CLUSTER ATTENTION FOR GRAPH DEEP LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Message Passing Neural Networks have recently become the most popular approach to graph machine learning tasks; however, their receptive field is limited by the number of message passing layers. To increase the receptive field, Graph Transformers with global attention have been proposed; however, global attention does not take into account the graph topology and thus lacks graph-structure-based inductive biases, which are typically very important for graph machine learning tasks. In this work, we propose an alternative approach: cluster attention (CLATT). We divide graph nodes into clusters with off-the-shelf graph community detection algorithms and let each node attend to all other nodes in each cluster. CLATT provides large receptive fields while still having strong graph-structure-based inductive biases. We show that augmenting Message Passing Neural Networks or Graph Transformers with CLATT significantly improves their performance on a wide range of graph datasets including datasets from the recently introduced GraphLand benchmark representing real-world applications of graph machine learning.

1 INTRODUCTION

Graphs are a natural way to represent data from different domains such as social networks (both real-life and virtual), computer networks, transportation networks, co-purchasing networks, molecules, connectomes, neural network architectures, various physical and biological systems, or even interconnected abstract concepts. Thus, machine learning on graph-structured data, henceforth referred to as Graph Machine Learning (GML), has attracted a lot of attention recently. In particular, Graph Neural Networks (GNNs) (Duvenaud et al., 2015; Kipf & Welling, 2017; Gilmer et al., 2017; Hamilton et al., 2017) have become the most popular models for most GML tasks in the past decade. Most modern GNNs can be unified by the Message Passing Neural Networks (MPNNs) framework (Gilmer et al., 2017). In this framework, within each neural network layer each node in the graph sends messages to its neighbors and aggregates incoming messages to form its new representation based on them. The graph-structure based relational inductive biases of MPNNs, which send messages along graph edges, turned out to be very effective for modeling many real-world networks (Battaglia et al., 2018). However, in each MPNN layer, the model only exchanges information between neighboring nodes. Thus, the model's receptive field, i.e., the maximum shortest path distance information can travel between nodes in the graph, is equal to the number of message passing layers. This has been argued to be a limitation of MPNNs as it prevents taking into account long-range interaction between nodes that might happen in some networks. Thus, Graph Transformers with global (all-to-all) attention have been proposed as an alternative neural model architecture for GML (Kreuzer et al., 2021; Ying et al., 2021; Wu et al., 2022; Rampášek et al., 2022), inspired by the success of the Transformer architecture in the field of natural language processing (Vaswani et al., 2017). However, all-to-all attention does not have any information about the graph structure, which is arguably very important for solving GML tasks, and thus this information has to be integrated into the model in some other ways (typically via positional encodings or MPNN-Transformer hybridization).

It has been observed that many real-world networks exhibit community structure — presence of well-defined clusters of densely connected nodes with sparser connections between these clusters (Girvan & Newman, 2002). The problem of finding such clusters — graph clustering — has been extensively studied in network science and machine learning communities, and many methods for it have been proposed over the years (Fortunato, 2010). Such algorithms can be used to divide graph nodes into clusters where nodes within the same cluster are strongly related, where the exact definition of this relation and thus the meaning of the obtained clusters can differ between different clustering algorithms.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this work, we argue that various partitions of a graph into node clusters provide very useful information about the graph structure and thus it can be beneficial to integrate these partitions into GML models. Specifically, we propose *cluster attention (CLATT)* — a graph-based attention mechanism in which nodes can attend to other nodes in the same cluster and thus exchange information with them. We view cluster attention as a middle ground between MPNNs in which only neighboring nodes can interact with each other and Graph Transformers in which all nodes can interact with each other. Cluster attention allows capturing longer-ranged dependencies than MPNNs while still providing strong graph-structure-based inductive biases which Graph Transformers lack.

We argue that the graph-structure-based inductive biases of cluster attention are complementary to those of MPNNs and thus propose augmenting MPNNs with cluster attention. Similarly, Graph Transformers can be augmented with CLATT to provide these global attention models with stronger graph-structure-based inductive biases. In experiments on a diverse set of 12 real-world graph datasets, we demonstrate that cluster attention can significantly improve the performance of both MPNNs and Graph Transformers.

The rest of this paper is structured as follows. Section 2 provides the necessary background on GML with neural networks and on graph node clustering. In Section 3, we formally define cluster attention and discuss its implementation details. In Section 4, we discuss our approach to selecting graph node clustering algorithms to use with cluster attention. Section 5 presents our experimental results. Section 6 provides concluding remarks.

2 Background and related work

2.1 Graph Neural Networks and Graph Transformers

In recent years, Graph Neural Networks (GNNs), in particular Message Passing Neural Networks (MPNNs) (Gilmer et al., 2017), have become the most dominant approach to graph machine learning tasks. In each message passing layer of an MPNN, each node aggregates representations from its neighbors in the graph (by receiving messages from them) and updates its own representation based on the aggregated information. Many GNN architectures falling under the MPNNs framework have been proposed (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018; Xu et al., 2019; Brody et al., 2022), mostly differing in their aggregation function. The receptive field of MPNNs, i.e., the maximum shortest path distance in the graph at which nodes can interact with each other, is equal to the number of message passing layers. This is often viewed as a limitation, since long-range dependencies can exist in some graph machine learning problems. While there have been some works attempting to increase the receptive field of MPNNs (Abu-El-Haija et al., 2019; Finder et al., 2025), a different and more radical approach has become more popular: replacing MPNNs with models with all-to-all Transformer-style attention (Vaswani et al., 2017) that has global receptive field. Such models became known as Graph Transformers (GTs) (Kreuzer et al., 2021; Ying et al., 2021; Wu et al., 2022; Rampášek et al., 2022). However, global attention has no information about graph topology and thus this approach foregoes graph-structure-based inductive biases of MPNNs and essentially treats the graph nodes as a set rather than a graph. To inject graph information into GTs, various positional encoding (PEs) are typically added to node features or attention weights. Such PEs are often based on Laplacian eigenvectors (Dwivedi et al., 2020), graph distances (Ying et al., 2021), or random walks (Dwivedi et al., 2022). However, despite the success of the Transformer architecture in other fields such as natural language processing, computer vision, and audio processing, and despite considerable research efforts to adapt Transformers to graph machine learning, recent benchmarking works show that GTs tend to not provide advantages over MPNNs (Tönshoff et al., 2023; Luo et al., 2024; 2025).

It is also important to note that there are actually two types of models commonly referred to as Graph Transformers. Besides Graph Transformers with all-to-all attention discussed above, another type consists of models that adopt the Transformer attention mechanism (and often other elements of the Transformer architecture) to GNNs but limit attention to each node's neighborhood, i.e., each node can only attend to its neighbors (Shi et al., 2021; Dwivedi & Bresson, 2021). Such models fit into the MPNNs framework and have been shown to be a very strong MPNN variant (Platonov et al., 2023b; Bazhenov et al., 2025). To distinguish between these two model classes, we will henceforth refer to models with all-to-all Transformer attention as Global Graph Transformers (GGTs) and to models with neighborhood Transformer attention as Local Graph Transformers (LGTs).

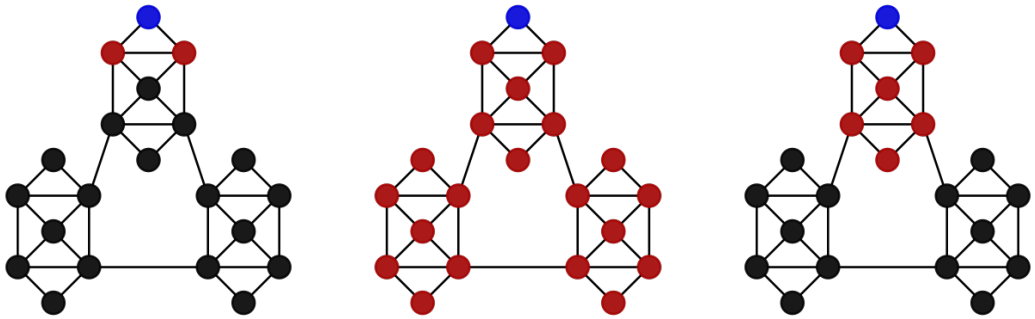
2.2 GRAPH CLUSTERING

Graph node clustering (henceforth graph clustering), also referred to as community detection, graph partitioning, and modular structure inference, is an important and long-studied problem in network science and graph machine learning with many applications in various fields. Many widely different approaches to graph clustering have been proposed, see Fortunato (2010); Fortunato & Hric (2016); Schaeffer (2007) for detailed overviews. Here, we briefly review some of the most well-known approaches. Many methods are based on intuitive heuristics such as iterative propagation of influence between nodes (Raghavan et al., 2007) or divisive clustering by removing "bridge-like" edges based on betweenness centrality (Girvan & Newman, 2002) or approximate minimum cuts (and variants such as ratio cuts and normalized cuts) (Fiedler, 1973; Pothen et al., 1990; Shi & Malik, 2000; Meila & Shi, 2000; Andersen et al., 2007; Riolo & Newman, 2014). Another approach that gained a lot of popularity is maximizing a certain measure (quality function) that shows how well-connected the nodes within a single cluster are compared to nodes from different clusters. The most well-known such measure is modularity (Newman & Girvan, 2004; Newman, 2006b), and many algorithms to maximize it have been proposed (Clauset et al., 2004; Duch & Arenas, 2005; Reichardt & Bornholdt, 2006; Guimera & Nunes Amaral, 2005; Newman, 2006a), the most popular one being the greedy Louvain algorithm (Blondel et al., 2008). However, modularity maximization for graph clustering has some limitations (Fortunato & Hric, 2016), including a problem known as the resolution limit (Fortunato & Barthelemy, 2007). An alternative to the modularity measure that does not suffer from the resolution limit is the Constant Potts Model (CPM) (Traag et al., 2011), which can also be maximized with the Louvain algorithm. An improvement to the Louvain algorithm that can be used to greedily maximize either modularity or CPM is the Leiden algorithm (Traag et al., 2019). Another popular approach to graph clustering is based on statistical inference: the graph is assumed to be generated from a certain random graph model family parametrized by community assignments of nodes, and the assignment with maximum likelihood is searched for. Methods following this approach differ in what model family they assume and how they maximize likelihood (Hastings, 2006; Newman & Leicht, 2007; Zanghi et al., 2008; Hofman & Wiggins, 2008; Copic et al., 2009; Karrer & Newman, 2011; Peixoto, 2014a;b; Zhang et al., 2015; Prokhorenkova & Tikhonov, 2019; Zhang & Peixoto, 2020). The approaches discussed above consider unattributed graphs, i.e., graphs without node features. More recently, GNN-based methods for clustering attributed graphs that consider both graph structure and node features have been developed (Tsitsulin et al., 2023; Liu et al., 2026).

3 CLUSTER ATTENTION

3.1 METHOD

The standard MPNN mechanism of sending messages along graph edges provides the model with strong graph-structure-based inductive biases, but prevents nodes further away in the graph than the number of message passing layers from exchanging information. We would like to design a mechanism that allows longer-range interactions while still being rooted in the graph structure and thus providing useful inductive biases to the model (in contrast to the all-to-all attention of GGTs). We propose using graph clustering to achieve this. Specifically, we apply graph clustering as a preprocessing step and then allow nodes within the same cluster to interact with each other via attention. Essentially, we use all-to-all attention on the level of individual clusters rather than on the full-graph level (as in GGTs). Such attention mechanism allows capturing longer-range dependencies than the classic message passing mechanism while still limiting which nodes can interact with each other based on the graph structure. We call this mechanism *Cluster Attention (CLATT)*. We propose integrating CLATT into MPNNs alongside their message passing, since CLATT and message passing provide different kinds of graph-structure-based inductive biases that can be mutually beneficial. Similarly, CLATT can be integrated into GGTs alongside their global attention to provide GGTs with useful inductive biases. This approach is similar to how global attention of GGTs can be combined with message passing in a hybrid model (Rampášek et al., 2022). In Figure 1, we provide a simple example of how the receptive fields of a single layer of an MPNN, a GGT, and CLATT differ. Note that stacking multiple layers with message passing and CLATT allows nodes from different clusters to also interact with each other (by using several steps of message passing and/or cluster attention) thus allowing the model to capture even longer-range dependencies.



(a) MPNN (b) GGT (c) CLATT

Figure 1: Receptive fields of a single layer of different GML models. The blue node can interact with the red nodes. In an MPNN, the receptive field is based on the graph structure, but only neighboring nodes can interact with each other. GGT allows all nodes to interact with each other, but completely disregards the graph structure. CLATT allows nodes within the same cluster to interact, which provides longer-range interactions than in an MPNN, but still relies on the graph structure to disallow most possible pairwise interactions.

Note that modern efficient graph clustering algorithms such as the Leiden algorithm (Traag et al., 2019) run orders of magnitude faster than it takes to train a GNN; thus, the graph clustering preprocessing step takes negligible time. However, a question remains: how to choose a graph clustering algorithm for CLATT from the many different approaches that have been proposed in the literature. We will discuss clustering algorithm selection in detail in Section 4, but for now let us note that it is not necessary to limit CLATT to *a single* graph clustering algorithm. We can select *multiple* graph clusterings obtained with different algorithms, apply CLATT to each one, and then concatenate (or combine in some other way, e.g., via summation) the resulting representations for each node. Different clustering algorithms rely on different assumptions and have different inductive biases, thus producing significantly different clusterings that capture different types of information about the graph structure, and using multiple such clusterings lets a model access all of these different types of information.

Now let us describe CLATT more formally. Let $G = (V, E)$ be a graph with the nodeset V and the edgeset E. Let $\mathcal{C}$ be an ordered set of clusterings that contains one or more clusterings that we want to use with CLATT. For each clustering $C \in \mathcal{C}$ and each node $i \in V$ we will denote by $C(i)$ the cluster (the set of nodes) to which the node i belongs in the clustering C. Let $\mathbf{x}_i \in \mathbb{R}^d$ be the representation of node $i \in V$ before the CLATT operation, i.e., the input to CLATT, and let $\mathbf{y}_i \in \mathbb{R}^{d|\mathcal{C}|}$ be the representation of node $i \in V$ after the CLATT operation, i.e., the output of CLATT. The complete CLATT output representation $\mathbf{y}_i \in \mathbb{R}^{d|\mathcal{C}|}$ for node i is the concatenation of CLATT output representations $\mathbf{y}_i^C \in \mathbb{R}^d$ for each clustering $C \in \mathcal{C}$:

$$\mathbf{y}_i = \text{concatenate}\left(\mathbf{y}_i^C \text{ for } C \text{ in } \mathcal{C}\right). \tag{1}$$

The CLATT output representation $\mathbf{y}_i^C$ for a single clustering $C \in \mathcal{C}$ for node i is computed via attention between nodes belonging to the cluster $C(i)$ in the following way (for ease of notation we assume a single attention head, but we use multihead attention in practice):

$$\mathbf{y}_i^C = \sum_{j \in C(i)} p_{ij}^C \mathbf{v}_j^C, \tag{2}$$

where $p_{ij}^C \in [0, 1]$ is the weight (probability) of attention from node i to node j obtained with softmax as

$$p_{ij}^C = \frac{\exp\left(\alpha_{ij}^C\right)}{\sum_{j \in C(i)} \exp\left(\alpha_{ij}^C\right)}, \tag{3}$$

α_{ij}^C is the pre-softmax logit of attention from node i to node j obtained as a scaled dot product between the corresponding query and key vectors:

$$\alpha_{ij}^C = \frac{\langle \mathbf{q}_i^C, \mathbf{k}_j^C \rangle}{\sqrt{d}}, \tag{4}$$

and $\mathbf{q}_i^C, \mathbf{k}_i^C, \mathbf{v}_i^C \in \mathbb{R}^d$ are the query, key, and value vectors respectively for node $i \in V$ obtained as learnable linear transformations of the node's representation:

$$\mathbf{q}_i^C = \mathbf{W}_q^C \mathbf{x}_i + \mathbf{b}_q^C, \qquad \mathbf{k}_i^C = \mathbf{W}_k^C \mathbf{x}_i + \mathbf{b}_k^C, \qquad \mathbf{v}_i^C = \mathbf{W}_v^C \mathbf{x}_i + \mathbf{b}_v^C, \tag{5}$$

where $\mathbf{W}_q^C, \mathbf{W}_k^C, \mathbf{W}_v^C \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_q^C, \mathbf{b}_k^C, \mathbf{b}_v^C \in \mathbb{R}^d$ are learnable parameters. The complete CLATT output representation $\mathbf{y}_i \in \mathbb{R}^{d|\bar{e}|}$ for node i is then concatenated with the message passing output (in a CLATT-augmented MPNN model) or the global attention output (in a CLATT-augmented GGT model) for node i (which is typically of dimension d) and passed through a learnable linear layer that reduces its dimension back to d.

3.2 Implementation details

CLATT can be naively implemented by adding edges connecting all pairs of nodes within the same cluster to the graph. However, this will likely create a very dense graph for which graph attention operation meant for sparser graphs will be inefficient. Instead, we transform the tensor of node representations of shape `num_nodes × hidden_dim` to a padded tensor of shape `num_clusters × max_cluster_size × hidden_dim` and apply standard dense attention along the `max_cluster_size` dimension with a mask that blocks attention to padding. Modern efficient attention implementations[1] combined with JIT-compilation[2] make this operation very efficient. Further, some modern frameworks provide so-called "ragged / jagged" tensors[3][4] which can be used to stack sequences of different length into a single tensor-like structure, which can allow avoiding padding to make CLATT implementation even more efficient.

4 Clustering algorithms selection

Considering the wide and diverse range of graph clustering algorithms available (see Section 2.2), it is important to choose appropriate clustering algorithms to use with CLATT. Different clustering algorithms rely on different principles and make different assumptions about the data, and thus can produce significantly different clusterings. Since real-world graphs come from many different applications and have very different structures, the best clustering algorithm for use with CLATT can also be different for different datasets and tasks. First, we note that we do not need to limit CLATT to using *a single* clustering: we can select *multiple* clustering algorithms and apply CLATT to clusterings obtained from each of them. However, we still need to choose a relatively small set of clustering algorithms to use from the wide range of available ones. In this selection, we use the following two criteria: first, we only consider efficient algorithms that can be run relatively fast on large-scale real-world graphs, making graph clustering a cheap preprocessing step compared to GNN training; second, we aim to select a set of clustering algorithms that produce significantly different results compared to each other, thus providing CLATT with different information about the graph structure when used together. Based on these criteria, we select 4 clustering algorithms. In practice, different clustering algorithms can have different usefulness for different datasets and tasks, and there is no need to always use all 4 of them. Thus, we treat the choice of clustering algorithms for each dataset as a hyperparameter. First, we train a model 4 times with each clustering individually; then, we take those of the considered clusterings that improved results on the validation set and use them together for the main training run. To speed up experiments, we only run this procedure with one of the considered models (specifically, LGT) and use the same selection of clusterings for all other models (we assume that the selection of clusterings will transfer well between different models, but it is possible that even better results can be achieved if this selection is run for each model individually).

Now, let us describe the 4 clustering algorithms that we use.

- First, we consider the widely used Leiden algorithm (Traag et al., 2019) — an improvement of the classic Louvain algorithm (Blondel et al., 2008). Leiden algorithm greedily optimizes a measure

[1] `https://docs.pytorch.org/docs/stable/generated/torch.nn.functional.scaled_dot_product_attention.html`

[2] `https://docs.pytorch.org/tutorials/intermediate/torch_compile_tutorial.html`

[3] `https://docs.pytorch.org/docs/stable/nested.html`

[4] `https://docs.pytorch.org/tutorials/unstable/nestedtensor.html`

of how well-connected nodes within a single cluster are compared to nodes from different clusters. As this measure, we use the Constant Potts Model (CPM) (Traag et al., 2011), which does not suffer from the resolution limit problem (Fortunato & Barthelemy, 2007), in contrast to the more widely used modularity measure (Newman & Girvan, 2004; Newman, 2006b). We use the official algorithm implementation.[5]

- Next, we consider a statistical clustering algorithm. Specifically, we use an algorithm that fits a Bayesian planted partition model by Zhang & Peixoto (2020). We use the official algorithm implementation from the `graph-tool` library (Peixoto, 2017).
- Note that the two algorithms discussed above assume *assortative* community structure, i.e., nodes within the same cluster are more likely to be connected than nodes from different clusters. While such community structure is natural, sometimes it can also be useful to find *disassortative* clusters, i.e., clusters of nodes that are not necessarily connected with each other but share the same structural role in the graph. For example, when a lot of leaf nodes are connected to one node in the graph, they can be expected to share some properties, and it can be useful to put them in a cluster of their own even though they are not connected with each other. Thus, we also consider a statistical clustering algorithm that does not make the assumption of cluster assortativity and can detect disassortative clusters. Specifically, we use an algorithm by Peixoto (2014b). This algorithm produces a hierarchical clustering; however, we only consider the level of the hierarchy with the smallest nontrivial number of clusters as we find that other levels typically find clusters that are too small. We use the official algorithm implementation from the `graph-tool` library (Peixoto, 2017).
- Above, we have described 3 very different graph clustering algorithms. However, it can sometimes be useful to also consider the *feature similarity* of nodes, i.e., ignore the graph structure and apply a classic clustering algorithm to node features. This approach will allow the model to exchange information between nodes that have similar features no matter how far they are in the graph, which can be useful for some applications. For this purpose, we use the classic k-means algorithm (Forgy, 1965; Lloyd, 1982). We use the implementation from the `scikit-learn` library (Pedregosa et al., 2011). However, since the k-means algorithm relies on euclidean distances between feature vectors, it is not particularly suitable for datasets with heterogeneous features of different scales and distributions, which are common in practical GML applications (Bazhenov et al., 2025). Thus, instead of directly clustering node features, we train an auxiliary ResMLP model (a simple graph-agnostic model that is very fast to train) on the dataset, obtain hidden representations of graph nodes from this model, and cluster these representations. We believe it is interesting to compare this feature-based clustering to graph-structure-based clusterings. In our experiments, we find that this feature-based clustering rarely helps (see Appendix B), which confirms our intuition that graph-structure-based inductive biases are more useful for GML.

The 4 considered algorithms are very efficient and also typically produce significantly different clusterings, thus satisfying our desiderata. To verify that these algorithms produce significantly different results, we can compare the clusterings obtained from these algorithms using some *clustering similarity measure*. While many such measures have been proposed in the literature (Gösgens et al., 2021), we use the measure known as *Correlation Coefficient (CC)* since it has been shown to satisfy many desirable properties (Gösgens et al., 2021). We show the similarity matrices of the clusterings obtained by the 4 considered algorithms on 4 datasets used in our experiments (see Appendix C for dataset descriptions) in Figure 2. Indeed, the similarity values of clusterings are mostly relatively low, with the graph-agnostic k-means clustering typically being particularly dissimilar to the other (graph-based) clusterings, as expected.

5 Experiments

5.1 Datasets

In our experiments, we aim to show that CLATT can improve results on diverse datasets from different domains. In particular, we aim to go beyond the classic citation network datasets, extensive reliance on which has been criticized recently (Bechler-Speicher et al., 2025; Bazhenov et al., 2025; Platonov et al., 2026), and use graph datasets from more realistic and practically relevant applications. Thus, we

[5] `leidenalg.readthedocs.io`

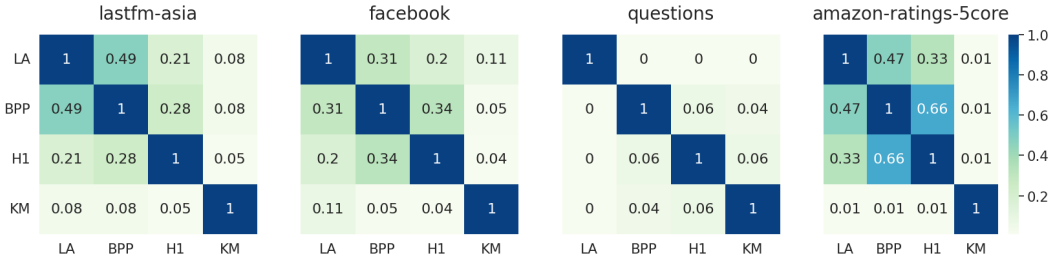


Figure 2: Similarities of clusterings produced by different node clustering algorithms on 4 different datasets: `lastfm-asia`, `facebook`, `questions`, `amazon-ratings-5core`. The Correlation Coefficient similarity measure is reported. LA is Leiden algorithm, BPP is Bayesian planted partition model, H1 is hierarchical statistical clustering (with the hierarchy level with the smallest non-trivial number of clusters being used), KM is k-means clustering of ResMLP node representations.

Table 1: Characteristics of the considered graph datasets.

	node regression			node classification								
	city-roads-M	city-roads-L	avazu-ctr	tolokers-2	city-reviews	hm-categories	pokec-regions	questions	lastfm-asia	facebook	amzn-r-5core	amzn-r-full
# nodes	57.1K	142.3K	76.3K	11.8K	148.8K	46.5K	1.6M	48.9K	7.6K	22.5K	24.5K	334.9K
# edges	107.1K	231.6K	11.0M	519.0K	1.2M	10.7M	22.3M	153.5K	27.8K	170.8K	93.1K	925.8K
avg degree	3.75	3.26	288.04	88.28	15.66	460.92	27.32	6.28	7.29	15.20	7.60	5.53
median degree	4	3	71	30	4	45	13	1	4	7	5	4
avg distance	126.75	194.05	3.55	2.79	4.91	2.45	4.68	4.29	5.23	4.97	16.24	11.95
diameter	383	553	14	11	19	13	14	16	15	15	46	47
global clustering	0.00	0.00	0.24	0.23	0.26	0.27	0.05	0.02	0.18	0.23	0.32	0.21
avg local clustering	0.00	0.00	0.85	0.53	0.41	0.70	0.11	0.03	0.22	0.36	0.58	0.40
degree assortativity	0.70	0.74	−0.30	−0.08	0.01	−0.35	0.00	−0.15	0.02	0.08	−0.09	−0.06
# classes	N/A	N/A	N/A	2	2	21	183	2	18	4	5	5
unbiased homophily	N/A	N/A	N/A	0.10	0.69	0.38	0.98	0.06	0.97	0.90	0.28	0.22
target assortativity	0.74	0.72	0.18	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

select 12 datasets from different fields and with different tasks, showcasing a wide range of graph sizes and structural characteristics. First, we use 7 datasets from the recently introduced GraphLand benchmark (Bazhenov et al., 2025) that focuses on graph datasets from realistic industrial applications and with rich node features. Specifically, we use road networks `city-roads-M` and `city-roads-L`, a network of internet-connected devices `avazu-ctr`, a network of crowdsourcing platform workers `tolokers-2`, a network of review service users `city-reviews`, a co-purchasing network `hm-categories`, and a social network `pokec-regions`. For datasets from GraphLand, we use the official `RL` (random low) 10%/10%/80% train/val/test data splits. Further, we use two classic social network datasets `lastfm-asia` (Rozemberczki & Sarkar, 2020) and `facebook` (Rozemberczki et al., 2019). For these two datasets, we use random 10%/10%/80% train/val/test data splits. Then, we use two datasets from Platonov et al. (2023b): co-purchasing network `amazon-ratings` and question-answering network `questions`. For these two datasets, we use random 50%/25%/25% train/val/test data splits. Finally, we notice that the `amazon-ratings` dataset has a peculiar graph structure due to being downsampled by taking the 5-core (Malliaros et al., 2020) of the full graph. For this reason, we further refer to this dataset as `amazon-ratings-5core`, and we also prepare a new dataset which is the full version of the same co-purchasing network, which we call `amazon-ratings-full`. Besides being more than an order of magnitude larger, this dataset exhibits very different structural characteristics compared to `amazon-ratings-5core`. For this dataset, we use a random 10%/10%/80% train/val/test data split. A detailed description of this new dataset and how it differs from `amazon-ratings-5core` is provided in Appendix D.

To showcase the diversity of graph structures in the considered datasets, we provide some of their structural characteristics in Table 1. More details on these datasets and their characteristics are provided in Appendix C. For example, note that we use a range of both homophilous datasets (`city-roads-M`, `city-roads-L`, `city-reviews`, `pokec-regions`, `lastfm-asia`,

facebook) and non-homophilous datasets (avazu-ctr, tolokers-2, hm-categories, questions, amazon-ratings-5core, amazon-ratings-full) to demonstrate that CLATT works well for both of these types of datasets.

The largest of the considered datasets — pokec-regions — has more than 1.6 million nodes. GGTs are not efficient enough to handle datasets of such size in reasonable time (a single training run of GGT on this dataset takes approximately 65 hours using an NVIDIA Tesla A100 80GB GPU), while our LGT-CLATT model (which is a CLATT-augmented version of the most computationally expensive of the considered MPNNs) can be trained on it in approximately half an hour (using the same GPU), which demonstrates the efficiency of our CLATT implementation.

5.2 Models

As discussed above, we propose integrating CLATT into MPNNs to combine the benefits of local message passing and more long-range cluster attention. We combine CLATT with three different MPNN models and show that CLATT improves results for all of them. Specifically, we use two classic GNNs — GCN (Kipf & Welling, 2017) and GraphSAGE (Hamilton et al., 2017) — and also Local Graph Transformer (LGT), which has been shown to often provide stronger results in recent benchmarking works (Platonov et al., 2023b; Bazhenov et al., 2025). For all models, we use the modifications from Platonov et al. (2023b) which augment models with skip-connections (He et al., 2016), layer normalization (Ba et al., 2016), and MLP blocks between message passing layers. These modifications have been shown to significantly improve the results of MPNNs (Luo et al., 2024; 2025).

Further, we also combine CLATT with Global Graph Transformers (GGTs) and show that it improves their performance as well. We experiment with two versions of GGT that differ by the positional encodings used: we use either classic DeepWalk node embeddings (Perozzi et al., 2014) or Laplacian eigenvectors (Dwivedi et al., 2020; Belkin & Niyogi, 2001). We denote these models GGT-DW and GGT-Lap, respectively. Note that while many alternative node embedding methods have been proposed since DeepWalk, it has been shown that in practice DeepWalk still provides some of the best results (Gurukar et al., 2022).

For all models, we perform extensive hyperparameter search using the search spaces from Bazhenov et al. (2025). More details of our experimental setting are provided in Appendix E.

5.3 Experimental results

We report the results of our experiments in Table 2. For each of the considered models, we compare the base version of the model with its version augmented with CLATT (which is denoted by -CLATT suffix). We can see that CLATT improves the performance of the base models in all cases with the only exception of the GraphSAGE model on the amazon-ratings-5core dataset, and these improvements are often quite substantial. For example, on the pokec-regions dataset, CLATT improves the performance of GCN and GraphSAGE by more than 13 and 10 percentage points, respectively. The improvements from CLATT are not limited to MPNNs: for GGTs, CLATT always improves performance by providing these models with graph-structure-based inductive biases. For example, on the hm-categories dataset, CLATT improves the performance of GGT-DW and GGT-Lap by more than 13 and 15 percentage points, respectively.

We note that CLATT brings improvements on datasets with very different graph structural characteristics. As discussed above, the considered datasets cover both homophilous and non-homophilous graphs. Further, CLATT improves model performance on both very dense graphs (avazu-ctr, hm-categories) and very sparse graphs (city-roads-M, city-roads-L, questions, amazon-ratings-full). Note that city-roads-M and city-roads-L datasets, representing transportation networks, do not exhibit particularly well-defined clusters (as can be expected given their zero clustering coefficients and very large average distances), in contrast to, for example, social networks, yet CLATT brings strong benefits on them as well. More details on the diversity of graph structural characteristics covered by the considered datasets are provided in Appendix C.

In Appendix A, we provide an analysis of attention patterns learned by local, global, and cluster attention. It demonstrates that, on the one hand, cluster attention often works at longer ranges than the receptive field of local attention models permits, which allows it to capture useful long-range

Table 2: Experimental results. R^2 is reported for regression datasets, Average Precision is reported for binary classification datasets, Accuracy is reported for multiclass classification datasets. TL means time limit exceeded (24 hours for a single model run). For each pair consisting of a base model and its version augmented with CLATT, we highlight the best of the two results with bold and additionally underline it if the difference between the two results is statistically significant.

(a) Experimental results for GraphLand datasets under the `RL` data split.

	regression			bin. class.		mult. class.	
	city-roads-M	city-roads-L	avazu-ctr	tolokers-2	city-reviews	hm-categories	pokec-regions
GCN	59.05 ± 0.16	53.26 ± 0.14	30.47 ± 0.27	51.48 ± 0.81	77.15 ± 0.28	61.88 ± 0.23	34.99 ± 0.17
GCN-CLATT	$\mathbf{59.78 \pm 0.28}$	$\mathbf{55.15 \pm 0.20}$	$\mathbf{30.74 \pm 0.26}$	$\mathbf{54.08 \pm 0.61}$	$\mathbf{77.79 \pm 0.18}$	$\mathbf{64.96 \pm 0.24}$	$\mathbf{48.88 \pm 0.44}$
GraphSAGE	57.51 ± 0.53	52.43 ± 0.25	31.84 ± 0.24	53.87 ± 0.78	77.82 ± 0.13	56.72 ± 0.31	37.80 ± 0.35
GraphSAGE-CLATT	$\mathbf{60.67 \pm 0.18}$	$\mathbf{55.59 \pm 0.24}$	$\mathbf{32.44 \pm 0.22}$	$\mathbf{54.76 \pm 0.63}$	$\mathbf{77.91 \pm 0.12}$	$\mathbf{61.90 \pm 0.25}$	$\mathbf{48.75 \pm 0.91}$
LGT	58.05 ± 0.58	53.38 ± 0.12	30.87 ± 0.47	55.70 ± 0.28	76.97 ± 0.21	69.25 ± 0.25	46.41 ± 0.20
LGT-CLATT	$\mathbf{60.26 \pm 0.28}$	$\mathbf{55.75 \pm 0.34}$	$\mathbf{31.48 \pm 0.45}$	$\mathbf{56.95 \pm 0.28}$	$\mathbf{77.40 \pm 0.16}$	$\mathbf{70.22 \pm 0.29}$	$\mathbf{49.34 \pm 0.29}$
GGT-DW	53.14 ± 0.50	48.24 ± 0.82	27.78 ± 0.62	56.27 ± 0.34	74.70 ± 0.19	41.72 ± 0.41	TL
GGT-DW-CLATT	$\mathbf{54.58 \pm 0.44}$	$\mathbf{48.72 \pm 0.41}$	$\mathbf{28.71 \pm 0.41}$	$\mathbf{57.93 \pm 0.45}$	$\mathbf{76.25 \pm 0.20}$	$\mathbf{55.44 \pm 0.34}$	TL
GGT-Lap	54.32 ± 0.64	46.48 ± 0.99	24.27 ± 1.16	49.46 ± 1.05	71.40 ± 0.36	37.79 ± 0.34	TL
GGT-Lap-CLATT	$\mathbf{57.53 \pm 0.18}$	$\mathbf{49.86 \pm 0.34}$	$\mathbf{25.98 \pm 0.33}$	$\mathbf{53.77 \pm 0.82}$	$\mathbf{76.62 \pm 0.16}$	$\mathbf{52.80 \pm 0.40}$	TL

(b) Experimental results for other datasets.

	bin. class.	mult. class.			
	questions	lastfm-asia	facebook	amazon-ratings-5core	amazon-ratings-full
GCN	18.60 ± 0.59	80.62 ± 0.36	91.00 ± 0.14	49.65 ± 0.50	39.28 ± 0.13
GCN-CLATT	$\mathbf{20.48 \pm 0.83}$	$\mathbf{84.01 \pm 0.32}$	$\mathbf{91.71 \pm 0.21}$	$\mathbf{52.84 \pm 0.53}$	$\mathbf{39.76 \pm 0.13}$
GraphSAGE	20.41 ± 0.47	80.90 ± 0.46	91.22 ± 0.21	$\mathbf{54.94 \pm 0.38}$	38.73 ± 0.12
GraphSAGE-CLATT	$\mathbf{20.89 \pm 0.30}$	$\mathbf{84.54 \pm 0.19}$	$\mathbf{92.09 \pm 0.24}$	54.70 ± 0.43	$\mathbf{39.21 \pm 0.11}$
LGT	17.86 ± 0.52	83.97 ± 0.27	92.17 ± 0.20	51.26 ± 0.38	39.03 ± 0.08
LGT-CLATT	$\mathbf{21.28 \pm 0.68}$	$\mathbf{84.55 \pm 0.21}$	$\mathbf{92.71 \pm 0.13}$	$\mathbf{53.70 \pm 0.49}$	$\mathbf{39.34 \pm 0.17}$
GGT-DW	16.26 ± 0.72	78.69 ± 0.50	86.84 ± 0.27	46.22 ± 0.79	TL
GGT-DW-CLATT	$\mathbf{19.69 \pm 1.09}$	$\mathbf{83.31 \pm 0.42}$	$\mathbf{91.31 \pm 0.27}$	$\mathbf{48.42 \pm 0.37}$	TL
GGT-Lap	18.38 ± 0.71	63.50 ± 1.16	73.73 ± 0.73	45.97 ± 0.71	TL
GGT-Lap-CLATT	$\mathbf{21.20 \pm 0.51}$	$\mathbf{82.43 \pm 0.37}$	$\mathbf{89.90 \pm 0.22}$	$\mathbf{52.03 \pm 0.50}$	TL

dependencies not available to local attention, but, on the other hand, cluster attention typically works at shorter ranges than global attention, which struggles to leverage useful information encoded in the graph topology due to its lack of appropriate graph-structure-based inductive biases.

Specific clustering algorithms selected during hyperparameter search for each of the considered datasets are provided in Appendix B. We can see that using CLATT with more than one clustering algorithm simultaneously is beneficial for 50% of the datasets. The Leiden algorithm is the most frequently selected among the considered algorithms (it turns out to be helpful for 75% of the datasets), but the two considered statistical clustering algorithms are also regularly selected. In contrast, the k-means algorithm applied to ResMLP node representations turns out to be helpful for only one of the considered datasets (`questions`). This shows that using graph-structure-based clusterings with CLATT is much more beneficial than using feature-similarity-based clusterings, which agrees with our intuition that CLATT is most useful for providing additional graph-structure-based inductive biases to GML models.

6 Conclusion

In this work, we introduce a new technique for graph machine learning — cluster attention (CLATT), which can be viewed as a middle ground between local message passing and global attention that incorporates strong graph-structure-based inductive biases while allowing for long-range interactions. We show that CLATT significantly improves the results of both MPNNs and GGTs on a wide range of diverse real-world graph datasets.

AI USE STATEMENT

We did not use generative AI tools in this work. In particular, we did not use generative AI tools for any of the tasks with required disclosure: generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, interpreting results.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International conference on machine learning*, pp. 21–29. PMLR, 2019.

Reid Andersen, Fan Chung, and Kevin Lang. Using pagerank to locally partition a graph. *Internet Mathematics*, 4(1):35–64, 2007.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, et al. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 929–947, 2024.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Gleb Bazhenov, Oleg Platonov, and Liudmila Prokhorenkova. Graphland: Evaluating graph machine learning models on diverse industrial data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Maya Bechler-Speicher, Ben Finkelshtein, Fabrizio Frasca, Luis Müller, Jan Tönshoff, Antoine Siraudin, Viktor Zaverkin, Michael M Bronstein, Mathias Niepert, Bryan Perozzi, et al. Position: Graph learning will lose relevance due to poor benchmarks. In *International conference on machine learning*. Pmlr, 2025.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14, 2001.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008 (10):P10008, 2008.

Stefano Boccaletti, Ginestra Bianconi, Regino Criado, Charo I Del Genio, Jesús Gómez-Gardenes, Miguel Romance, Irene Sendina-Nadal, Zhen Wang, and Massimiliano Zanin. The structure and dynamics of multilayer networks. *Physics reports*, 544(1):1–122, 2014.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 70(6):066111, 2004.

Jernej Copic, Matthew O Jackson, and Alan Kirman. Identifying community structures from network data via maximum likelihood methods. *The BE Journal of Theoretical Economics*, 9(1), 2009.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359, 2022.

Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 72(2):027104, 2005.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *AAAI 2021 Workshop on Deep Learning on Graphs: Methods and Applications (DLG-AAAI 2021)*, 2021.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. *International Conference on Learning Representations (ICLR)*, 2022.

VP Dwivedi, CK Joshi, T Laurent, Y Bengio, and X Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2): 298–305, 1973.

Shahaf E Finder, Ron Shapira Weber, Moshe Eliasof, Oren Freifeld, and Eran Treister. Improving the effective receptive field of message-passing neural networks. In *International conference on machine learning*. PMLR, 2025.

Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics*, 21:768–769, 1965.

Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the national academy of sciences*, 104(1):36–41, 2007.

Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics reports*, 659:1–44, 2016.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. Pmlr, 2017.

Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

Martijn M Gösgens, Alexey Tikhonov, and Liudmila Prokhorenkova. Systematic analysis of cluster similarity indices: How to validate validation measures. In *International Conference on Machine Learning*, pp. 3799–3808. PMLR, 2021.

Roger Guimera and Luís A Nunes Amaral. Functional cartography of complex metabolic networks. *nature*, 433(7028):895–900, 2005.

Saket Gurukar, Priyesh Vijayan, Balaraman Ravindran, Aakash Srinivasan, Goonmeet Bajaj, Chen Cai, Moniba Keymanesh, Saravana Kumar, Pranav Maneriker, Anasua Mitra, et al. Benchmarking and analyzing unsupervised network representation learning and the illusion of progress. *Transactions on Machine Learning Research*, 2022.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

Matthew B Hastings. Community detection as an inference problem. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 74(3):035102, 2006.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jake M Hofman and Chris H Wiggins. Bayesian approach to network modularity. *Physical review letters*, 100(25):258701, 2008.

Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 83(1):016107, 2011.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 2017.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5–es, 2007.

Yunhui Liu, Pengyu Qiu, Yu Xing, Yongchao Liu, Peng Du, Chuntao Hong, Jiajun Zheng, Tao Zheng, and Tieke He. Bridging academia and industry: A comprehensive benchmark for attributed graph clustering. *arXiv preprint arXiv:2602.08519*, 2026.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic gnns are strong baselines: Reassessing gnns for node classification. *Advances in Neural Information Processing Systems*, 37:97650–97669, 2024.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Can classic gnns be strong baselines for graph-level tasks? simple architectures meet excellence. In *International conference on machine learning*. PMLR, 2025.

Fragkiskos D Malliaros, Christos Giatsidis, Apostolos N Papadopoulos, and Michalis Vazirgiannis. The core decomposition of networks: Theory, algorithms and applications. *The VLDB Journal*, 29 (1):61–92, 2020.

Marina Meila and Jianbo Shi. Learning segmentation by random walks. *Advances in neural information processing systems*, 13, 2000.

Mikhail Mironov and Liudmila Prokhorenkova. Revisiting Graph Homophily Measures. *Learning on Graphs Conference*, 2024.

Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 74(3):036104, 2006a.

Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006b.

Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

Mark EJ Newman and Elizabeth A Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences*, 104(23):9564–9569, 2007.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Tiago P Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 89(1):012804, 2014a.

Tiago P Peixoto. Hierarchical block structures and high-resolution model selection in large networks. *Physical Review X*, 4(1):011047, 2014b.

Tiago P Peixoto. The graph-tool python library. 2017.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing Graph Datasets for Node Classification: Homophily-Heterophily Dichotomy and Beyond. *Advances in Neural Information Processing Systems*, 36:523–548, 2023a.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? *International Conference on Learning Representations (ICLR)*, 2023b.

Oleg Platonov, Gleb Bazhenov, Dmitry Eremeev, and Liudmila Prokhorenkova. A fair evaluation of graph foundation models for node property prediction. In *The Workshop on Graph Foundation Models at ICML*, 2026.

Alex Pothen, Horst D Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM journal on matrix analysis and applications*, 11(3):430–452, 1990.

Liudmila Prokhorenkova and Alexey Tikhonov. Community detection through likelihood optimization: in search of a sound model. In *The World Wide Web Conference*, pp. 1498–1508, 2019.

Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 76(3):036106, 2007.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 74(1):016110, 2006.

Maria A Riolo and MEJ Newman. First-principles multiway spectral partitioning of graphs. *Journal of Complex Networks*, 2(2):121–140, 2014.

Benedek Rozemberczki and Rik Sarkar. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, pp. 1325–1334. ACM, 2020.

Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019*, pp. 65–72. ACM, 2019.

Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021.

Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. *Learning on Graphs Conference*, 2023.

Vincent A Traag, Paul Van Dooren, and Yurii Nesterov. Narrow scope for resolution-limit-free community detection. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 84(1): 016114, 2011.

Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.

Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *Journal of Machine Learning Research*, 24(127):1–21, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315*, 2019.

Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in neural information processing systems*, 35:27387–27401, 2022.

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

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

Hugo Zanghi, Christophe Ambroise, and Vincent Miele. Fast online graph clustering via erdős–rényi mixture. *Pattern recognition*, 41(12):3592–3599, 2008.

Lizhi Zhang and Tiago P Peixoto. Statistical inference of assortative community structures. *Physical Review Research*, 2(4):043271, 2020.

Xiao Zhang, Travis Martin, and Mark EJ Newman. Identification of core-periphery structure in networks. *Physical Review E*, 91(3):032803, 2015.

A ATTENTION DISTANCES ANALYSIS

In this section, we investigate how far nodes typically look with different types of attention. Specifically, we analyze the average distances between each node and the nodes it attends to weighted with attention probabilities for the three considered graph attention mechanisms: local (1-hop neighborhood), global (full-graph), and cluster attention. We consider three base models — LGT (local attention), GGT-DW (global attention), GGT-Lap (global attention), and their cluster attention augmented variants — LGT-CLATT, GGT-DW-CLATT, GGT-Lap-CLATT, and analyze their attention patterns on two datasets — `lastfm-asia` and `tolokers-2`. We limit ourselves to these two datasets because they are the smallest among the ones considered in our work, and our analysis requires the full attention matrices for global attention, which require memory quadratic in the number of nodes (in our main experiments, we use efficient FlashAttention-based (Dao et al., 2022) attention algorithms which do not materialize the full attention matrices and thus allow us to scale even to very large graphs; however, for the analysis in this section, we need the full attention matrices).

For each dataset, each model, each attention type in the model (local, global, cluster attention), and each attention head, we compute the average distance to the nodes to which each node attends weighted by the corresponding attention probabilities, thus obtaining nk distances per dataset, model, and attention type, where n is the number of nodes in the graph and k is the number of attention heads corresponding to this attention type in all layers of the model combined (for cluster attention on the `lastfm-asia` dataset, there are actually $3nk$ distances since 3 different clusterings are used). In Figure 3, we provide histograms of distributions of these average attention distances, and in Table 3, we report the 0.05, 0.25, 0.50, 0.75, 0.95 quantiles of these distributions. Note that for local attention, the average attention distance is upper bounded by 1, since nodes can only attend to themselves (at a distance of 0) and to their direct neighbors in the graph (at a distance of 1).

First, let us note that the attention distance distributions for each of the 3 attention types are significantly different between the `lastfm-asia` and `tolokers-2` datasets. For global and cluster attention, this can be explained by the very different graph structure of these two datasets: in `lastfm-asia`, the unweighted average pairwise distance between nodes is a lot larger than in `tolokers-2`, and thus attention also typically works at greater distances in `lastfm-asia` than in `tolokers-2`. However, we note that local attention also has different distance distributions for these two datasets: in `lastfm-asia` nodes attend to themselves (rather than their neighbors) in local attention significantly more than in `tolokers-2`.

Now, let us focus on comparing cluster attention to other types of attention. We can see that in cluster attention, the vast majority of nodes attend on average at significantly larger distances than in local attention (in which the maximum distance is limited to 1), and often even at larger distances than the receptive field of a multilayer local-attention MPNN allows (which for this experiment is a distance of 3 since 3-layer MPNNs are used). Specifically, in `lastfm-asia`, more than 25% of nodes attend in cluster attention at distances larger than 3.5 on average, while in `tolokers-2`, more than 5% of nodes attend in cluster attention at distances larger than 3.1 on average, which are distances unattainable for local attention even with 3 layers. However, in cluster attention nodes also on average attend at significantly smaller distances than in global attention. For example, in `lastfm-asia`, the 0.75 quantile of cluster attention distance distributions is smaller than even the 0.05 quantile of global attention distance distributions for all model variants, which means that at least 75% of nodes in cluster attention attend on average at a distance smaller than that 95% of nodes in global attention attend on average at. The situation is similar for `tolokers-2`, although a bit less drastic, since distances are generally smaller in this dataset. Overall, it can be seen that global attention struggles to focus on nearby nodes in the graph, with the vast majority of nodes in global attention attending on average at a distance of more than 4 in `lastfm-asia` and more than 2 in `tolokers-2`, while cluster attention gives much larger focus to nearby nodes. This shows that, despite the use of positional encodings, global attention struggles to leverage useful information about node relations encoded in the graph structure and does not focus enough on graph neighbors, which explains why integrating CLATT into GGTs improves their performance significantly.

Our results from Section 5 show that adding cluster attention consistently improves the performance of both local-attention and global-attention models, and our observations from this section provide an explanation for why this happens. It can be useful for a model to pass information beyond the receptive field of local-attention models (and other MPNNs), which can be achieved with cluster

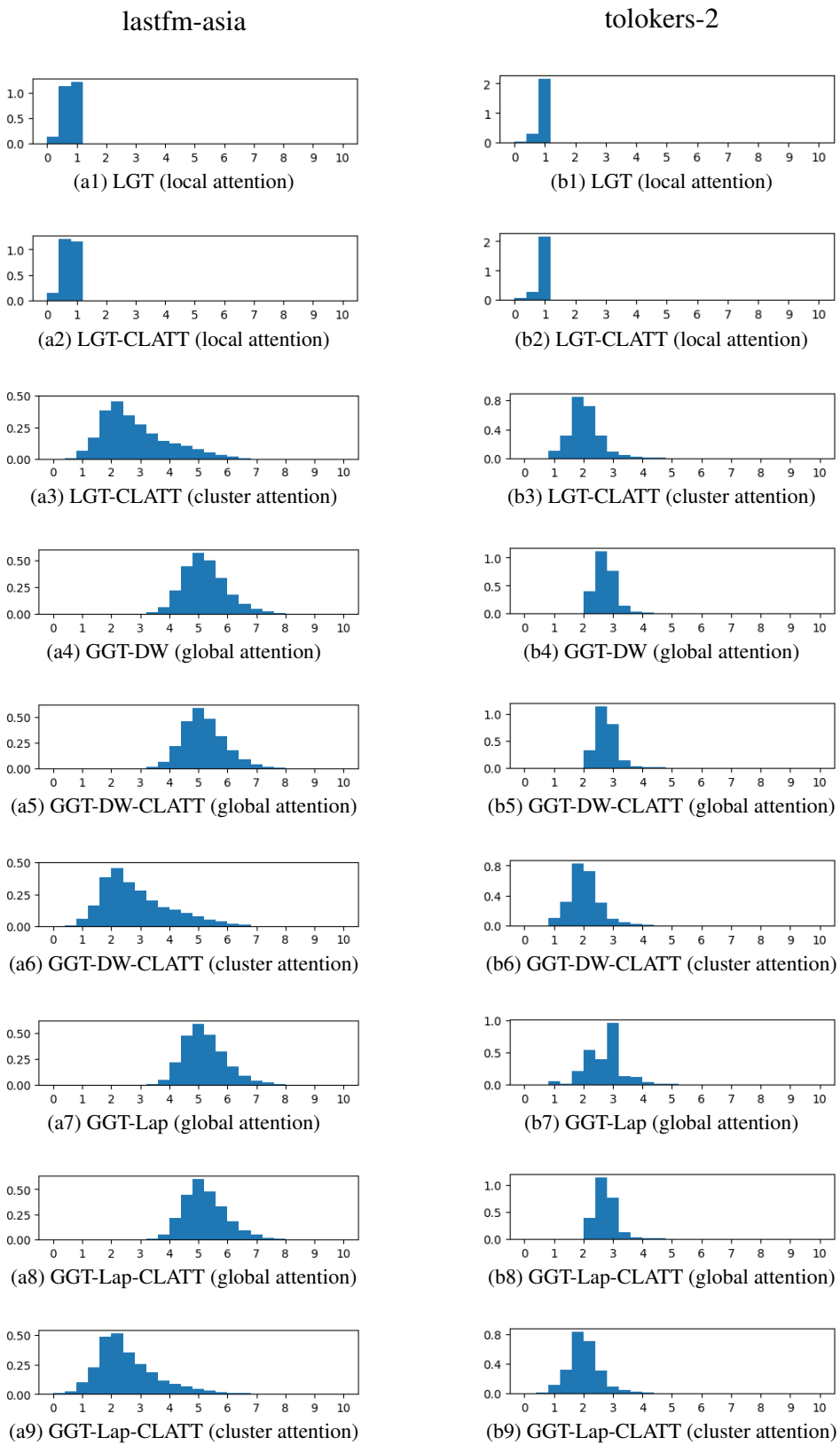


Figure 3: Distributions of average attention distances for different types of attention. Note that the unweighted average pairwise distance is 5.23 for `lastfm-asia` and 2.79 for `tolokers-2`.

Table 3: Quantiles of distributions of average attention distances for different types of attention. Note that the unweighted average pairwise distance is 5.23 for `lastfm-asia` and 2.79 for `tolokers-2`.

	lastfm-asia					tolokers-2				
quantile →	0.05	0.25	0.50	0.75	0.95	0.05	0.25	0.50	0.75	0.95
LGT (local attention)	0.39	0.62	0.79	0.90	0.97	0.59	0.91	0.97	0.99	1.00
LGT-CLATT (local attention)	0.38	0.60	0.78	0.89	0.96	0.56	0.91	0.98	0.99	1.00
LGT-CLATT (cluster attention)	1.35	1.99	2.56	3.50	5.23	1.23	1.73	1.99	2.32	3.14
GGT-DW (global attention)	4.13	4.71	5.16	5.67	6.61	2.23	2.49	2.71	2.93	3.35
GGT-DW-CLATT (global attention)	4.10	4.69	5.13	5.65	6.60	2.26	2.52	2.73	2.94	3.36
GGT-DW-CLATT (cluster attention)	1.39	2.01	2.60	3.55	5.25	1.23	1.73	2.00	2.33	3.16
GGT-Lap (global attention)	4.18	4.71	5.13	5.65	6.59	1.96	2.15	2.82	3.00	3.90
GGT-Lap-CLATT (global attention)	4.16	4.72	5.14	5.67	6.62	2.24	2.50	2.71	2.93	3.34
GGT-Lap-CLATT (cluster attention)	1.19	1.86	2.32	3.07	4.70	1.20	1.72	1.99	2.32	3.14

Table 4: Clustering algorithms selected for each of the considered datasets. LA is Leiden algorithm, BPP is Bayesian planted partition model, H1 is hierarchical statistical clustering (with the hierarchy level with the smallest non-trivial number of clusters being used), KM is k-means clustering of ResMLP node representations.

clustering algorithm →	LA	BPP	H1	KM
city-roads-M	✓	✓		
city-roads-L	✓	✓		
avazu-ctr	✓			
tolokers-2		✓		
city-reviews			✓	
hm-categories	✓			
pokec-regions			✓	
questions	✓			✓
lastfm-asia	✓	✓	✓	
facebook	✓		✓	
amazon-ratings-5core	✓	✓		
amazon-ratings-full	✓			

attention. While global-attention models also provide an opportunity for using long-range interactions, due to their lack of graph-structure-based inductive biases, they struggle to focus on nearby nodes and often use attention to attend further in the graph than is actually needed, which is why they also benefit from cluster attention which has strong graph-structure-based inductive biases and allows these models to more effectively focus on nodes close in the graph.

B CLUSTERING ALGORITHMS SELECTED

As described in Section 4, for each dataset, during hyperparameter search we select a subset of the 4 considered clustering algorithms based on the validation set performance. Table 4 specifies which clustering algorithms were selected for each of the considered datasets. All the considered algorithms except for k-means use the graph structure to cluster graph nodes, while k-means is applied to node representations obtained from a graph-agnostic ResMLP model and thus uses a variant of feature similarity (i.e., similarity of features transformed by a ResMLP). It can be seen that for 6 datasets (i.e., 50% of the considered datasets), only one clustering algorithm is selected, for another 5 datasets, two clustering algorithms are selected, and for 1 dataset (`lastfm-asia`), three clustering algorithms are selected. The Leiden algorithm is the most frequently selected clustering algorithm: it is selected for 9 datasets, i.e., 75% of the considered datasets. The two considered statistical clustering algorithms are also regularly selected: the Bayesian planted partition model is selected for 5 datasets, while the hierarchical clustering algorithm that does not assume assortative clustering structure is selected for 4 datasets. In contrast, the k-means algorithm is the least frequently selected algorithm — it is only selected once (for the `questions` dataset). This shows that algorithms that use the graph structure

to cluster graph nodes are typically more useful for CLATT than algorithms that use node feature similarity, which agrees with our intuition that CLATT is most beneficial in graph machine learning when used to provide additional graph-structure-based inductive biases.

C DATASETS

C.1 DATA SPLITS

In our experiments, we aim to use a range of datasets from different real-world applications of graph machine learning. Brief descriptions of these datasets are provided in Section 5.1. For datasets from the GraphLand benchmark (Bazhenov et al., 2025) (`city-roads-M`, `city-roads-L`, `avazu-ctr`, `tolokers-2`, `city-reviews`, `hm-categories`, `pokec-regions`) we use the official `RL` (random low) data splits, which are random stratified 10%/10%/80% train/val/test data splits. Similarly, for the `lastfm-asia` (Rozemberczki & Sarkar, 2020) and `facebook` (Rozemberczki et al., 2019) datasets, as well as for our new `amazon-ratings-full` dataset, we use random stratified 10%/10%/80% train/val/test data splits (see Appendix D for more details on the available `amazon-ratings-full` splits). The datasets `amazon-ratings-5core` and `questions` from Platonov et al. (2023b) originally come with 10 official random stratified 50%/25%/25% train/val/test data splits. However, we notice that when, as in our work, extensive hyperparameter tuning on the validation set is performed (which is very important to achieve the best GNN performance in practice), using 10 different train/val/test splits leads to a certain type of test information leakage, as nodes that appear in the test data subset in one split can appear in validation subsets in other splits. Thus, instead of using 10 data splits, we use only one data split in our experiments. Specifically, we use the first data split from the official 10 data splits.

C.2 DATASET CHARACTERISTICS

The datasets used in our experiments are diverse in domains, sizes, and graph structural characteristics. Some structural characteristics of the considered graphs are provided in Table 1. First, note that the datasets range in size from 7.6K nodes to 1.6M nodes and from 27.8K edges to 22.3M edges. We use both rather sparse and rather dense datasets, with average node degree ranging from 3.26 to 460.92 and median node degree ranging from 1 to 71. The average distance between two nodes in the considered graphs ranges from 2.45 to 194.05, while the maximum distance (graph diameter) ranges from 11 to 553. Clustering coefficients show how typical it is for two neighbors of a node to also be neighbors. There are two widely used definitions of clustering coefficients (Boccaletti et al., 2014): the global clustering coefficient and the average local clustering coefficient. In graphs used in our experiments, both clustering coefficients range from approximately zero to rather high values. Note that high values of clustering coefficients are not required for the strong performance of CLATT, as it can bring substantial benefits even on datasets with approximately zero clustering coefficients such as `city-roads-M`, `city-roads-L`, and `questions`. The degree assortativity coefficient is the Pearson correlation coefficient of degrees for pairs of linked nodes. Among the considered datasets, there are graphs with both positive, approximately zero, and negative degree assortativity coefficients.

Let us also discuss the relationships between graph structure and node labels in the considered datasets. Datasets where nodes tend to connect to other nodes with similar labels are known as *homophilous*, whereas datasets which do not exhibit this connectivity pattern are known as *non-homophilous*. To measure the levels of homophily for regression datasets, we use the target assortativity coefficient, which is the Pearson correlation coefficient of targets for pairs of linked nodes. To measure the levels of homophily for classification datasets, we use unbiased homophily (Mironov & Prokhorenkova, 2024) (with the α parameter set to 0), which satisfies more properties desirable for a homophily measure from Platonov et al. (2023a) than other commonly used homophily measures. Among the datasets used in our experiments, there is a range of both homophilous datasets (`city-roads-M`, `city-roads-L`, `city-reviews`, `pokec-regions`, `lastfm-asia`, `facebook`) and non-homophilous datasets (`avazu-ctr`, `tolokers-2`, `hm-categories`, `questions`, `amazon-ratings-5core`, `amazon-ratings-full`). As our experimental results show, CLATT can substantially improve performance on both of these types of datasets.

Note that some of the graphs considered in our experiments are directed; however, we convert all graphs to undirected ones (by duplicating each edge with its direction reversed and then removing multiple edges) for clustering, training models, and also for reporting statistics in Table 1.

D AMAZON-RATINGS-FULL DATASET

The `amazon-ratings-5core` dataset from Platonov et al. (2023b) (originally called `amazon-ratings`) was obtained from data collected by Leskovec et al. (2007). Since this dataset was used by Platonov et al. (2023b) to evaluate models specifically designed for non-homophilous graphs, many of which are not scalable, only the 5-core of the graph (Malliaros et al., 2020) was used, i.e., nodes of degree less than 5 were iteratively removed from the graph until no such nodes were left. While this procedure was used to reduce the size of the graph, we notice that it resulted in a graph with a peculiar structure: `amazon-ratings-5core` has a lot of small clusters of 5 or slightly more nodes that are very densely interconnected (often being cliques or almost cliques) but connected with the rest of the graph with only one or two edges. We hypothesize that such graph structure might be particularly amenable to node clustering and might overestimate the performance of CLATT (compared to its performance on the full co-purchasing network). Thus, we constructed a new dataset — `amazon-ratings-full` — which follows the same dataset construction process as used in Platonov et al. (2023b) but skips the reduction of the graph to its 5-core. Thus, we obtain the full Amazon co-purchasing network from Leskovec et al. (2007) (more specifically, its full largest connected component, as the original data also contains many small connected components and isolated nodes) with the same task and node features as used by Platonov et al. (2023b). As can be seen from Table 1, `amazon-ratings-full` is more than an order of magnitude larger than `amazon-ratings-5core`, while having almost the same diameter but smaller average node degree, average distance between nodes, and unbiased homophily.

When creating data splits for our new dataset, we follow the approach from Bazhenov et al. (2025) and create 3 different data splits that can be used for different purposes. Specifically, we create 2 random stratified data splits with different proportions: the `RL` (random low) data split is a 10%/10%/80% train/val/test data split, while the `RH` (random high) data split is a 50%/25%/25% train/val/test data split. Further, the `TH` (temporal high) data split is a temporal data split with exactly the same proportions as the `RH` split, which can be used to evaluate model performance under the challenging setting of temporal distributional shifts. Since the original data does not have information about the time a product appeared on Amazon, we use the time the first review for a product appeared as a proxy for it to create the temporal data split. Additionally, the `THI` (temporal high / inductive) setting can be used with the `amazon-ratings-full` dataset by considering three snapshots of the graph taken at different timestamps for training, validation, and testing. In all our experiments in this work, we use the `RL` (random low) data split for the `amazon-ratings-full` dataset, similar to our setup for datasets from the GraphLand benchmark.

It can be seen that the improvements from CLATT are typically smaller on `amazon-ratings-full` than on `amazon-ratings-5core`, which suggests that indeed the specific graph structure of `amazon-ratings-5core` is particularly beneficial for CLATT. However, even on `amazon-ratings-full`, CLATT brings statistically significant increases in model performance for all the considered models.

E EXPERIMENTAL SETUP

For our experiments, we follow the experimental protocol from Bazhenov et al. (2025). To compute the mean and standard deviation of model results, we train each model 10 times with different random seeds, except for the largest considered dataset `pokec-regions`, for which we train each model 5 times.

We train all our models in a full-batch setting as is common for GNNs, i.e., we do not use any subgraph sampling methods and train the models on the full graph (in particular, for GraphSAGE we only use the model architecture but not the neighbor sampling technique).

Hyperparameter tuning is extremely important for achieving optimal performance with GNNs. Thus, we conduct a hyperparameter search on the validation set for all models using the hyperparameter

search spaces from Bazhenov et al. (2025). For GGTs, we set the dimension of positional encodings (DeepWalk embeddings or Laplacian eigenvectors) to 128. We train all models for 1000 steps with the Adam optimizer (Kingma & Ba, 2015), except for the `amazon-ratings-5core` dataset, for which we have found longer training to often be beneficial, and thus train all models for 3000 steps.

Our model implementations use PyTorch (Paszke et al., 2019; Ansel et al., 2024) and DGL (Wang et al., 2019). All experiments were run on an NVIDIA Tesla A100 80GB GPU.

F LIMITATIONS

Graphs can be used to represent data from very different domains, and thus can exhibit a wide range of structural characteristics and relationships between graph structure, node features, and node labels. Thus, we do not expect that a single method will perform well on all realistic graphs. Yet, we show that CLATT can improve the performance of both MPNNs and GGTs (two currently most popular families of neural architectures for graph machine learning) on a wide range of graph datasets from real-world applications of graph machine learning. However, counterexamples of graphs on which CLATT does not improve performance are of course possible. We expect that the effectiveness of CLATT is correlated with the degree of the presence of meaningful cluster structure in the graph. While many graphs exhibit easily detectable cluster structure, some graphs, for example, highly regular grid-like graphs, are unlikely to contain meaningful clusters and thus probably will not benefit from CLATT. However, many real-world networks do contain useful cluster structure that CLATT can leverage to improve model performance, which we demonstrate in our experiments.

G REPRODUCIBILITY

We provide our code with the implementation of CLATT and instructions to reproduce all experimental results in our paper in the following repository: `https://anonymous.4open.science/r/cluster-attention-940C`.