GOAT: A GLOBAL TRANSFORMER ON LARGE-SCALE GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph transformers have been competitive on graph classification tasks, but they fail to outperform Graph Neural Networks (GNNs) on node classification, which is a common task performed on large-scale graphs for industrial applications. Meanwhile, existing GNN architectures are limited in their ability to perform equally well on both homophilious and heterophilious graphs as their inductive biases are generally tailored to only one setting. To address these issues, we propose GOAT, a scalable global graph transformer. In GOAT, each node conceptually attends to all the nodes in the graph and homophily/heterophily relationships can be learnt adaptively from the data. We provide theoretical justification for our approximate global self-attention scheme, and show it to be scalable to large-scale graphs. We demonstrate the competitiveness of GOAT on both heterophilious and homophilious graphs with millions of nodes.

1 INTRODUCTION

Transformers (Vaswani et al., 2017) have demonstrated efficacy in many domains including language understanding and computer vision (Devlin et al., 2018; Dosovitskiy et al., 2020), and this has spurred interest in applying transformers to the graph domain. However, the success of transformers for graphs has been more modest, and has mostly been in the fairly narrow regime of graph classification tasks like molecule classification (Ying et al., 2021; Mialon et al., 2021; Hussain et al., 2021; Dwivedi & Bresson, 2020; Rong et al., 2020; Kreuzer et al., 2021; Maziarka et al., 2021). The success of transformers in this regime is largely a result of the small size of each problem instance; the mean node count of graphs from the `ogbg-molhiv` dataset of the *Open Graph Benchmark* (Hu et al., 2020a) is only 25.5. This tiny size enables self-attention across a larger percent of the graph, enabling *long-range* or even *global* self-attention. Despite a recent surge in interest in attention-based networks, standard Graph Neural Networks (GNNs) (Kipf & Welling, 2016; Veličković et al., 2017) are still the de-facto model of choice for broader applications involving node classification or large industrial graphs.

Research in other domains suggests that a transformer's ability to utilize long-range signals that were previously inaccessible in more constrained sequential models are its key success factor. However, it is well-known that this self-attention is expensive, with time and memory overhead growing quadratically with the length of the input. To address this issue for language transformers, a range of efficient variants have been proposed (Wang et al., 2020; Kitaev et al., 2020; Zaheer et al., 2020; Zhu et al., 2021; Choromanski et al., 2020) and have demonstrated competitive performance.

When applying transformers to graphs, sequence length is akin to the size k of the k-hop neighborhood, but the size grows exponentially instead of linearly. For large enough k, the model becomes *global* and attends to the entire graph. Tasks such as node classification are usually done on large-scale graphs such as `ogbn-products` (2.4M nodes). Attending to the entirety of this graph in a naive way would require 24TB GPU memory (Geisler et al., 2021). It is an open and pressing challenge to design efficient graph transformer models that scale to graphs of this size. Existing works on graph transformers for node classification never go beyond recursive 1-*hop-neighbor* message passing (Shi et al., 2020; Zhao et al., 2021; Hu et al., 2020b) so they fail to learn from larger context and perhaps fail to achieve the full potential demonstrated in other domains.

At the same time, an established limitation of GNNs is their over-reliance on the homophily principle (McPherson et al., 2001) causing them to perform poorly on heterophilious graphs. Homophily (or heterophily) means that a node's neighbors are likely (unlikely) to be of the same class. Standard GNNs are built on the message passing scheme (Gilmer et al., 2017), where features are aggregated recursively between 1-hop neighbors. This scheme has strong inductive bias towards homophilious graphs and does not tolerate heterophily well. To address this, researchers have proposed various heterophily-centered GNN models (Lim et al., 2021; Abu-El-Haija et al., 2019; Pei et al., 2020; Zhu et al., 2020). These models usually involve specialized message passing schemes that do not work for homophilious graphs. GPR-GNN (Chien et al., 2020) can adapt to different homophily/heterophily profiles, but this model uses the entire graph for each training update ("full-batch" training) and thus cannot scale to huge problems. While it is reasonable to have specialized model designs for different kinds of graphs, this practice can be problematic when the homophily or heterophily characteristics of the target dataset are unknown, or the graph has mixed behavior.

Present work. We view a global transformer which makes each node attend to all the nodes in the graph as a universal architecture towards both homophilious and heterophilious graphs. We propose GOAT, a global transformer for large-scale node classification tasks. To implement the intractable $O(n^2)$ global self-attention on large-scale graphs, we leverage a dimensionality reduction algorithm and reduce memory complexity from quadratic to linear. Using a K-Means based projection algorithm, we theoretically show that our scalable global attention method has bounded error relative to graph attention without dimensionality reduction. Besides the global design, we also strengthen the model by a novel scalable local attention module. For each node, we sample its k-hop neighbors and make it *directly* attend to them, unlike the recursive smoothing pattern of GNNs. Empirically, GOAT shows strong performance on both homophilious and heterophilious datasets as large as `ogbn-products` (2.4M nodes) (Hu et al., 2020a) and `snap-patents` (2.9M nodes) (Lim et al., 2021). We summarize our contributions as follows:

1. We propose GOAT, a scalable global graph transformer model where each node is able to attend to all nodes in the graph.
2. We develop a novel local attention module that enables GOAT to absorb rich local information.
3. We demonstrate the strong performance of GOAT on both large-scale homophilous and heterophilious node classification benchmarks.
4. We provide theoretical justification to show our scalable global attention scheme has bounded error relative to unscalable standard attention.

2 PRELIMINARIES

In this section we introduce the preliminaries of GNNs, transformers, and homophily.

Graph Neural Networks (GNNs). We represent a graph as $\mathcal{G}(\mathcal{V}, \mathcal{E})$. GNNs are often built with recursive message-passing schemes, where features are passed and shared directly between 1-hop neighbors. Formally the k-th iteration of message passing, or the forward propogation of the k-th layer of GNNs, is defined as follows:

$$\begin{aligned} message_v^{(k)} &= \text{AGGREGATE}^{(k)}\left(\left\{\left(h_v^{(k-1)}, h_u^{(k-1)}, e_{uv}\right), \forall u \in \mathcal{N}(v)\right\}\right), \\ h_v^{(k)} &= \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, message_v^{(k)}\right), \end{aligned} \tag{1}$$

where $h_v^{(k)}$ is the hidden feature of node v at the k-th layer, e_{uv} is the edge feature between node u and v, $\mathcal{N}(v)$ is node v's 1-hop neighbor set. AGGREGATE$(\cdot)$ and COMBINE$(\cdot)$ are functions parameterized by neural networks.

Transformers. The key component of a transformer model is the self-attention scheme, which allows each element of a set or token to attend to information of other tokens at various locations. The forward pass of a self-attention module is defined as:

$$\mathrm{Attn}(H) = \mathrm{Softmax}\left(\frac{HW_Q\left(HW_K\right)^\top}{\sqrt{d}}\right)HW_V, \tag{2}$$

where $H \in \mathbb{R}^{n\times f}$ is the hidden feature matrix. $W_Q, W_K, W_V \in \mathbb{R}^{f\times d}$ are linear projection matrices with trainable weights. Here the Equation 2 denotes the single-head self-attention module, which can straightforwardly generalize to multi-head attention. Note that in practice multi-headed self-attention is widely used. It is easy to see that attention is expensive. The time and memory complexity is $O(n^2)$, which leads to low efficiency and is a bottleneck for transformers. In this work, we call the output of the Softmax function the "attention matrix."

Homophily indicates connected nodes are likely to share common labels. We follow Lim et al. (2021) to focus on edge homophily in this work. The edge homophily is defined as the proportion of edges that link two nodes with the same label as below:

$$h = \frac{|\{(u,v) \in \mathcal{E} : y_u = y_v\}|}{|\mathcal{E}|}, \tag{3}$$

where $\mathcal{E}$ is the edge set and y is the node label. Non-homophily (or **heterophily**) graph indicates the dataset with dissimilar labels/features sharing edges.

3 METHOD

Intuition. Existing GNNs usually have hardcoded message-passing patterns and will only work on either homophilious or heterophilious graphs. A global attention scheme makes each node attend to all the nodes in the graph and does not explicitly have inductive bias towards either one. Instead of one specially tailored or fixed message passing and aggregating pattern, the attention scheme freely learn to adapt to different priors. In addition, intuitions from the NLP literature make it clear that *long-range* self-attention derives much of its efficacy from the larger and more informative contexts it incorporates. Thus, faithfully we have reason to believe that *global* transformer increases expressive power by modeling *long-range* interactions over the whole graph and can be a universal architecture which makes accurate predictions on both homophilious and heterophilious graphs. However, for applications in industry where large graphs with millions of nodes are ubiquitous, it is not possible to train and deploy a fully global transformer due to the quadratic overhead.

Given that, we propose **GOAT**, the scalable global transformer. GOAT uses an approximate global attention which reduces complexity from quadratic to linear and supports mini-batch training. GOAT also has a local attention module to process information from the local neighborhood for better prediction. Fig 1 illustrates the local sampling procedure and the whole attention module. In this section, we describe our methodological designs in detail.

Global. To realize the idea of global attention, we intuitively propose to use a *codebook* $\mu \in \mathbb{R}^{k\times f}$ with constant scale k, which is the outcome of dimensionality reduction of hidden node features, to represent all the nodes in the graph. During the forward pass, each node will only be attending to features stored in the codebook to approximate the attention computed by authentically attending to the whole graph. Then we have the complexity relaxation from $O(n^2)$ to $O(nk)$. Remember that the codebook needs to reflect the hidden neural node features which are evolving with the progress of training. So a novel dimensionality reduction scheme that can handle dynamic features is required.

Technically, we leverage the Exponential Moving Average (EMA) K-Means algorithm for dimension reduction on the fly during the model training process. The algorithm is summmarized in Algorithm 2. The codebook represents the centroids of the K-Means algorithm and is consistently materialized and updated in GPU. Each node is assigned to a specific centroid and such mapping is determined by finding the nearest centroid with respect to each node. Instead of hardcoding the new centroids as the mean vector of all the node features belonging to each centroid as in the plain K-Means, we update them in the EMA manner. Such practice is necessary because the momentum of the centroid needs to be kept to represent the nodes that belong to the centroid but is not updated in the current iteration due to mini-batch sampling. Hyperparameter γ is used to control the momentum. One further improvement we have is using a batch norm module to whiten the data. Also, before returning

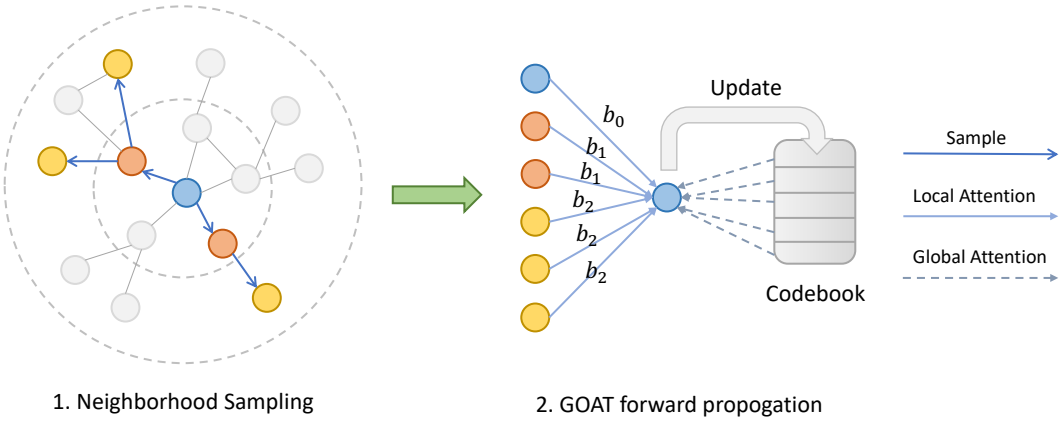


Figure 1: The local sampling procedure and forward propogation of the GOAT model. b_k denotes the trainable positional bias for neighbors at a distance of k.

Algorithm 1 Global Transformer mini-batch forward propogation algorithm

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; $\mathrm{MLP_a}$ and $\mathrm{MLP_b}$ are two independent MLPs; PE is the precomputed positional encoding based on the structure of graph $\mathcal{G}$; μ_x, μ_{pe} are the centroids computed by the K-Means algorithm; P is the centroid assignment index for each node; W_Q, W_K, W_V are trainable parameters.

for $v \in \mathcal{V}$ **do**
  $x \leftarrow \mathrm{MLP_a}(X_v)$
  $pe \leftarrow PE_v$
  $q \leftarrow \mathrm{Concat}(x, pe)W_Q$
  $K \leftarrow \mathrm{Concat}(\mu_x, \mu_{pe})W_K$
  $V \leftarrow \mu_x W_V$
  $x_{out} \leftarrow \mathrm{Softmax}\left(\frac{qK^\top}{\sqrt{d}} + \log\left(\mathbf{1}_n P\right)\right) V$
  $x_{out} \leftarrow \mathrm{MLP_b}(x_{out})$
  Update μ_x, μ_{pe} by x, pe using the EMA K-Means algorithm as in Algorithm 2.

the centroids for use, they will be mapped back into the original space using the running mean and std values of batch norm. We find this technique stablizes the training of model.

In addition, it is well established in the literature that good positional encodings are required to make transformers work effectively. In our global attention scheme, only absolute positional encodings are feasible because the hidden features and positional encodings must be concatenated. We argue that the design of positional encoding on graphs is still an open question for the community (Dwivedi et al., 2021; Kreuzer et al., 2021) and detailed discussion for such design is beyond the scope of this work. To simplify the setup of the experiments, we use pretrained node2vec (Grover & Leskovec, 2016) node embeddings as our positional embeddings. We summarize the forward pass computation of the global module in Algorithm 1. Note that the key attention computation in Algorithm 1 is formulated as $x_{out} = \mathrm{Softmax}\left(\frac{qK^\top}{\sqrt{d}} + \log\left(\mathbf{1}_n P\right)\right) V$, where $\log\left(\mathbf{1}_n P\right)$ is used to adjust the attention scores according to the size of different clusters. In Section 4, we provide in-depth theoretical justifications to show the solidity of our global module and prove that the approximated global attention has bounded estimation error compared with the fully global attention.

Local. Along with the design of our global transformer, we propose a novel local attention module, which allows each node to *directly* attend to its k-hop neighbors to attain rich local information. Empirically, we find that the local attention module effectively helps the model learn better representations. Although this module is responsible for aggregating information from the *local* neighborhood as in normal message passing feature aggregation, unlike the standard GNNs' recursive 1-hop neighbor smoothing pattern, our local module provides flexible attention weights for neighbors at different hop distances. We believe the module provides increased capacity to learn the inductive biases required

Algorithm 2 EMA K-Means update algorithm

Require: Inputs are the hidden features X and positional encodings PE for a batch. $\mathrm{bn}(\cdot)$ is the batch norm module. $\mathrm{FindNearest}(\cdot)$ finds the nearest centroid for each feature.

function UPDATE(X, PE)
- $F \leftarrow \mathrm{Concat}(X, PE)$
- $F \leftarrow \mathrm{bn}(F)$ ▷ data whitening using batch norm
- $\mu \leftarrow \mathrm{Concat}(\mu_x, \mu_{pe})$
- $P \leftarrow \mathrm{FindNearest}(F, \mu)$ ▷ compute cluster assignment
- $c \leftarrow c \cdot \gamma + P^\top \mathbf{1} \cdot (1-\gamma)$ ▷ EMA accumulation
- $v \leftarrow v \cdot \gamma + P^\top F \cdot (1-\gamma)$ ▷ EMA accumulation
- $v \leftarrow v/c$
- $\mu \leftarrow v \cdot \mathrm{bn.running\_std} + \mathrm{bn.running\_mean}$ ▷ project the centroids back
- $\mu_x, \mu_{pe} \leftarrow \mu$

for accurate predictions beyond the hardcoded structures of existing GNNs. To support mini-batch training, we adopt the widely-used neighbor sampling (NS) method (Hamilton et al., 2017) to sample all k-hop neighbors. We make each node directly attend to the sampled neighbors. We refer readers to Fig 1 for a clear view of the local module. For this module we select the *relative* positional encoding scheme to distinguish neighbors at diverse distances from the source node. Our attention scores inside the Softmax function is: $S_{ij} = X_i W_Q \left(X_j W_K\right)^\top / \sqrt{d} + b_{D(i,j)}$, where $b_{D(i,j)}$ is a trainable bias parameter indexed by $D(i,j)$, the shortest distance between node i and j. Since we leverage different positional encodings for the global and local modules, we have separate attention functions, after which the respective sets of node features and positional encodings are concatenated and fed into subsequent layers.

4 THEORETICAL JUSTIFICATION

Our global module addresses expensive quadratic complexity issue of global attention. We formulate the usage of codebook in the global module as *dimensionality reduction*. Specifically, we want to find a projection matrix P that can reduce the feature matrices into a low-dimensional space so the cost of computing their product is reduced. At the same time we also expect such dimensionality reduction will not overly degrade the quality of the outcome. Below, we provide theoretical analysis on the existence of such a P matrix. Note that in the demonstration process below, we assume positional information is already fused into node features for simplicity.

Proposition 1. *There exists a distribution of random projection matrices $P \in \mathbb{R}^{n \times k}$ such that for any linear weight matrices $W_K, W_Q, W_V \in \mathbb{R}^{f \times d}$, node feature matrix $X \in \mathbb{R}^{n \times f}$, mini-batch of nodes $X_B \in \mathbb{R}^{b \times f}$, column vector $v \in \mathbb{R}^n$ of XW_V, and any choice of $\varepsilon > 0$,*

$$\Pr\left(\|A_B P P^\top v - A_B v\|_F < \varepsilon \|A_B v\|_F\right) > 1 - O(1/n), \tag{4}$$

with $A_B = \mathrm{Softmax}\left(X_B W_Q \left(XW_K\right)^\top / \sqrt{d}\right)$ and $k = O\left(\log(n)/\varepsilon^2\right)$.

Our goal here is to find a projection matrix P to project both the attention matrix A_B and features v into some low dimension space, enabling us to replace the attention matrix $A_B \in \mathbb{R}^{b \times n}$ with $\widetilde{A}_B = A_B P \in \mathbb{R}^{b \times k}$ and $v \in \mathbb{R}^n$ with $\widetilde{v} = P^T v \in \mathbb{R}^k$. Despite the existence of such P matrix, we argue that it is impractical to apply Proposition 1 in practice. The reason is that it is hard to materialize $\widetilde{A}_B$ in memory so the explicit computation of A_B is inevitable, which does not help improve scalability as calculating all attention matrices batch-by-batch is still $O(n^2)$. Below, we describe a route to escape this problem.

Theorem 1. *For any linear weight matrices $W_K, W_Q, W_V \in \mathbb{R}^{f \times d}$, node feature matrix $X \in \mathbb{R}^{n \times f}$, mini-batch of nodes $X_B \in \mathbb{R}^{b \times f}$, row vector $x \in \mathbb{R}^{1 \times n}$ of X, and $\varepsilon > 0$, there exists projection matrices $P_A, P_V \in \mathbb{R}^{n \times k}$, such that*

$$\Pr\left(\left\|\mathrm{Softmax}\left(SP_A\right) P_V^\top XW_V - \mathrm{Softmax}(S) XW_V\right\|_F \leqslant \varepsilon \|\,\mathrm{Softmax}(S)\|_F \,\|XW_V\|_F\right) > 1 - O(1/n), \tag{5}$$

with $S = X_B W_Q \left(XW_K\right)^\top / \sqrt{d}$ *and* $k = O\left(\log(n)/\varepsilon^2\right)$.

We argue that the scheme provided by Theorem 1 makes global attention possible on large graphs. The expression $SP_A = \left(X_B W_Q \left(XW_K\right)^\top / \sqrt{d}\right) P_A$ inside the Softmax function can be computed as a product between $Q_B = X_B W_Q$ and $\widetilde{K} = P_A^\top X W_K$, which is only $O(bk)$ work. Note that the new value feature matrix $\widetilde{V} = P_V^\top X W_V$ is also low dimensional. Then we see that the problem comes to how we materialize $\widetilde{K}, \widetilde{V}$ without explicitly computing the multiplication of $(XW_K)^\top P_A$ or $P_V^\top X W_V$ which is $O(nfd + nkd)$ and not scalable when the graph is large. In our global module, $\text{diag}^{-1}(\mathbf{1}_n P) P^\top X$ is the *codebook*. Projection matrix P is computed by the K-Means algorithm as the sparse indexing matrix. Each row of $P \in \mathbb{R}^{n \times k}$ is a one-hot vector representing the clustering centroid the node is assigned to. In this way, $\widetilde{K} = \text{diag}^{-1}(\mathbf{1}_n P) P^\top X W_K$ and $\widetilde{V} = \text{diag}^{-1}(\mathbf{1}_n P) P^\top X W_V$. Note that we do not explicitly compute $\widetilde{K}, \widetilde{V}$ on each iteration but instead cache and update the codebook on the fly in the EMA manner using batch statistics. P is stored sparsely to save memory. Now we go back to show that the forward pass under such a scheme yields a bounded-error approximation compared with the authentic output. We start off with two definitions for clearer demonstration. We define $\widetilde{X} = P\text{diag}^{-1}(\mathbf{1}_n P) P^\top X$, as the approximate X using the codebook. We define the attention matrix for a batch X_B computed by parameterized function f_W as $f_W(X) = \text{Softmax}\left(X_B W_Q \left(XW_K\right)^\top / \sqrt{d}\right)$.

Theorem 2. *If the function f_W has Lipschitz constant upper-bounded by* $\text{lip}\left(f_W\right)$ *and the quantization error is ε, the estimation error is bounded as,*

$$\left\| \widetilde{X}_B^{out} - X_B^{out} \right\|_F \leq \varepsilon \cdot \left[1 + O\left(\text{lip}\left(f_W\right)\right)\right] \|A_B\|_F \cdot \|X\|_F \cdot \|W_V\|_F, \tag{6}$$

with $\widetilde{X}_B^{out} = \widetilde{A}_B \widetilde{X} W_V$, $X_B^{out} = A_B X W_V$, $A_B = f_W(X) = \text{Softmax}\left(X_B W_Q \left(XW_K\right)^\top / \sqrt{d}\right)$, *and* $\widetilde{A}_B = f_W(\widetilde{X}) = \text{Softmax}\left(X_B W_Q (\widetilde{X} W_K)^\top / \sqrt{d}\right)$. *Quantization error ε is defined as* $\|X - \widetilde{X}\|_F \leq \varepsilon \|X\|_F$.

Note that the computation of $\widetilde{A}_B$ denoted in Theorem 2 is expensive because it requires the multiplication of $X_B W_Q$ and $\widetilde{X} W_K$ and the cost is $O(bn)$. We articulate it in this way for the simplicity of theoretical proof. Notice that $\widetilde{X}_B^{\text{out}}$ can be reformulated as $\widetilde{A}_B P \widetilde{K}$. $\widetilde{K}$ is low dimensional. P here works to sum the same Softmax weights belonging to the same centroid. We intend to put the summation operation inside the Softmax function and we can achieve for any row vector x of X,

$$\begin{aligned} &\text{Softmax}\left(xW_Q \left(\text{diag}^{-1}\left(\mathbf{1}_n P\right) P^\top X W_K\right)^\top / \sqrt{d} + \log\left(\mathbf{1}_n P\right)\right) \\ &\qquad = \text{Softmax}\left(xW_Q \left(P\,\text{diag}^{-1}\left(\mathbf{1}_n P\right) P^\top X W_K\right)^\top / \sqrt{d}\right) P = \widetilde{A}_B P, \end{aligned} \tag{7}$$

where $\log\left(\mathbf{1}_n P\right)$ stores the size of each cluster and the log function will be canceled during exp calculation in Softmax. In that way the summation is done inside Softmax and we have $\widetilde{x}_{\text{out}} = \text{Softmax}\left(xW_Q \widetilde{K}^\top / \sqrt{d} + \log\left(\mathbf{1}_n P\right)\right) \widetilde{V}$, where x gets to directly attends to $\widetilde{K}$ instead of $\widetilde{X}$. As we have discussed, both $\widetilde{K}$ and $\widetilde{V}$ will be materialized in memory and updated in the EMA way. The computation for the feature of a node $\widetilde{x}_{\text{out}}$ is $O(k)$ and thus scalable. All the proofs are deferred to the Appendix A.1.

5 Experiments

In this section, we detail the empirical evaluation of our GOAT model.

Datasets. We select four datasets to evaluate. To represent homophilious graph problems we choose two datasets, `ogbn-arxiv` and `ogbn-products`, from the well-known *Open Graph Benchmark* (OGB) (Hu et al., 2020a). For heterophilious examples we utilize the `arxiv-year` and `snap-patents` datasets curated by Lim et al. (2021). These are all large-scale (multi-million) node classification datasets. For the train and validation splits, we use the official splits from OGB

Table 1: Dataset statistics.

Dataset	# Nodes	# Edges	# Feat.	# Class	Class type	Split	Edge hom.
ogbn-arxiv	169,343	1,166,243	128	40	product category	official	.66
ogbn-products	2,449,029	61,859,140	100	47	subject area	official	.81
arxiv-year	169,343	1,166,243	128	5	pub year	random	.222
snap-patents	2,923,922	13,975,788	269	5	time granted	random	.073

for both `ogbn-arxiv` and `ogbn-products` and for `arxiv-year` and `snap-patents` we follow the practice of Lim et al. (2021) and randomly sample the train and validation sets. As there are no official splits or train set ratios for these two datasets, we experiment with training sets that comprise 10%, 20%, and 50% of the data while fixing validation set ratio at 25%, and report separate results for each split. We refer readers to Table 1 for detailed statistics of the datasets.

Setup. We focus on the transductive node classification task, where we see all the nodes at training time, but only the train set has labels. Our baseline models include GCNJK (Xu et al., 2018), GAT (Veličković et al., 2017), LINKX (Lim et al., 2021), and MixHop (Abu-El-Haija et al., 2019). All of our training procedures use pure empirical risk minimization (ERM) and we do not use techniques like data augmentation (Kong et al., 2020), label propogation (Huang et al., 2020), powerful embeddings (Chien et al., 2021), or other tricks, as these additional regularizers have not been uniformly studied for all model types, and this simple setting enables fair comparisons across model architectures. For the baseline models, we perform a hyperparameter sweep and select the best performing settings and report the corresponding results for each model and dataset pairing (full details in the Appendix). For GOAT, the attention function is multi-headed but we only implement a single layer of attention module. We leave the discussion of a multi-layer variant to future work. For the local attention, we sample neighbors that live within 3-hops. For each node we sample [20, 10, 5] neighbors recursively. The size of the codebook is fixed at $4,096$ and the dimensionality is 64. We always use a dropout rate of 0.5 and also use batch norm. Each experiment is carried out on a either a single GeForce RTX 2080 Ti (11GB memory) or RTX A4000 (16GB memory).

Results. Table 2 reveals the competitive performance of our GOAT model compared with baselines. GCNJK and GAT intuitively register strong performances on `ogbn-arxiv` and `ogbn-products` as they resonate with the inductive bias of the two homophilious datasets. In contrast, on the `arxiv-year` and `snap-patents` datasets, their scores are poor. LINKX and MixHop are architectures designed for heterophilious graphs, a specialization validated by their strong performances in Table 2 on those datasets. However, GOAT constantly achieves competitive results on *all four* datasets, which reveals the adaptive ability of the attention functions. We highlight that our goal is not to beat GNNs universally; specific priors can still be useful factors when designing architectures. Further, in accordance with the no-free-lunch theorem, winning all comparisons on datasets with diverse properties is expected to be difficult. Rather, our intention is to develop a model that can learn different inductive biases as required, adapting seamlessly to different use cases. This flexibility is an important quality in practice as practitioners may encounter datasets for which knowledge about appropriate priors is scarce.

6 Ablation Studies & Analysis

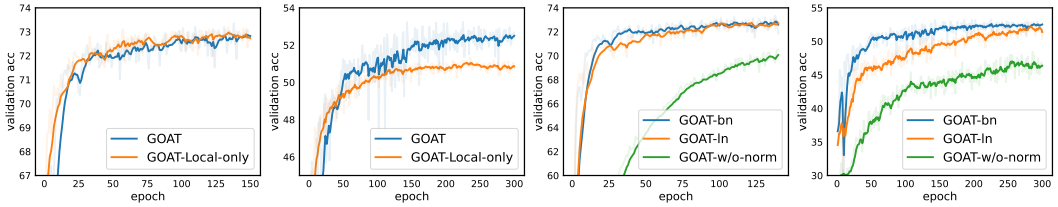
GOAT vs. GOAT-Local-only. In Fig 2a & 2b we ablate to only use the local module to analyze our architecture design. On the `ogbn-arxiv` dataset where homophily is prevalent, the local-only model is as strong as the complete GOAT. While on the `arxiv-year` dataset there shows clear discrepancy between the two, where the global module brings a salient 3% boost. The results provide a strong validatioin towards our main intuition, that the global module increases expressive power by modeling long-range interactions. And such ability is especially effective towards heterophilious graphs, which is intuitive as many other successful heterophilious GNN architectures aimed at broadening the range of message passing as well as learning from long-range interactions.

Batch norm vs. Layer norm. Normalization is important to transformers. We ablate on the normalization technique selection in Fig 2c & 2d. We see that layer norm can be as good as batch

Table 2: Experimental results. Note that the "Overall performance average" is the mean value of the two "Performance avgerage" values in the upper and lower tables. The darker blue marks the best result in each row, while the lighter shade marks the runner-up. Evaluation metric is prediction accuracy on the test set. Test accuracies are reported based on the hyperparameter setting yielding the highest validation accuracies. Where possible, we validate our baselines against the results in Lim et al. (2021).

		General GNN		Heterophilious GNN		Ours
Dataset	Train ratio	GCNJK	GAT	LINKX	MixHop	GOAT
ogbn-arxiv	official 54%	69.57 ± 0.20	71.95 ± 0.36	66.18 ± 0.33	71.29 ± 0.29	72.41 ± 0.40
ogbn-products	official 8%	72.84 ± 0.36	79.45 ± 0.59	71.59 ± 0.71	73.48 ± 0.29	82.00 ± 0.43
Performance average		71	76	69	72	77
arxiv-year	random 10%	43.34 ± 0.08	38.34 ± 0.10	46.22 ± 0.24	45.13 ± 0.25	49.44 ± 0.11
arxiv-year	random 20%	44.77 ± 0.10	39.19 ± 0.12	49.16 ± 0.42	47.18 ± 0.24	51.21 ± 0.44
arxiv-year	random 50%	47.74 ± 0.23	40.27 ± 0.20	53.53 ± 0.36	50.37 ± 0.25	53.57 ± 0.18
snap-patents	random 10%	32.50 ± 0.10	32.72 ± 0.10	49.74 ± 0.46	33.57 ± 0.06	44.31 ± 0.43
snap-patents	random 20%	32.97 ± 0.06	32.96 ± 0.09	54.32 ± 0.50	33.96 ± 0.06	49.55 ± 0.31
snap-patents	random 50%	33.52 ± 0.05	33.10 ± 0.09	60.12 ± 0.23	34.28 ± 0.07*	54.97 ± 0.23
Performance average		39	36	52	41	51
Overall performance average		55	56	61	57	64

For * we show performance of MixHop with GraphSAINT sampling (Zeng et al., 2019) on snap-patents despite Lim et al. (2021) reporting 46.82 ± 0.11 for MixHop with ClusterGCN sampling (Chiang et al., 2019) as their runner-up to LINKX on this dataset. This setting was OOM/T on our hardware, but as it is weaker than LINKX and GOAT in either case, reporting the stronger numbers would not have changed the results or analysis of average model performance.



(a) ogbn-arxiv (b) arxiv-year (c) ogbn-arxiv (d) arxiv-year

Figure 2: Ablation studies.

norm in our model but converges much slower, and evidently "no normalization" is not a good choice for our architecture.

Depth. We constantly have 1 layer of attention modules in our model, for both the global and local part. In fact it is challenging to extend our GOAT model to multi-layer attention computation like common transformers. Note that our attention modules are not strict *self-attention* modules, whose input and output are of the same length. As denoted in Fig 1, our input includes the predictive node, sampled neighbors, and the codebook, but only the the predictive node attends to others and gets aggregated features. So after the attention function the state of feature for sampled neighbors and the codebook is behind that of the predictive node, where future rounds of attention computation is less intuitive. A straightforward strategy to address this conundrum is to recursively sample neighbors of neighbors and add more codebooks by each layer, but apparently this will greatly adds to the overhead and renders the model less efficient. Due to computational limitation we do not carry out the experiments, but how to make the model deeper is an interesting research opportunity.

Limitations. The neighbor sampling bottleneck is the major limitation of our method. Note that although the local module intuitively helps learn better representations, the neighbor sampling (NS)

algorithm does introduce a small scalability concern. It is well-known that the neighborhood explosion problem for NS has complexity $O(d^k)$ for k-hop neighborhoods, where d is the average degree. Our sampling method in the local module is a plain adaptation of NS so that GOAT will share efficiency issues of NS.

7 Related Work

Scalability. The poor scalability of GNNs to large graphs remains a major obstacle to deploying GNNs for enterprise-scale applications. Scalable GNN algorithms mainly involve node-, layer-, and graph-wise sampling methods. Hamilton et al. (2017) proposed neighbor sampling (NS) to repeatedly sample neighbors for message passing, but this approach leads to the exponential neighborhood explosion problem mentioned at the end of Section 3. One technique to address this issue is the layer-wise sampling proposed in Huang et al. (2018). ClusterGCN (Chiang et al., 2019) and GraphSAINT (Zeng et al., 2019) both perform subgraph sampling so that GNNs can be run in a "full-batch" manner but on tractable subgraphs that hopefully approximate the global graph semantics - we employ the latter of the two for training our baseline GCNJK, GAT, and MixHop models. A recent method, GAS (Fey et al., 2021), stores historical node features to help both training and inference process. Finally, it is also established that transformers suffer from inherent scalability issues. In response, a plethora of efficient transformers have been proposed that employ different techniques to improve their complexity including sparse attention maps (Zaheer et al., 2020; Beltagy et al., 2020), clustering-based schemes (Kitaev et al., 2020; Tay et al., 2020), and low-rank projections of the attention matrix (Wang et al., 2020; Tay et al., 2021).

Graph transformers. Most successful applications of transformers to graphs problems have only considered graph classification. Graphormer (Ying et al., 2021) is the representative global transformer on small graphs with customized centrality, edge, and spatial encodings. Other strong architectures include Kreuzer et al. (2021); Maziarka et al. (2021); Dwivedi & Bresson (2020); Rong et al. (2020). Existing graph transformers (Shi et al., 2020; Hu et al., 2020b; Dwivedi & Bresson, 2020) for node classification use the same recursive message passing scheme as traditional GNNs. The attention module computes attention scores as the weights for 1-hop neighborhood smoothing, similar to the attention mechanism in GATs (Veličković et al., 2017). These models fail to demonstrate the ability to learn from larger contexts and generally do not outperform traditional GNNs. An example architecture that does implement a form of global attention, GraphBERT (Zhang et al., 2020), fails to solve scalabilty issues due to requiring full-batch training.

Heterophily. Recently a body of work has focused on adapting GNNs to heterophilious graphs. LINKX (Lim et al., 2021) is a simple model that coerces 2-hop (but not 1-hop) neighbors to have the similar labels. H2GCN (Zhu et al., 2020) aggregates features from 1-hop and 2-hop neighbors separately. MixHop (Abu-El-Haija et al., 2019) aggregates information from diverse degrees of smoothed features. GPR-GNN (Chien et al., 2020) is similar to MixHop, but uses a more complex adaptive aggregation operation. GPR-GNN works well on some kinds of data but does not scale to large graphs. One closely related work is Non-local GNN (Liu et al., 2021), which does an approximate global attention leveraging attention-based sorting. The model successfully reduces time complexity from $O(n^2)$ to $O(nlog(n))$, but cannot support mini-batch training, so its scalability is limited.

While progress has certainly been made towards more effective graph transformers and models specifically suited to heterophilious graphs, to date, no graph transformer has emerged that simultaneously relaxes the restriction of a homophilious inductive bias (whilst remaining performant in that setting) and easily handles large-scale node classification tasks.

8 Conclusion

We propose GOAT, a global transformer that works on both homophilious and heterophilious node classification tasks. Our model makes global attention possible by dimensionality reduction and we prove our approximate approach has bounded error compared with the true global mechanism. Experiments reveal GOAT's strong performances on large-scale graphs. We hope our work can spur more research into universal graph neural architecture who can adapt to different inductive biases.

9 ETHICS STATEMENT

Regarding the potential negative social impact, our model is potentially useful in anomaly detection since graphs comprising attackers and threats are typically heterophilious. Our method could be studied by malicious actors to subvert transformer based detectors.

10 REPRODUCIBILITY

We list our contributions towards reproducibility as below:

- Implementation open source. We include our implementation of the GOAT model in the supplement materials.
- Hyperparameter. The hyperparameter settings for both GOAT and baselines are discussed in Appendix A.2.
- Datasets. Ways to download datasets are shown in Appendix A.2.
- Proofs. Theoretical proofs of our proposition and theorems can be found in the Appendix A.1.

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.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 257–266, 2019.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988*, 2020.

Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S Dhillon. Node feature extraction by self-supervised multi-scale neighborhood prediction. *arXiv preprint arXiv:2111.00064*, 2021.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*, 2021.

Matthias Fey, Jan E Lenssen, Frank Weichert, and Jure Leskovec. Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings. In *International Conference on Machine Learning*, pp. 3294–3304. PMLR, 2021.

Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems*, 34, 2021.

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.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

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. *Advances in neural information processing systems*, 33:22118–22133, 2020a.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pp. 2704–2710, 2020b.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993*, 2020.

Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. *Advances in neural information processing systems*, 31, 2018.

Md Shamim Hussain, Mohammed J Zaki, and Dharmashankar Subramanian. Edge-augmented graph transformers: Global self-attention is enough for graphs. *arXiv preprint arXiv:2108.03348*, 2021.

William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space 26. *Contemporary mathematics*, 26:28, 1984.

Daniel M Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):1–23, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Gavin Taylor, and Tom Goldstein. Flag: Adversarial data augmentation for graph neural networks. *arXiv preprint arXiv:2010.09891*, 2020.

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

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems*, 34, 2021.

Meng Liu, Zhengyang Wang, and Shuiwang Ji. Non-local graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Łukasz Maziarka, Dawid Majchrowski, Tomasz Danel, Piotr Gaiński, Jacek Tabor, Igor Podolak, Paweł Morkisz, and Stanisław Jastrzębski. Relative molecule self-attention transformer. *arXiv preprint arXiv:2110.05841*, 2021.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33:12559–12571, 2020.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pp. 9438–9447. PMLR, 2020.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *International Conference on Machine Learning*, pp. 10183–10192. PMLR, 2021.

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. *arXiv preprint arXiv:1710.10903*, 2017.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5453–5462. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/xu18c.html`.

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, 2021.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*, 2020.

Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. Gophormer: Ego-graph transformer for node classification. *arXiv preprint arXiv:2110.13094*, 2021.

Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. Long-short transformer: Efficient transformers for language and vision. *Advances in Neural Information Processing Systems*, 34, 2021.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

A APPENDIX

A.1 PROOFS

This section lists theoretical proofs of our propositions and theorems.

A.1.1 PROOF OF PROPOSITION 1

Proof. To help build the proof, we utilize the Johnson–Lindenstrauss (JL) Lemma from Johnson & Lindenstrauss (1984); Kane & Nelson (2014).

Lemma 1. *For any integer $d > 0$, and any $0 < \varepsilon$, $\delta < 1/2$, there exists a probability distribution on $k \times d$ real matrices for $k = \Theta(\varepsilon^{-2} \log(1/\delta))$ such that for any $x \in \mathbb{R}^d$,*

$$\mathop{\mathbb{P}}_{P}((1-\varepsilon)\|x\|_2 \leq \|Px\|_2 \leq (1+\varepsilon)\|x\|_2) > 1-\delta. \tag{8}$$

Using Lemma 1 and Boole's inequality, for any $x, y \in \mathbb{R}^{1\times n}$, we have

$$\Pr(\|xPP^\top y^\top - xy^\top\|_F \leq \varepsilon\|xy^\top\|_F) > 1-2\delta.$$

Now we choose to swap x with any row vector a out of A_B and y with v so we have

$$\Pr(\|aPP^\top v^\top - av^\top\|_F \leq \varepsilon\|av^\top\|_F) > 1-2\delta.$$

Leveraging Boole's inequality, we get

$$\Pr(\|A_BPP^\top v^\top - A_Bv^\top\|_F \leq \varepsilon\|A_Bv^\top\|_F) > 1-2b\delta > 1-2n\delta.$$

By choosing δ to be $O(1/n^2)$ we finally get

$$\Pr(\|A_BPP^\top v^\top - A_Bv^\top\|_F \leq \varepsilon\|A_Bv^\top\|_F) > 1-O(1/n),$$

with $k = O\left(\log(n)/\varepsilon^2\right)$.

□

A.1.2 PROOF OF THEOREM 1

Proof. We follow the techniques in Wang et al. (2020) to provide the proof. We define $P_A = \delta P$ and $P_V = e^{-\delta}P$, δ is a constant. We aim to prove

$$\Pr\left(\left\|\exp\left(SP\right)P^\top X_V - \exp(S)X_V\right\|_F \leq \varepsilon\|\exp(S)\|_F\left\|X_V\right\|_F\right) > 1-O(1/n), \tag{9}$$

where $X_V = XW_V$ for simplicity. Using triangle inequality, we have

$$\begin{aligned}\left\|\exp\left(SP\right)P^\top X_V - \exp(S)X_V\right\|_F \leq &\underbrace{\left\|\exp(S)PP^\top X_V - \exp(S)X_V\right\|_F}_{a} \\ &+ \underbrace{\left\|\exp\left(SP\right)P^\top X_V - \exp(S)PP^\top X_V\right\|_F}_{b}.\end{aligned}$$

For part a, based on Proposition 1, we have

$$a \leq \varepsilon\|\exp(S)\|_F\left\|X_V\right\|_F.$$

For part b, we have

$$b \leq \|\exp(SP) - \exp(S)P\|_F \, \|P^\top X_V\|_F.$$

Given that the exponential function is Lipschitz continuous in a compact region, and also based on the Equation 1, we have

$$b \leq o(\|\exp(S)\|_F \|X_V\|_F).$$

Based on part a and part b, we finally attain

$$\Pr\left(\left\|\exp(SP)\, P^\top X_V - \exp(S) X_V\right\|_F \leq \varepsilon \|\exp(S)\|_F \, \|X_V\|_F\right) > 1 - O(1/n).$$

□

A.1.3 Proof of Theorem 2

Proof. This proof is built upon the usage of Lipschitz constant and quantization error.

$$\begin{aligned}
\left\|\widetilde{X}_B^{\text{out}} - X_B^{\text{out}}\right\|_F &= \|\widetilde{A}_B \widetilde{X} W_V - A_B X W_V\|_F \\
&= \|f_W(\widetilde{X})\widetilde{X} W_V - f_W(X) X W_V\|_F \\
&\leq \underbrace{\|f_W(\widetilde{X})\widetilde{X} - f_W(X)X\|_F}_{a} \|W_V\|_F,
\end{aligned}$$

where for the part a, we have

$$\begin{aligned}
a &= \|f_W(\widetilde{X})\widetilde{X} - f_W(X)\widetilde{X} + f_W(X)\widetilde{X} - f_W(X)X\|_F \\
&\leq \|f_W(\widetilde{X}) - f_W(X)\|_F \|\widetilde{X}\|_F + \|f_W(X)\|_F \|\widetilde{X} - X\|_F.
\end{aligned}$$

Note that

$$\|\widetilde{X}\|_F \leq \|P \text{diag}^{-1}(\mathbf{1}_n P) P^\top\|_F \|X\|_F \leq \sqrt{k}\|X\|_F,$$

$$\|X - \widetilde{X}\|_F \leq \varepsilon \|X\|_F,$$

$$\|f_W(\widetilde{X}) - f_W(X)\|_F \leq \text{lip}\,(f_W)\, \|X - \widetilde{X}\|_F.$$

When we assume $\|X\|_F = O(\|A_B\|_F)$, we have

$$\left\|\widetilde{X}_B^{\text{out}} - X_B^{\text{out}}\right\|_F \leq \varepsilon \cdot [1 + O\,(\text{lip}\,(f_W))]\, \|A_B\|_F \cdot \|X\|_F \cdot \|W_V\|_F\,.$$

□

A.2 Experimental Details

This section describes in more detail the experimental setup for the empirical results presented in Section 4.

A.2.1 Dataset Downloading

We refer readers to Hu et al. (2020a) and Lim et al. (2021) and their official repos for dataset downloading.

Table 3: Proposed Model Dataset-Specific Hyperparameter Settings

Model	Dataset	# Heads	# Hidden Channels
GOAT	ogbn-arxiv	4	128
	ogbn-products	2	256
	arxiv-year	4	128
	snap-patents	2	128

A.2.2 GOAT

For the hyperparameter selections of our GOAT model, besides what we have covered in the setup part of the experiment section that datasets share in common, we list other settings in Table 3. Each experiment is repeated four times to get the mean value and error bar. We use Adam optimizer with lr 1e-3.

A.2.3 BASELINE MODELS

Since the heterophilius datasets on which we benchmark the GOAT model are derived from Lim et al. (2021), in order to facilitate as fair a comparison as possible, especially against the LINKX model proposed in the same work, we utilize their implementation provided at the official repo[1]. This library also provides reference implementations of other GNN architectures, and we utilize those as well. Following the procedure described both in the paper and implicitly by their codebase, we run a hyperaparameter sweep for each model on each dataset. For each combination of parameters, 5 models are trained using different initialization seeds to determine a mean value and error bars, and then the final hyperparameters are selected based on accuracy on the validation set. These final parameters correspond to the settings used to realize the "official train split" numbers in Table 2 in the main work for `ogbn-arxiv` and `ogbn-products` and the 50% train split numbers for `arxiv-year` and `snap-patents`. These same parameters are also used when performing the sample complexity experiments with train splits of 10% and 20% for the latter two datasets.

For the baseline GNNs we chose a Graph Convolutional Network with Jumping Knowledge (GCNJK) and a Graph Attention Network (GAT), and for a second heterophily-specific model to complement LINKX, we consider MixHop. As described in Section 4 of the main work, we use ERM to train all models including the baselines. We also focus on the minibatch setting rather than "full-batch" training as a primary feature of the GOAT model is its native scalability through minibatch training. The batching algorithm we use for the GCNJK, GAT, and MixHop models is the GraphSAINT Random Walk based sampler (LINKX uses its own adjacency row-wise sampling scheme, see Lim et al. (2021) for details). In the spirit of fair evaluation, we make the model and sampler choices based on the fact that according to Lim et al. (2021), each of these are the most performant two models in the homophily and heterophily-specific design categories on the datasets under evaluation.

Hyperparameter Settings: For all four baseline models we tune two main parameters: the number of layers, and the number of hidden channels (dimension) of each layer. We also evaluate two subgraph sizes, or batch sizes, for the SAINT sampler (number of roots used for the random walk). For the GAT model only, we also tune the number of attention heads. We evaluate the same parameter ranges described in Lim et al. (2021) and defined by their codebase, and simply report the final parameters selected in Table 4. Selected parameters that are shared amongst all model and dataset pairs include training for 500 epochs, using the AdamW optimizer with a learning rate of 0.01, and creating 5 SAINT subgraphs per epoch. Model specific selected parameters include using 8 heads for the GAT, concatenative jumping knowledge for the GCNJK model, and the 2 hop setting for MixHop. For all settings that we choose as final, if possible, we verify that the accuracy is within $\pm 1\%$ of the value reported in Lim et al. (2021) as "best" for each model on each dataset.

There are two details of particular note concerning Table 4 (and corresponding results in Table 2 in the main work). First, for the `snap-patents` dataset we do not use the ClusterGCN sampler for the MixHop architecture as it proved too time and memory intensive for our hardware. We

[1]https://github.com/CUAI/Non-Homophily-Large-Scale

Table 4: Baseline Model Dataset-Specific Hyperparameter Settings

Model	Dataset	# Layers	# Hidden Channels	SAINT Batch Size
GAT	ogbn-arxiv	2	32	10,000
	ogbn-products *	–	–	–
	arxiv-year	2	32	10,000
	snap-patents	2	32	10,000
GCNJK	ogbn-arxiv	2	128	10,000
	ogbn-products	4	256	5,000
	arxiv-year	4	256	10,000
	snap-patents	2	256	10,000
MixHop	ogbn-arxiv	2	128	10,000
	ogbn-products	3	128	5,000
	arxiv-year	4	128	10,000
	snap-patents	2	128	10,000
LINKX	ogbn-arxiv	1	64	N/A
	ogbn-products	1	128	
	arxiv-year	1	256	
	snap-patents	1	16	

ground this choice in the fact that according to the performance reported in Lim et al. (2021), the performance of MixHop with cluster sampling would still have been below the performance of GOAT by approximately 8 accuracy points. As an additional comment, the computational costliness of this method was also a challenge in their work, precluding it from parts of their evaluation. In the spirit of scalability, overall, we see GraphSAINT as being a more relevant choice for our comparison due to its more favorable scaling characteristics.

Second, also in service of a competitive evaluation, we chose to report performance of the GAT model on ogbn-products * pulled from the Open Graph Benchmark's official leaderboard[2] for this dataset since the minibatch performance we achieved with the SAINT sampler was significantly lower than the reference result (as well as that of our GOAT model). The result from the leaderboard was trained using the neighbor sampling algorithm.

[2]https://ogb.stanford.edu/docs/leader_nodeprop/