.05	3.3 ± 0.4	11.9 ± 2.6
GAT + CAR	$\mathbf{0.55 \pm 0.07}$	$\mathbf{0.76 \pm 0.05}$	$\mathbf{3.2 \pm 0.4}$	$\mathbf{10.3 \pm 2.8}$
GATv2	0.52 ± 0.06	0.78 ± 0.05	3.3 ± 0.4	$\mathbf{10.2 \pm 2.4}$
GATv2 + CAR	$\mathbf{0.54 \pm 0.07}$	$\mathbf{0.77 \pm 0.05}$	$\mathbf{3.2 \pm 0.4}$	10.3 ± 2.4
Transformer	0.55 ± 0.04	0.77 ± 0.04	3.2 ± 0.4	11.1 ± 3.0
Transformer + CAR	$\mathbf{0.56 \pm 0.05}$	$\mathbf{0.76 \pm 0.04}$	3.2 ± 0.4	11.1 ± 2.5

A.9 CAR-Guided Graph Rewiring

While graph attention networks have demonstrated notable performance gains, its inclusion of graph attention layers currently limits its use in large-scale applications compared to GCNs, for which a number of advances in scalability have been made (Rossi et al., 2020). To leverage the advantages of CAR in graph attention alongside the scalability of GCNs, we explored a graph rewiring approach based on CAR-guided edge pruning. For a given dataset, we first use a trained graph attention network to assign an attention weight for each edge in the training and validation sets, after which edges with attention weights below a threshold α_T are pruned. A GCN is then trained on the rewired training set with early stopping imposed with respect to the validation loss on the rewired validation set. The trained GCN is then evaluated on a similarly rewired test set. We use a similar network architecture for the GCN as the various graph attention networks described in **Appendix** A.2, with the graph attention layers replaced with graph convolutional layers. We set the number of hidden dimensions in the GCN models to be $F' = 100$.

For the Chameleon dataset, we identified the hyperparameter settings that contributed to the highest validation accuracy for each of the one-layer GAT, GATv2, and Transformer CAR-trained models. We then trained GCN models on graphs that are pruned based on each of these models' attention mechanisms. We also pruned graphs using the counterparts of these models that were trained without CAR and trained another set of GCN models on these pruned graphs. We evaluated the test accuracy of the GCN models when performing this procedure across various attention thresholds (**Figure** 6). We observed that training and evaluating GCN models on pruned graphs contributed to enhanced test accuracy compared to the baseline GCN models that were trained and evaluated on the original graph. Furthermore, we compared GCN models trained and evaluated on CAR-guided pruned graphs against similar GCN models trained and evaluated on graphs pruned without CAR by

computing the area under the curve (AUC) associated with the test accuracies at various attention thresholds. Each AUC was calculated as the area below its models' test accuracies line and above the baseline GCN models' test accuracy. CAR-guided graph pruning was associated with higher AUC values across the three graph attention mechanisms, demonstrating the potential for CAR's utility in graph pruning tasks.

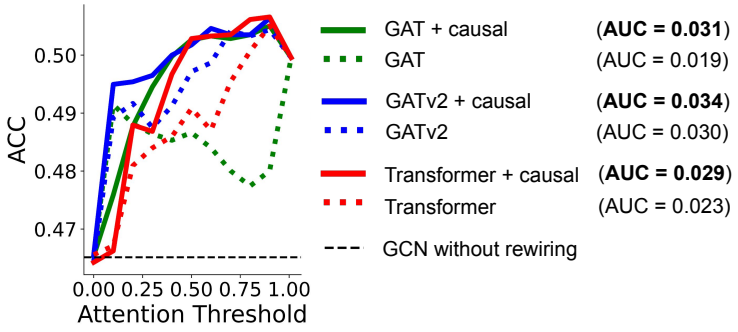



Figure 6: Test accuracy of models trained and evaluated on graphs rewired using graph attention