Automated Loss function Search for Class-imbalanced Node Classification

Xinyu Guo [1] **Kai Wu** [1] **Xiaoyu Zhang** [2] **Jing Liu** [3]

Abstract

Class-imbalanced node classification tasks are prevalent in real-world scenarios. Due to the uneven distribution of nodes across different classes, learning high-quality node representations remains a challenging endeavor. The engineering of loss functions has shown promising potential in addressing this issue. It involves the meticulous design of loss functions, utilizing information about the quantities of nodes in different categories and the network's topology to learn unbiased node representations. However, the design of these loss functions heavily relies on human expert knowledge and exhibits limited adaptability to specific target tasks. In this paper, we introduce a high-performance, flexible, and generalizable automated loss function search framework to tackle this challenge. Across 15 combinations of graph neural networks and datasets, our framework achieves a significant improvement in performance compared to state-of-the-art methods. Additionally, we observe that homophily in graph-structured data significantly contributes to the transferability of the proposed framework.

1. Introduction

In recent years, the significance of learning qualitative node representations has grown in the context of accurately classifying node properties within real-world graphs (Wu et al., 2021; Xu et al., 2019b; Zhou et al., 2020). The adoption of graph neural networks (GNNs) (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018) for handling graph-structured data has garnered substantial success across various domains. Nevertheless, inherent class imbalances in natural graphs can introduce a bias toward major classes, resulting in reduced accuracy for minor classes when these imbalances are not addressed.

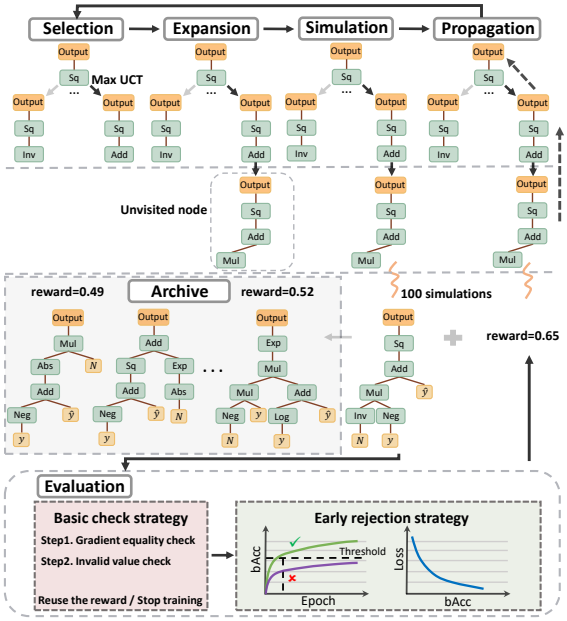


Figure 1: A schematic diagram of the AutoLINC framework with two main modules. The first module is the Monte Carlo tree search, which iteratively performs selection, expansion, simulation, and backpropagation steps to find the optimal loss function. The second module is the loss function check strategy. AutoLINC proactively filter out low-quality loss functions through the loss inspection strategy when evaluating the loss function.

To mitigate the challenges of class-imbalanced node classification, various methodologies have been explored (Ma et al., 2023). Notably, engineered loss functions have shown promise (Chen et al., 2021; Song et al., 2022), offering tailored solutions to combat class imbalance. Recent works like ReNode (Chen et al., 2021) and TAM (Song et al., 2022) have integrated graph topology information into their loss function designs. Another line of research explores the integration of contrastive learning into the context of class-imbalanced node classification (Zeng et al., 2023; Qian et al., 2022). However, these approaches often rely heavily on human expert knowledge and may exhibit limited adaptability to specific target tasks.

[1]School of Artificial Intelligence, Xidian University, Xi'an, China [2]School of Cyber Engineering, Xidian University, Xi'an, China [3]Guangzhou Institute of Technology, Xidian University, Guangzhou, China. Correspondence to: Kai Wu <kwu@xidian.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

To address these challenges, this paper presents an automated framework for searching loss functions, called AutoLINC (AutoLoss for Imbalanced Node Classification). AutoLINC defines a search space uniquely suited to the task and leverages the Monte Carlo Tree Search (MCTS) algorithm to discover effective loss functions. We introduce both a Basic Check Strategy and an Early Rejection Strategy to expedite the search process. The performance of our AutoLINC framework is validated through node classification experiments involving GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017) (abbreviated as SAGE) in three citation networks and two Amazon co-purchase networks. AutoLINC demonstrates substantial performance improvements when compared to state-of-the-art (SOTA) loss functions and non-loss function engineering methods.

The loss function search framework most closely aligned with AutoLINC's objectives is Autoloss-Zero (Li et al., 2022a), based on Regularized Evolution, which purports to address generic tasks. However, it faces challenges in direct application to class-imbalanced node classification problems. AutoLINC has tailored a search space, proxy tasks, and acceleration strategies specifically attuned to the complexities of imbalanced node classification problems. The comparative results against Autoloss-Zero underscore the strengths of our approach. Our contributions can be summarized as follows.

1. We introduce an automated loss function search framework with the objective of transcending the limitations associated with manually crafted loss functions. AutoLINC autonomously explores loss functions tailored to class-imbalanced node classification tasks, resulting in substantial performance improvements.

2. AutoLINC can be easily extended to other task scenarios by simply adjusting the search space and proxy tasks, making it a versatile framework. In addition to current Autoloss frameworks, we present a new high-performance loss function search framework for the Autoloss domain.

3. Within the AutoLINC framework, we have fashioned a search space specifically designed for addressing class-imbalanced node classification issues. Additionally, we have made refinements to the MCTS algorithm to expedite the quest for optimal loss functions.

4. We emphasize the noteworthy impact of homophily in graph-structured data on the adaptability of these loss functions to a wide range of graph datasets. Furthermore, AutoLINC demonstrates resilient generalization capabilities across varying class imbalance ratios.

2. Related Work

Class-imbalanced Node Classification. There is currently a substantial body of work dedicated to addressing the task of semi-supervised imbalanced node classification (Zhao et al., 2021a; Park et al., 2022; Liu et al., 2023; Ma et al., 2023). Recently, the field has seen a resurgence in the application of loss function engineering, which has yielded improved classification performance. ReNode (Chen et al., 2021), for instance, recalibrates the impact of labeled nodes based on their proximity to class boundaries. In contrast, TAM (Song et al., 2022) adapts to the local topology of individual nodes by dynamically adjusting margins for topologically improbable instances. However, constrained by the limitations of expert knowledge, the design of loss functions is often challenging to adequately account for the characteristics of both Graph and GNNs, thereby restricting their performance.

Loss Function Learning. Automated loss learning aims to alleviate the considerable human effort and expertise traditionally required for loss function design. While several studies (Xu et al., 2019a; Li et al., 2019; Wang et al., 2020; Liu & Lai, 2020; Li et al., 2021; Gao et al., 2022) have sought to learn loss functions automatically, they still heavily rely on human expertise in the loss search process, often initiating their search from existing loss functions. In related efforts, (Liu et al., 2021; Li et al., 2022a; Raymond et al., 2023) have employed evolutionary algorithms to search for loss functions composed of primitive mathematical operators for various computer vision tasks. In the realm of loss function learning for recommendation systems, Zhao *et al.* (Zhao et al., 2021b) have introduced a framework for discovering an appropriate loss function for specific data examples, utilizing a set of base loss functions and dynamically adjusting their weights for loss combination. In contrast, Li *et al.* (Li et al., 2022b) focus on the generation of entirely new loss functions rather than combining existing ones. However, it's worth noting that, in the context of graph data, there have been no prior attempts to apply loss function learning, necessitating the redesign of search spaces and algorithms to accommodate the unique characteristics of graph data.

3. Preliminary

Notation. Given an graph $G = \{V, E\}$, where $V = \{v_1, \cdots, v_C\}$ is a set of C nodes, and E is a set of edges. The node feature matrix is represented as $X \in \mathbb{R}^{C \times d}$, where d is the node's feature dimension. Node labels are denoted as $Y = \{y_1, \cdots, y_C | y_i \in \{1, \cdots, K\}\}$, where K is the number of classes.

Imbalanced Node Classification. The aim of the semi-supervised class-imbalanced node classification task is to train a classifier on an imbalanced node set X_{train}, typi-

cally employing a graph neural network denoted as f_θ, to generate unbiased predictions $\hat{y}$ for the remaining nodes. In this context, θ symbolizes the parameters of the GNN.

Definition 3.1. Context-free Grammar (CFG)(Hopcroft et al., 2006; Kusner et al., 2017) is a formal grammar characterized by a tuple comprised of 4 elements, namely, $\mathcal{G} = (U, \Sigma, R, O)$, where U denotes a finite set of non-terminal nodes, Σ a finite set of terminal nodes, R a finite set of production rules, each interpreted as a mapping from a single non-terminal symbol in U to one or multiple terminal/non-terminal node(s) in $(U \cup \Sigma)^*$ where $*$ represents the Kleene star operation, and O a single non-terminal node standing for a start symbol.

4. AutoLINC

4.1. Problem Definition

Given an imbalanced graph dataset, a GNN model, and a chosen metric, AutoLINC can autonomously search for a suitable loss function $\mathcal{L}$ to train the GNN, enabling the GNN to achieve competitive performance on the test set.

Definition 4.1. The problem of AutoLoss for imbalanced node classification can be framed as a nested optimization problem, given a measure σ:

$$\begin{array}{cc} \arg\max_{\mathcal{L}} & r\left(\mathcal{L}\right) = \sigma\left(f_{\theta^*(\mathcal{L})}\left(X_{val}\right), Y_{val}\right) \\ \text{s.t.} & \theta^*\left(\mathcal{L}\right) = \arg\min_\theta \mathcal{L}\left(f_\theta\left(X_{train}\right), Y_{train}\right) \end{array} \tag{1}$$

where $r(\mathcal{L})$ is the reward of loss function $\mathcal{L}$, X_{val} is the validation dataset, Y_{val} is the label of the validation dataset, and Y_{train} is the label of the training dataset.

4.2. Search Space

Input Nodes: We take the model's output logits $\hat{y}$ and their corresponding labels y as input nodes, supplemented with some constants to enhance the flexibility of the search space. Moreover, for class-imbalanced problems, we also introduce the category node count N as an input node.

Table 1: Primitive Operators.

ELEMENT-WISE	EXPRESSION	ARITY
ADD	$x + y$	2
MUL	$x \times y$	2
NEG	$-x$	1
ABS	$\lvert x \rvert$	1
INV	$1/(x + \epsilon)$	1
LOG	$sign(x) \bullet \log(\lvert x \rvert + \epsilon)$	1
EXP	e^x	1
TANH	$\tanh x$	1
SQUARE	x^2	1
SQRT	$sign(x) \bullet \sqrt{\lvert x \rvert + \epsilon}$	1
AGGREGATIONR	EXPRESSION	ARITY
MEAN	$\frac{1}{C}\sum_{i=1}^{C}(x_i)$	1

Primitive Operators: The primitive operators consist of element-wise operators and aggregation operator, as shown in Appendix Table 1. We use only one aggregation operator, Mean, as the final aggregation.

Solution Expression: Loss functions are fundamentally mathematical expressions that can be represented in the form of parse trees using a CFG. In our work, using the elements we have, U denotes symbols like (D). O corresponds to the output of the loss function (Output, o). Σ is $\{y, \hat{y}, N\}$. R is our primitive operators, mapping from one non-terminal node (e.g., o/D) to another node. The parse tree of each loss function starts from the root node, Output, and traverses pre-order based on the product rule until all leaf nodes are represented by symbols from the terminal node set. As shown in Fig. 1, the loss function is presented in the form of a parse tree (for clarity, non-terminal node D is hidden, while the product rules are retained). When an expression is evaluated, the model's output $\hat{y}$ has the same shape as the one-hot encoded labels y and the number of categories N. The tree's output is eventually averaged to obtain a single value after the leaf nodes input into the loss function expression tree are calculated.

4.3. Search Algorithm

Unlike the current Autoloss search framework based on Regularized evolution (Real et al., 2020; 2019), we design a more capable search algorithm. AutoLINC primarily consists of two parts: 1) MCTS (Coulom, 2006; Sun et al., 2023), which explores the most promising expressions through repeated selection, expansion, simulation, and back-propagation steps; 2) Task evaluation with a loss check strategy, which filters out evaluated, training-unfriendly, poorly converging, and poor-performing loss functions, speeding up the evaluation of proxy tasks. AutoLINC's framework is illustrated in Fig. 1, and the detailed sections can be found in Algorithm 1. MCTS repeats the below steps until the termination conditions are met.

Step 1: Selection - MCTS starts from the root node and uses the Upper Confidence Bounds applied for Trees (UCT) selection strategy to choose the next node, repeatedly selecting until reaching a leaf node or an expandable node.

$$UCT(s,a) = Q(s,a) + c\sqrt{\ln[W(s)]/W(s,a)} \tag{2}$$

Here, $Q(s,a)$ represents the average reward of taking action $a \in \mathcal{A}$ in state s. $Q(s,a)$ encourages exploitation of the current best child node. In this paper, our objective is to determine the optimal expression for the loss function. To achieve this, we define the maximum simulated result value as $Q(s,a)$ in state s when taking action a. $W(s)$ is the number of visits to node s, and $W(s,a)$ is the number of times action a has been taken in state s. Therefore, $\sqrt{\ln[W(s)]/W(s,a)}$ encourages exploration of other

Table 2: Compared results of AutoLINC with the SOTA loss functions. Values in bold indicate the best performance, while values underlined indicate the second-best performance. $* \pm *$ denotes the mean and standard errors.

GNN	Dataset / Imbalance Ratio:10	Cora		CiteSeer		PubMed		Amazon-computers		Amazon-photo	
		bAcc	F1	bAcc	F1	bAcc	F1	bAcc	F1	bAcc	F1
GCN	CE	53.89±0.77	49.13±1.20	35.93±0.45	23.58±0.74	61.96±0.51	47.55±1.01	80.99±0.69	72.67±0.48	78.77±1.08	73.17±1.53
	Re-Weight	60.91±1.05	59.18±1.31	40.79±1.56	31.95±1.95	67.18±1.20	59.74±2.19	82.82±0.36	73.44±0.43	81.87±0.73	75.04±1.09
	PC Softmax	68.15±0.82	67.90±0.91	49.70±1.28	45.24±1.33	70.44±0.62	69.21±0.76	**85.10±0.06**	**75.20±0.17**	78.20±0.66	75.41±0.66
	BalancedSoftmax	68.96±0.52	68.67±0.49	53.57±1.44	50.55±1.76	71.00±1.12	69.72±1.18	83.70±0.31	74.48±0.48	81.93±0.95	76.19±1.55
	BalancedSoftmax+TAM	69.17±0.77	69.00±0.73	55.65±1.19	54.06±1.39	72.15±1.08	71.79±1.35	83.47±0.32	74.40±0.50	82.50±0.79	76.25±1.41
	ReNode	64.64±1.09	64.00±1.43	42.02±1.39	33.36±1.82	68.58±1.19	62.97±2.51	82.75±0.22	72.96±0.37	81.68±0.75	74.82±1.01
	AutoLINC	**70.21±0.67**	**69.67±0.79**	**56.56±1.97**	**56.09±2.06**	**72.43±1.03**	**72.42±1.07**	83.23±0.23	71.83±0.45	**83.06±0.80**	**76.66±1.18**
GAT	CE	49.26±0.97	44.80±1.36	34.34±0.42	21.86±0.35	60.31±0.23	46.41±0.52	66.68±0.88	61.28±0.88	57.36±1.15	51.26±1.03
	Re-Weight	67.27±0.67	66.35±0.72	44.73±1.64	39.73±2.19	67.89±1.18	60.83±2.15	73.66±0.65	66.61±0.98	64.06±1.16	58.94±1.25
	PC Softmax	64.67±1.02	63.60±1.22	50.50±1.59	45.86±1.94	72.49±0.62	70.97±0.77	72.18±1.16	64.80±1.15	66.49±1.39	62.57±1.75
	BalancedSoftmax	65.92±0.70	65.43±0.82	53.70±1.56	50.31±1.93	71.24±0.84	69.66±0.91	74.88±0.53	68.84±0.86	64.81±1.05	60.09±1.35
	BalancedSoftmax+TAM	66.30±1.01	65.28±1.03	54.14±1.31	51.84±1.63	72.24±0.85	71.96±0.76	75.66±0.62	69.80±1.09	65.88±0.98	61.47±0.99
	ReNode	64.19±1.20	64.01±1.42	41.15±1.26	35.45±1.49	68.19±1.46	62.78±2.71	75.17±0.79	67.47±1.04	63.54±1.43	59.27±1.58
	AutoLINC	**70.63±1.01**	**70.16±1.04**	**58.48±1.57**	**57.38±1.89**	**72.57±0.88**	**72.27±1.02**	**78.86±0.59**	**71.73±0.73**	**74.41±0.69**	**69.11±0.91**
SAGE	CE	51.57±0.54	45.11±0.83	35.34±0.25	22.82±0.38	61.01±0.55	48.03±0.88	68.51±1.52	62.78±1.78	59.74±1.67	51.53±1.96
	Re-Weight	55.17±0.88	52.07±1.33	38.53±1.35	30.60±1.91	62.00±0.87	52.05±2.03	72.41±1.05	63.77±0.87	60.28±1.47	52.03±1.74
	PC Softmax	65.49±0.65	64.73±0.82	49.54±1.45	45.04±1.82	**71.38±0.77**	**70.52±1.04**	75.17±0.54	68.78±0.48	64.04±0.74	58.62±1.09
	BalancedSoftmax	65.38±0.66	65.06±0.76	51.56±1.68	48.60±1.91	69.77±0.70	68.63±0.83	75.76±1.02	68.03±0.74	67.62±1.86	60.96±2.09
	BalancedSoftmax+TAM	66.54±0.49	66.24±0.65	52.46±1.23	49.27±1.72	70.26±0.87	69.94±2.83	78.11±0.77	70.53±0.65	69.45±2.01	62.64±2.17
	ReNode	59.36±0.57	57.70±0.48	41.12±1.56	32.61±1.80	64.67±1.21	55.82±2.72	76.25±0.71	69.03±0.56	64.72±1.25	57.16±1.35
	AutoLINC	**68.51±1.02**	**68.49±1.02**	**56.78±1.27**	**56.16±1.37**	70.97±1.04	70.32±0.96	**80.02±0.68**	**71.19±0.32**	**72.98±1.53**	**66.66±1.99**

nodes. c is the exploration rate, generally defined empirically for a specific problem. In the selection, expansion, or simulation process, we limit the maximum number for selecting product rules from the root node to 10 to avoid lengthy loss function expressions.

Step 2: Expansion - Upon reaching an expandable node, a randomly unvisited child node is chosen for expansion. After expansion, an evaluation is conducted if a terminal state is reached. The evaluation results are backtracked to update the parent node until the root node.

Step 3: Simulation - If the currently expanded node is still non-terminal, a random selection is made for simulation, continuously choosing child nodes until a terminal state is reached. In calculating rewards for loss functions reaching a terminal state, we use the bAcc calculated on the validation set as the reward.

Step 4: Backpropagation- Update the visit count and Q value of the parent node until the root node based on the simulation results.

4.4. Loss Function Check

Legality Check of the Loss Function: A legitimate loss function should encompass the GNN's output logits $\hat{y}$ and the corresponding labels y. To ensure the loss function's competence in addressing class-imbalanced node classification issues, it is imperative that the function accounts for the class-specific node counts, represented as N. Any loss functions lacking GNN output logits $\hat{y}$, labels y, or class-specific node counts N will not undergo evaluation.

Basic Check Strategy: This strategy encompasses fundamental checks, such as detecting invalid Nan values and ensuring gradient equality. Because loss functions are represented in the form of expression trees, they may exhibit symmetrical cases, where different-looking loss function trees are equivalent. Additionally, distinct loss functions can sometimes yield equivalent gradients. We maintain records of the evaluated loss function formulas and their respective rewards. In the case of equivalent formulas, their rewards are directly reused.

Early Rejection Strategy: While legality checks ensure that the evaluated solutions incorporate $y, \hat{y}, N$, and equality checks prevent the redundant evaluation of equivalent formulas, there remain a significant number of underperforming, training-unfriendly loss functions. To address this, we employ the Early Rejection Strategy to discard poorly converging and ineffective loss functions. The Early Rejection Strategy comprises Monotonicity Checks and Poor-performance Rejections.

1) *Monotonicity Check*: As per (Liu et al., 2021), a crucial criterion during the proxy task's training is assessing the loss function's quality based on its impact on accuracy metrics. An ideal loss function should exhibit a monotonous increase in accuracy metrics on the training set as the loss value decreases. If the loss function's reduction is not positively correlated with improved accuracy metrics on the training set, it is deemed invalid. In such cases, the proxy task is prematurely terminated to mitigate computational costs.

2) *Poor-performance Rejection*: Throughout the proxy task's training phase, the loss function is considered suboptimal if the metrics on the validation set significantly lag behind the performance achieved by the Top loss functions searched thus far. In such scenarios, the proxy task is concluded early to reduce computational costs.

Table 3: The comparison results of the loss functions discovered by AutoLINC in Table 11 on datasets with $\rho = 5$.

GNN	Dataset / Imbalance Ratio:5	Cora		CiteSeer		PubMed		Amazon-computers		Amazon-photo	
		bAcc	F1	bAcc	F1	bAcc	F1	bAcc	F1	bAcc	F1
GCN	CE	64.19±0.80	64.08±1.08	41.23±0.98	32.82±1.41	65.32±0.84	55.58±1.77	83.80±0.40	76.22±0.37	86.50±0.76	82.72±1.15
	Re-Weight	69.74±0.98	70.09±1.00	49.49±1.62	45.45±2.00	71.52±0.91	67.24±1.35	84.34±0.30	77.52±0.34	87.02±0.69	82.64±1.06
	PC Softmax	73.36±0.55	73.35±0.57	59.18±1.29	57.85±1.45	73.77±0.64	72.55±0.68	**85.49±0.11**	**78.73±0.23**	**87.77±0.10**	**83.99±0.10**
	BalancedSoftmax	73.98±0.81	73.61±0.77	60.66±1.08	59.72±1.13	**74.67±0.87**	73.35±1.03	84.99±0.39	77.48±0.29	87.29±0.66	82.92±0.91
	BalancedSoftmax+TAM	74.02±0.70	74.08±0.67	**61.14±1.29**	**60.41±1.31**	74.55±0.82	73.92±0.99	84.88±0.38	77.21±0.22	87.52±0.61	83.21±0.88
	ReNode	72.50±0.81	73.11±0.76	53.48±1.55	50.81±1.75	71.86±0.82	67.96±1.17	84.89±0.18	76.98±0.22	87.37±0.71	82.74±1.11
	AutoLINC	**75.26±0.54**	**74.79±0.47**	60.64±1.05	60.39±1.01	74.42±0.86	**74.61±0.88**	84.63±0.37	73.80±0.58	87.03±0.83	81.57±1.28
GAT	CE	57.12±1.15	56.32±1.38	37.76±1.29	28.27±1.95	64.11±0.84	55.76±1.66	73.30±0.81	67.85±1.04	72.05±1.62	67.48±1.60
	Re-Weight	72.99±0.88	72.05±1.08	53.72±1.24	51.61±1.52	72.65±0.77	68.80±1.14	76.98±0.58	69.89±0.78	77.68±0.77	73.72±1.21
	PC Softmax	72.11±0.57	71.85±0.69	57.86±1.46	55.91±1.71	73.61±0.70	72.37±0.88	75.11±0.75	68.31±1.11	79.85±0.90	**75.93±0.80**
	BalancedSoftmax	73.04±0.37	72.44±0.43	58.97±1.27	58.20±1.33	74.25±0.63	72.82±0.77	78.21±0.66	71.52±0.74	75.72±0.88	71.54±1.21
	BalancedSoftmax+TAM	73.37±0.67	72.47±0.78	60.26±0.84	59.48±0.87	**74.75±0.66**	74.09±0.83	79.04±0.89	72.56±1.11	77.52±1.11	73.48±1.48
	ReNode	74.12±0.75	74.44±0.77	54.43±1.95	52.82±2.41	72.35±1.05	68.89±1.76	78.84±0.67	71.56±1.01	76.08±1.69	70.89±1.91
	AutoLINC	**75.57±0.42**	**75.24±0.40**	**63.58±0.76**	**63.29±0.71**	74.51±0.82	**74.50±0.79**	**82.60±0.68**	**74.39±1.00**	**79.98±1.02**	74.39±1.75
SAGE	CE	58.76±0.85	56.39±1.22	39.18±0.97	30.11±1.56	63.71±0.89	53.82±1.70	80.13±0.55	72.55±0.82	78.11±1.01	72.51±1.50
	Re-Weight	65.84±1.13	65.23±1.35	45.60±1.46	40.75±1.95	67.81±0.79	62.56±1.50	81.03±0.54	74.40±0.45	79.09±1.14	72.93±1.71
	PC Softmax	71.20±0.96	71.07±1.00	56.82±1.63	55.18±1.82	**73.69±0.32**	**72.87±0.41**	82.42±0.36	72.27±0.45	81.77±0.29	77.74±0.27
	BalancedSoftmax	71.05±0.75	71.00±0.91	57.88±1.25	57.00±1.26	73.19±0.30	72.05±0.43	81.35±0.49	**74.76±0.48**	**82.94±1.00**	**77.93±1.52**
	BalancedSoftmax+TAM	71.71±0.58	71.55±0.66	59.55±1.29	58.68±1.27	73.36±0.55	72.46±0.66	82.46±0.32	74.56±0.34	82.74±1.17	77.18±1.68
	ReNode	69.30±1.40	69.21±1.44	53.14±1.53	51.05±1.87	68.39±1.09	62.26±2.03	82.41±0.32	73.82±0.71	82.79±0.69	76.90±1.17
	AutoLINC	**74.15±0.69**	**73.80±0.70**	**61.41±0.85**	**61.13±0.78**	72.59±0.71	72.08±0.74	**82.79±0.36**	72.96±0.64	82.31±0.95	77.67±1.36

4.5. Framework of AutoLINC

To manage non-terminal nodes within the parsing tree, we employ a last-in-first-out strategy. The last non-terminal node placed on the stack, denoted as NT, represents the current node. We define the action space $\mathcal{A}$ as R and the state space $\mathcal{S}$ as all possible traversals of complete or incomplete parse trees in ordered sequences. In the current state $s_t = [a_1, a_2, \cdots, a_t]$, the MCTS agent filters out invalid production rules for the present non-terminal node. Subsequently, it selects a valid rule as action a_{t+1}. This leads to the expansion of the parse tree with a new terminal or non-terminal branch, determined by a_{t+1}. Concurrently, the agent progresses to a new state $s_{t+1} = [a_1, a_2, \cdots, a_t, a_{t+1}]$. The agent proceeds by removing the current non-terminal symbol from NT and adding any non-terminal nodes, if they exist, on the right-hand side of the selected rule to the stack.

Additionally, legality check, basic check strategies, and early rejection are executed, followed by reward calculation based on the proxy task (Eq. 1). The discovered $\mathcal{L}$ and its associated reward are recorded in M. When the agent encounters an unvisited node, a series of simulations commence, with the agent randomly selecting the next node until the parse tree is completed. In a similar manner, the reward is computed after performing the loss function check strategy. The highest result from these attempts serves as the reward for the current simulation phase, backpropagating from the current unvisited node to the root node. We then select the top 10 loss functions, denoted as M_{top}, from M and extract the best loss function, $\mathcal{L}^\star$, from M_{top}. Note that the reward is calculated based on the complete task. Appendix Algorithm 1 describes the detailed procedure of AutoLINC.

4.6. Proxy Task

To accelerate the search process, rewarding loss function expressions typically necessitate training graph neural networks, a time-intensive endeavor. In evaluating these loss functions, we employ a lightweight proxy training task. Given the inherent challenges in partitioning graph data structures, we curtail the number of training iterations on real datasets for this proxy task. Subsequently, rewards are computed based on the validation set's balanced accuracy.

5. Experiments

We illustrate the effectiveness and adaptability of AutoLINC by addressing the following key questions:

Q1: Can the loss functions discovered by AutoLINC better adapt to class-imbalanced node classification tasks than SOTA alternatives? (see Section 5.2)

Q2: Are the loss functions found by AutoLINC effective across datasets with varying degrees of class imbalance? (see Section 5.2)

Q3: Can the loss functions derived from a single GNN and graph dataset demonstrate effective transferability to other GNN models and datasets? (see Section 5.3)

Q4: Is AutoLINC superior to SOTA non-loss function engineering methods and loss function search method? (see Section 5.4 and 5.5)

5.1. Experimental Setup

Datasets We validate AutoLINC on well-known citation networks (Yang et al., 2016), comprising three datasets: Cora, CiteSeer, and PubMed, as well as Amazon's co-purchase networks (McAuley et al., 2015), which consist of two

Table 4: The performance of loss functions A, E, I, J, and K on various combinations of datasets and graph neural networks.

GNN	Dataset / Imbalance Ratio: 10	Cora		CiteSeer		PubMed		Amazon-computers		Amazon-photo	
		bAcc	F1	bAcc	F1	bAcc	F1	bAcc	F1	bAcc	F1
GCN	BalancedSoftmax+TAM	69.17±0.77	69.00±0.73	55.65±1.19	54.06±1.39	72.15±1.08	71.79±1.35	**83.47±0.32**	**74.40±0.50**	82.50±0.79	76.25±1.41
	Cora-GCN(A)	70.21±0.67	69.67±0.79	56.64±2.04	55.82±2.03	71.95±1.12	71.64±1.19	26.82±3.29	17.67±3.12	33.40±3.84	24.00±4.20
	CiteSeer-GAT(E)	69.86±0.83	68.77±0.74	56.15±2.00	55.56±2.02	71.40±1.49	71.17±1.48	35.42±5.00	21.68±4.72	35.79±4.12	23.91±4.56
	PubMed-SAGE(I)	**71.13±0.82**	**70.47±0.88**	**58.32±1.56**	**57.64±1.61**	**72.66±1.01**	**72.50±1.18**	16.74±1.20	6.48±1.21	27.24±2.20	19.37±2.94
	computers-GCN(J)	68.41±0.70	67.28±0.66	53.05±1.36	51.06±1.48	68.03±1.02	61.65±1.89	83.23±0.23	71.83±0.45	80.64±0.69	73.80±1.01
	photo-GCN(K)	64.50±0.34	63.46±0.34	52.93±0.76	51.82±0.86	67.18±0.80	60.28±1.57	82.36±0.21	70.55±0.38	**83.06±0.80**	**76.66±1.18**
GAT	BalancedSoftmax+TAM	66.30±1.01	65.28±1.03	54.14±1.31	51.84±1.63	72.24±0.85	71.96±0.76	75.66±0.62	**69.80±1.09**	65.88±0.98	61.47±0.99
	Cora-GCN(A)	69.83±0.96	70.05±0.95	58.30±1.39	57.17±1.74	72.80±1.06	71.92±1.29	50.85±7.11	43.45±6.16	59.37±6.79	54.25±6.74
	CiteSeer-GAT(E)	70.79±0.78	70.63±0.83	58.48±1.57	57.38±1.89	72.50±1.04	71.55±1.03	40.61±7.01	35.02±6.54	55.09±7.59	49.64±7.84
	PubMed-SAGE(I)	**71.51±0.69**	**71.32±0.68**	**59.11±1.28**	**58.02±1.43**	**73.62±0.83**	**72.98±1.10**	20.47±2.01	14.45±2.06	20.62±2.16	16.29±2.82
	computers-GCN(J)	57.82±1.24	56.47±1.24	44.01±1.27	41.47±1.53	64.25±1.04	56.20±1.83	75.15±0.73	67.50±0.79	59.90±1.66	55.01±1.82
	photo-GCN(K)	60.46±0.89	59.04±0.85	45.45±1.35	43.07±1.59	66.78±0.98	61.24±1.46	**77.95±0.86**	69.21±1.04	**71.17±1.59**	**67.49±1.42**
SAGE	BalancedSoftmax+TAM	66.54±0.49	66.24±0.65	52.46±1.23	49.27±1.72	70.26±0.87	69.94±2.83	78.11±0.77	70.53±0.65	69.45±2.01	62.64±2.17
	Cora-GCN(A)	**68.46±0.94**	**68.20±0.89**	56.68±1.24	55.92±1.44	70.12±0.98	69.34±1.23	14.83±0.73	5.68±0.73	21.37±1.57	11.95±1.37
	CiteSeer-GAT(E)	68.21±0.98	68.03±1.01	**56.88±1.29**	**56.04±1.33**	70.38±0.94	69.00±1.16	14.71±1.10	5.33±1.10	20.41±1.04	10.41±1.03
	PubMed-SAGE(I)	67.66±0.99	67.20±1.04	56.03±1.42	55.21±1.58	**70.97±1.04**	**70.32±0.96**	10.93±0.32	3.80±0.36	18.83±1.06	8.69±0.62
	computers-GCN(J)	67.80±0.62	66.35±0.61	53.01±1.39	51.05±1.48	67.66±0.85	61.92±1.32	**79.89±0.67**	**71.11±0.41**	**69.77±1.85**	**63.20±2.26**
	photo-GCN(K)	67.47±0.65	66.52±0.61	54.55±1.38	53.13±1.39	66.44±0.67	59.34±1.12	55.88±3.68	47.77±3.88	68.76±1.27	61.79±1.70

Table 5: The comparative results of loss functions A, E, I, J, and K on datasets with a class imbalance ratio of 5.

GNN	Dataset / Imbalance Ratio: 5	Cora		CiteSeer		PubMed		Amazon-computers		Amazon-photo	
		bAcc	F1	bAcc	F1	bAcc	F1	bAcc	F1	bAcc	F1
GCN	BalancedSoftmax+TAM	74.02±0.70	74.08±0.67	61.14±1.29	60.41±1.31	**74.55±0.82**	**73.92±0.99**	**84.88±0.38**	**77.21±0.22**	87.52±0.61	**83.21±0.88**
	Cora-GCN(A)	75.26±0.54	74.79±0.47	62.79±0.97	62.66±0.93	73.61±1.23	73.74±1.36	31.40±4.33	18.99±3.84	45.40±6.47	35.51±7.14
	CiteSeer-GAT(E)	74.52±0.52	74.27±0.40	**63.10±0.82**	**62.93±0.83**	73.36±1.14	73.37±1.26	36.50±4.17	23.23±4.18	35.87±4.20	24.98±3.98
	PubMed-SAGE(I)	**75.41±0.38**	**74.84±0.48**	63.04±0.78	62.85±0.75	73.31±0.94	73.26±0.92	14.29±0.68	6.20±0.66	17.73±0.80	10.25±1.28
	computers-GCN(J)	74.02±0.35	72.88±0.36	59.68±0.88	58.92±0.93	72.08±0.68	68.72±1.05	84.63±0.37	73.80±0.58	**87.61±0.75**	82.77±1.20
	photo-GCN(K)	71.47±0.54	71.21±0.55	59.31±0.96	58.23±1.05	70.18±0.79	65.18±1.30	84.22±0.29	73.63±0.55	87.03±0.83	81.57±1.28
GAT	BalancedSoftmax+TAM	73.37±0.67	72.47±0.78	60.26±0.84	59.48±0.87	74.75±0.66	74.09±0.83	79.04±0.89	**72.56±1.11**	77.52±1.11	73.48±1.48
	Cora-GCN(A)	**76.08±0.58**	**75.37±0.55**	**64.23±0.95**	**63.84±0.93**	73.72±0.88	73.48±0.82	44.93±7.73	38.84±7.07	77.64±6.34	73.06±6.70
	CiteSeer-GAT(E)	75.34±0.33	74.56±0.48	63.58±0.76	63.29±0.71	73.83±0.90	73.61±0.93	48.14±7.60	41.72±6.72	71.81±7.11	66.25±7.51
	PubMed-SAGE(I)	75.87±0.42	75.34±0.52	63.85±0.88	63.49±0.88	**74.82±0.63**	**74.63±0.83**	23.65±1.36	17.06±1.28	24.37±2.25	20.34±2.88
	computers-GCN(J)	61.75±1.08	60.77±1.26	49.17±0.95	47.93±0.91	69.57±0.88	66.43±1.47	79.78±0.62	72.00±0.88	77.79±0.52	72.63±0.90
	photo-GCN(K)	65.27±0.72	64.72±0.82	49.68±1.04	48.31±1.20	71.09±0.62	67.56±0.94	**80.71±0.73**	72.12±1.18	**81.00±0.82**	**76.61±1.10**
SAGE	BalancedSoftmax+TAM	71.71±0.58	71.55±0.66	59.55±1.29	58.68±1.27	**73.36±0.55**	**72.46±0.66**	82.46±0.32	**74.56±0.34**	82.74±1.17	77.18±1.68
	Cora-GCN(A)	72.92±0.58	72.98±0.56	62.16±0.88	**61.89±0.86**	72.06±0.97	71.62±1.11	15.30±1.07	5.62±0.81	19.86±0.80	10.78±0.85
	CiteSeer-GAT(E)	**74.01±0.67**	**73.70±0.73**	**62.16±0.72**	61.73±0.72	72.25±0.70	71.83±0.67	13.33±0.65	4.54±0.54	18.80±0.46	9.72±0.83
	PubMed-SAGE(I)	73.93±0.62	73.56±0.67	61.77±0.74	61.25±0.85	72.59±0.71	72.08±0.74	11.69±0.43	4.45±0.55	18.20±0.98	8.02±0.88
	computers-GCN(J)	73.18±0.51	71.61±0.61	58.03±1.08	57.31±1.10	71.40±0.57	68.70±0.71	**82.72±0.36**	73.15±0.68	**84.43±0.51**	**79.28±1.18**
	photo-GCN(K)	72.41±0.57	71.48±0.60	59.74±1.04	58.70±1.03	68.32±0.59	62.93±0.92	73.82±2.41	63.91±2.22	82.68±0.75	78.31±1.23

datasets: Computers and Photo.

Baseline To evaluate the effectiveness of the loss functions learned by AutoLINC, we compare them with several baseline methods, including cross-entropy, re-weight (Japkowicz & Stephen, 2002), balanced softmax (Ren et al., 2020), PC softmax (Hong et al., 2021), ReNode (Chen et al., 2021), and TAM (Song et al., 2022). All experiments are conducted ten times to calculate bAcc and F1 scores.

GNN Settings Prominent GNNs: GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and SAGE (Hamilton et al., 2017), consist of a 2-layer neural network with a hidden layer dimension of 256. More details on datasets, baselines and parameters are shown in Appendix A.

5.2. Comparison with SOTA Loss Functions

The outcomes of AutoLINC and SOTA loss functions are detailed in Tables 2 and 3. The discovered loss functions A-O and their convergence trends are presented in Appendix Table 11 and Appendix Fig. 4, respectively. Our observations are as follows: 1) AutoLINC consistently demonstrates significant performance enhancements, with the most substantial gains seen with GAT and SAGE, while GCN experiences a more modest improvement. 2) AutoLINC leads to remarkable performance improvements on datasets such as Cora, CiteSeer, Amazon-computers, and Amazon-photo. However, the enhancement on PubMed is moderate, possibly due to its high degree of topology imbalance.

5.3. Transferability and Convergence

Transferability to Different Imbalance Ratios. We additionally examined the transferability of the discovered loss functions from datasets with an initial imbalance ratio of $\rho = 10$ to datasets with $\rho = 5$, as outlined in Table 3. Impressively, these loss functions retained their high level of performance, underlining their robustness in facilitating

node classification across varying class imbalance ratios.

Transferability across different datasets and GNNs. We evaluated the transferability of five distinct loss functions, indicated as A, E, I, J, and K (as listed in Appendix Table 11), in various combinations of data sets and graph neural networks, as presented in Table 4. Our observations are: 1) Loss functions A, E, and I demonstrate robust transferability within datasets of the same type (citation networks) and even surpass other loss functions in certain scenarios. 2) Likewise, J and K exhibit excellent performance when applied to datasets of the same type (Amazon co-purchase networks). 3) However, when these loss functions are transferred to datasets of different types, their transfer performance diminishes. Notably, A, E, and I, trained on citation network datasets, exhibit limited classification performance on Amazon datasets.

Consequently, we can conclude that homophily in graph-structured data significantly enhances AutoLINC's transferability. Additionally, as detailed in Table 5, as the class imbalance ratio is reduced to 5, models trained using the loss functions from Table 11 display enhanced classification capabilities due to the decreased class imbalance. Similarly, they continue to exhibit superior transfer performance within datasets of the same type, but their transfer performance diminishes. In some instances, they exhibit no transferability to datasets of different types.

Convergence. In Appendix Fig. 4, we present the average bAcc curves for the discovered loss functions detailed in Appendix Table 11. Although the convergence is relatively slow, the final model outperforms other loss functions in most cases. Additionally, Table 2 illustrates the highly competitive performance of the best-trained model.

Discussion. We further investigate the potential performance enhancements achieved by AutoLINC across various GNNs. Unlike Eq. 1, we employ the lowest score among the three scores obtained from GCN, GAT, and SAGE models for a particular dataset to evaluate the loss function. In Table 6, we observe: 1) AutoLINC demonstrates superior performance compared to loss functions discovered within the scope of a single GNN model; 2) This strategy consistently delivers enhanced classification performance when compared to state-of-the-art loss functions; 3) However, the loss functions generated by this approach exhibit reduced transferability across different GNNs when compared to loss functions A, E, and I.

Our method requires additional training and evaluation by training the model on the training set using the learned loss function and evaluating the model on the validation set to obtain scores for the loss function. Methods based on loss functions train the model on the training set, which is similar to our approach in terms of the use of the training set. Additionally, the training set under the adopted class imbalance setting is also small. Our method does not heavily rely on the training set. The impact on the validation set is that our proxy task requires evaluating the score of the loss function on the validation set to guide AutoLINC in searching for the loss function. If the validation set is too small, it may lead to the search framework discovering loss functions overfitting the validation set. To investigate, we conducted experiments to evaluate on smaller validation sets. We used three different combinations of datasets and networks, namely Cora-GCN, CiteSeer-GAT, and PubMed-GraphSAGE. The training and testing sets remained unchanged. The size of the validation set was set to maintain 5, 10, and 20 nodes per class. The compared methods were Balanced Softmax and TAM, and we did not perform any tuning for any method. Other settings were the same as in Table 2. The experimental results are presented in Table 8.

If reducing the size of the validation set leads to a decrease in the performance of node classification, it may be due to overfitting of the loss function found by our method on the validation set, or it may be because the selected best model is not optimal, or due to the deterioration of other model parameters. This is difficult to analyze. Therefore, the conclusions drawn from experiments with different validation set sizes may not be convincing. However, by comparing with Balanced Softmax and TAM under the same validation set size settings, our method still demonstrates good performance on smaller validation sets. This suggests that our method does not heavily rely on the validation set.

5.4. Comparison with SOTA Non-loss Function Engineering Methods

To enhance the generality of the baseline, we have included comparative experiments with SOTA non-loss function engineering, GraphSMOTE (Zhao et al., 2021a), GraphENS (Park et al., 2022), and GraphSHA (Li et al., 2023) see Appendix A for more information). The experimental setting is the same as in Section 5.2. The results of AutoLINC derived from Table 2 and 3 are competitive to GraphSHA. We also combine AutoLINC with GraphSHA to discover loss functions so as to explore the scalability of AutoLINC. In Table 7, in comparison to GraphSMOTE, GraphENS and GraphSHA, AutoLINC combined with GraphSHA demonstrates advanced performancere across different datasets, GNN backbones, and class imbalance ratios. This illustrates the scalability of AutoLINC and the feasibility of overcoming the challenges of class imbalance in node classification from a loss function standpoint.

5.5. AutoLINC v.s. Autoloss-Zero

We demonstrate the advantages of AutoLINC over Autoloss-Zero (Li et al., 2022a). AutolossZero-A adopts the original

Table 6: The performance of AutoLINC trained on three GNNs. Here, $\rho = 10$. AutoLINC-Cora: $(\tanh(-((1/N \times (-y) + \hat{y})))^2$; AutoLINC-CiteSeer: $(\hat{y} + ((1/N \times (-y)) + \hat{y}))^2$; AutoLINC-PubMed: $N \times \hat{y}^4 + (-(\hat{y} \times y))$.

GNN	Dataset / Imbalance Ratio: 10	Cora bACC	Cora F1	CiteSeer bACC	CiteSeer F1	PubMed bACC	PubMed F1
GCN	BalancedSoftmax+TAM	69.17±0.77	69.00±0.73	55.65±1.19	54.06±1.39	72.15±1.08	71.79±1.35
	AutoLINC-Cora	**69.63±0.77**	68.95±0.79	57.52±1.76	**57.14±1.82**	71.82±1.20	71.27±1.41
	AutoLINC-CiteSeer	69.55±0.60	69.06±0.61	**57.78±1.63**	57.10±1.71	72.09±1.54	71.29±1.60
	AutoLINC-PubMed	69.07±0.61	**69.23±0.51**	55.48±1.81	54.91±1.81	**73.37±1.03**	**72.11±1.43**
GAT	BalancedSoftmax+TAM	66.30±1.01	65.28±1.03	54.14±1.31	51.84±1.63	72.24±0.85	**71.96±0.76**
	AutoLINC-Cora	70.55±0.96	69.97±1.02	58.31±1.61	57.40±1.64	72.23±1.09	70.98±1.26
	AutoLINC-CiteSeer	70.13±0.82	69.89±0.98	**59.15±1.43**	**58.11±1.65**	72.22±1.19	71.72±1.41
	AutoLINC-PubMed	**71.41±0.62**	**71.66±0.67**	58.51±1.45	57.21±1.75	**73.00±0.96**	71.80±1.12
SAGE	BalancedSoftmax+TAM	66.54±0.49	66.24±0.65	52.46±1.23	49.27±1.72	69.64±0.93	70.26±0.87
	AutoLINC-Cora	**68.73±0.92**	**68.72±0.89**	56.34±1.40	55.10±1.65	69.68±0.90	68.76±1.15
	AutoLINC-CiteSeer	68.47±0.99	68.43±1.02	**56.54±1.14**	**55.45±1.36**	69.98±0.96	69.31±1.17
	AutoLINC-PubMed	67.27±0.71	67.29±0.70	54.76±1.60	53.66±1.88	**71.50±0.67**	**70.42±1.04**

Table 7: Comparison with SOTA non-loss function engineering methods.

GNN	Dataset / Imbalance Ratio	Cora 5 bACC	Cora 5 F1	Cora 10 bACC	Cora 10 F1	CiteSeer 5 bACC	CiteSeer 5 F1	CiteSeer 10 bACC	CiteSeer 10 F1	PubMed 5 bACC	PubMed 5 F1	PubMed 10 bACC	PubMed 10 F1
GCN	GraphSMOTE	69.53±0.85	69.17±1.02	65.01±0.95	64.40±1.08	50.50±0.71	47.16±0.98	41.75±1.45	36.71±2.16	67.83±0.73	65.53±1.21	64.90±1.04	61.65±1.95
	GraphENS	75.68±0.39	75.51±0.57	70.03±0.83	69.98±0.89	62.81±0.82	62.25±0.86	54.32±1.53	51.53±2.20	75.09±0.66	73.92±0.91	72.59±1.27	70.44±1.63
	GraphSHA	76.69±0.32	**76.45±0.36**	73.37±0.60	**73.09±0.65**	62.50±1.69	61.81±1.82	56.42±1.88	52.97±2.26	**75.40±0.64**	73.81±0.69	72.35±0.98	69.66±1.37
	AutoLINC	75.26±0.54	74.79±0.47	70.21±0.67	69.67±0.79	60.64±1.05	60.39±1.01	56.56±1.97	56.09±2.06	74.42±0.86	**74.61±0.88**	72.43±1.03	72.42±1.07
	AutoLINC(GraphSHA)	**76.88±0.43**	76.24±0.40	**73.75±0.77**	72.93±0.86	**63.10±1.31**	**62.59±1.39**	**58.57±1.88**	**56.68±2.25**	75.27±0.58	74.53±0.65	**73.32±1.10**	**72.74±0.91**
GAT	GraphSMOTE	71.84±0.55	71.44±0.55	68.93±0.86	68.16±0.97	60.66±1.04	59.73±1.43	53.94±2.59	52.52±3.14	69.96±1.27	67.11±1.72	67.93±0.90	63.28±1.58
	GraphENS	75.18±0.68	74.39±0.68	70.02±0.94	68.99±1.00	58.94±1.63	57.79±1.82	52.13±1.56	48.54±2.07	74.10±0.80	72.85±1.04	71.22±1.04	68.22±1.39
	GraphSHA	73.07±0.21	73.40±0.33	68.16±1.00	68.31±1.14	60.47±1.87	58.95±2.33	53.64±2.61	48.84±3.22	74.00±0.92	72.56±1.11	71.56±1.16	68.33±1.65
	AutoLINC	75.57±0.42	75.24±0.40	70.63±1.01	70.16±1.04	**63.58±0.76**	**63.29±0.71**	**58.48±1.57**	**57.38±1.89**	74.51±0.82	**74.50±0.79**	72.57±0.88	72.27±1.02
	AutoLINC(GraphSHA)	**75.85±0.55**	**75.76±0.58**	**71.97±0.74**	**71.57±0.76**	61.97±1.21	61.44±1.26	57.26±1.74	55.07±2.11	**74.86±0.54**	74.22±0.71	**73.71±0.87**	**72.60±0.87**
SAGE	GraphSMOTE	63.64±1.03	63.12±1.19	58.53±1.01	56.59±1.35	51.93±1.22	50.39±1.33	44.52±2.00	40.15±2.66	72.02±0.32	69.50±0.65	68.42±0.88	62.97±1.23
	GraphENS	72.35±0.53	72.39±0.70	66.58±0.74	66.35±0.83	61.11±0.77	60.63±0.77	53.91±1.24	52.23±1.51	72.90±0.62	72.59±0.69	71.34±0.79	69.56±0.96
	GraphSHA	75.74±0.44	75.24±0.54	72.47±0.52	71.91±0.60	**62.09±1.53**	**61.81±1.57**	57.01±1.85	53.95±2.31	74.48±0.53	73.91±0.62	72.84±0.81	70.79±1.19
	AutoLINC	74.15±0.69	73.80±0.70	68.51±1.02	68.49±1.02	61.41±0.85	61.13±0.78	56.78±1.27	**56.16±1.37**	72.59±0.71	72.08±0.74	70.97±1.04	70.32±0.96
	AutoLINC(GraphSHA)	**76.76±0.53**	**76.07±0.47**	**73.50±0.69**	**72.72±0.83**	61.09±1.56	59.88±1.94	**57.03±1.80**	53.86±2.27	**75.44±0.38**	**75.11±0.31**	**73.29±1.12**	**72.37±1.29**

Table 8: Experimental results under different sizes of validation set. Here, we set it to 5, 10, and 20 nodes per class.

Datasets & GNNs	Methods	5 bACC	5 F1	10 bACC	10 F1	20 bACC	20 F1
Cora-GCN	BalancedSoftmax	67.48±1.04	66.20±1.28	68.96±0.55	**68.67±0.52**	68.96±0.55	**68.73±0.51**
	BalancedSoftmax+TAM	66.98±1.41	65.88±1.55	67.48±1.40	66.37±1.59	68.76±0.91	68.73±1.02
	AutoLINC	**68.65±1.26**	**67.33±1.13**	**69.08±0.87**	67.80±0.71	**69.51±0.97**	67.30±1.07
CiteSeer-GAT	BalancedSoftmax	51.18±1.82	47.73±2.57	51.08±1.32	48.84±1.79	53.70±1.64	50.31±2.03
	BalancedSoftmax+TAM	51.00±1.49	47.98±2.31	51.70±1.69	48.48±2.37	53.51±1.64	51.99±1.83
	AutoLINC	**54.46±2.27**	**53.18±2.47**	**54.69±2.43**	**53.32±2.76**	**55.39±2.24**	**53.61±2.71**
PubMed-SAGE	BalancedSoftmax	68.98±0.76	65.57±1.21	69.09±0.74	66.78±1.16	69.77±0.73	68.63±0.87
	BalancedSoftmax+TAM	**69.26±0.91**	**66.89±1.38**	**69.99±0.71**	**68.53±1.03**	69.23±0.65	**69.10±0.73**
	AutoLINC	67.54 ± 0.83	65.78 ± 1.26	69.58±0.79	67.81±1.34	**70.14±0.47**	68.20±1.02

search space of AutolossZero, modified to accommodate graph data by excluding some inapplicable primitive operators. Correspondingly, its loss function rejection strategies and proxy task align with AutoLINC to ensure fairness. AutolossZero-B is a variant that introduces the parameter N into the loss functions within its search space, while the rest remains unchanged. Under the same time constraints (1 hour CPU time), experimental results are presented in Table 9. With the exception of differences in the search method and a search depth set to 4, all other aspects remain consistent with our AutoLINC method. AutoLINC achieves better performance than Autoloss-Zero. This is attributed to AutoLINC's inclination to discover structurally simple trees in proxy tasks, whereas AutolossZero tends to find complex loss functions. In the task of node classification, structurally simple trees are more likely to exhibit good performance in overall task evaluation.

5.6. Ablation Study

The ablation studies were conducted on Cora using the GCN model, with a total of 10 trials. In Figure 2, we showcase the contrasting impacts of the Basic Check Strategy and the Early Rejection Strategy, both essential components of our approach. The Basic Check strategy consistently outperforms the Naive approach within the same number of evaluations, thus expediting the search process. The adaptability of the rejection threshold, which aligns with the Top 10, ensures that as the search progresses, only loss functions that converge rapidly and demonstrate superior

Table 9: The comparative results of loss functions searched by different frameworks.

GNN	Dataset Imbalance Ratio:10	Cora bAcc	 F1	CiteSeer bAcc	 F1	PubMed bAcc	 F1
GCN	AutoLoss-Zero-A	55.57±1.02	54.71±1.08	39.63±1.07	33.02±1.74	65.79±1.61	57.17±3.43
	AutoLoss-Zero-B	62.64±0.89	61.58±1.21	55.73±1.67	54.29±1.94	72.41±1.08	71.71±1.19
	AutoLINC(1 hour)	**70.60±0.62**	**69.13±0.53**	**56.63±1.97**	**55.98±2.05**	**72.90±0.71**	**72.56±0.86**
GAT	AutoLoss-Zero-A	63.07±0.84	60.13±0.89	46.44±0.99	41.05±1.47	**69.43±1.19**	**65.85±2.48**
	AutoLoss-Zero-B	68.03±0.49	67.67±0.53	55.61±1.24	55.50±1.12	65.82±1.77	64.65±1.57
	AutoLINC(1 hour)	**71.34±0.90**	**71.16±0.91**	**57.78±1.62**	**57.19±1.76**	63.69±1.69	62.65±2.27
SAGE	AutoLoss-Zero-A	55.34±0.32	51.30±0.56	43.66±0.77	36.78±1.21	65.80±1.82	64.71±1.94
	AutoLoss-Zero-B	68.68±0.86	**68.37±0.87**	42.63±1.63	37.83±1.91	**72.46±0.60**	**70.49±0.88**
	AutoLINC(1 hour)	**68.92±0.82**	67.99±0.90	**56.59±1.41**	**55.56±1.53**	71.08±0.88	70.02±0.96

performance are retained.

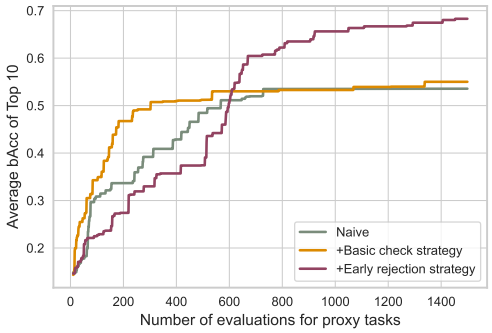


Figure 2: This figure illustrates the average scores of the Top 10 loss metrics during the search process. Naive represents MCTS without the proposed strategies.

Further insight into the efficiency of AutoLINC is provided in Table 10, which details the number of valid loss functions explored within a 12-hour timeframe. The integration of the Basic Check Strategy accelerates the search process by nearly 6-fold, while the addition of the Early Rejection Strategy enhances search efficiency by almost 8-fold. These findings underscore the substantial improvements in the search efficiency of AutoLINC due to the incorporation of these two loss-checking strategies.

Table 10: AutoLINC explores the number of valid loss functions within 12 hours using different strategy combinations.

Basic Check Strategy	Early Rejection Strategy	No. Loss Functions	Speed-Up
✗	✗	8.3×10^4	1×
✓	✗	4.8×10^5	5.7×
✓	✓	6.5×10^5	7.8×

Time In Figure 3, AutoLINC exhibits the highest time consumption among the compared methods. However, this time consumption is justified by the ratio of performance gain achieved. Additionally, the discovered loss functions demonstrate a degree of transferability, suggesting the potential for addressing the problem with reduced time costs.

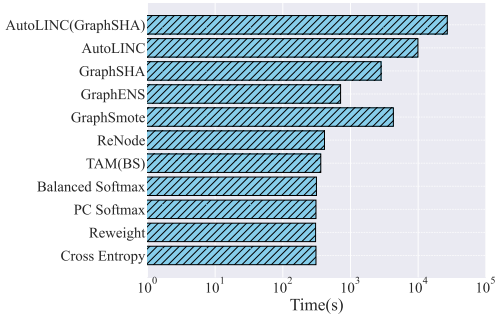


Figure 3: The runtime on GCN across PubMed datasets using 10 random seeds.

6. Conclusions

This paper introduces an automatic loss function search framework with high performance and generalization capabilities for addressing class-imbalanced node classification problems. Compared to SOTA loss functions, the functions discovered within this framework demonstrate significant improvements in classification performance, affirming the effectiveness of the proposed framework. We also find that the loss functions discovered based on a single GNN and dataset exhibit transferability to homogeneous datasets. Crucially, they compete favorably with SOTA loss functions. Furthermore, we observe that loss functions discovered under high class-imbalance ratios generalize well to scenarios with lower class-imbalance ratios, highlighting the adaptability of our proposed approach. Finally, we validate that the employed Basic Check and Early Rejection Strategies can accelerate the operation of the search algorithm. Further research is warranted to design an Autoloss framework with transferability on heterogeneous graph.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62206205, in part by the Young Talent Fund of Association for Science and Technology in Shaanxi, China under Grant 20230129, in part by the Guangdong High-level Innovation Research Institution Project under Grant 2021B0909050008, and in part by the Guangzhou Key Research and Development Program under Grant 202206030003.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Chen, D., Lin, Y., Zhao, G., Ren, X., Li, P., Zhou, J., and Sun, X. Topology-imbalance learning for semi-supervised node classification. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 29885–29897. Curran Associates, Inc., 2021.

Coulom, R. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Gao, B., Gouk, H., Yang, Y., and Hospedales, T. Loss function learning for domain generalization by implicit gradient. In *International Conference on Machine Learning*, pp. 7002–7016. PMLR, 2022.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Hong, Y., Han, S., Choi, K., Seo, S., Kim, B., and Chang, B. Disentangling label distribution for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6626–6636, June 2021.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. Automata theory, languages, and computation. *International Edition*, 24(2), 2006.

Japkowicz, N. and Stephen, S. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1945–1954. JMLR. org, 2017.

Li, C., Yuan, X., Lin, C., Guo, M., Wu, W., Yan, J., and Ouyang, W. Am-lfs: Automl for loss function search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8410–8419, 2019.

Li, H., Tao, C., Zhu, X., Wang, X., Huang, G., and Dai, J. Auto seg-loss: Searching metric surrogates for semantic segmentation. In *International Conference on Learning Representations*, 2021.

Li, H., Fu, T., Dai, J., Li, H., Huang, G., and Zhu, X. Autoloss-zero: Searching loss functions from scratch for generic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1009–1018, June 2022a.

Li, W.-Z., Wang, C.-D., Xiong, H., and Lai, J.-H. GraphSHA: Synthesizing harder samples for class-imbalanced node classification. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 1328–1340, New York, NY, USA, 2023. Association for Computing Machinery.

Li, Z., Ji, J., Ge, Y., and Zhang, Y. Autolossgen: Automatic loss function generation for recommender systems. SIGIR '22, pp. 1304–1315, New York, NY, USA, 2022b. Association for Computing Machinery.

Liu, P., Zhang, G., Wang, B., Xu, H., Liang, X., Jiang, Y., and Li, Z. Loss function discovery for object detection via convergence-simulation driven search. In *International Conference on Learning Representations*, 2021.

Liu, Q. and Lai, J. Stochastic loss function. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04): 4884–4891, Apr. 2020.

Liu, Z., Zeng, Z., Qiu, R., Yoo, H., Zhou, D., Xu, Z., Zhu, Y., Weldemariam, K., He, J., and Tong, H. Topological augmentation for class-imbalanced node classification, 2023.

Ma, Y., Tian, Y., Moniz, N., and Chawla, N. V. Class-imbalanced learning on graphs: A survey. *arXiv preprint arXiv:2304.04300*, 2023.

McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pp. 43–52, New York, NY, USA, 2015. Association for Computing Machinery.

Park, J., Song, J., and Yang, E. GraphENS: Neighbor-aware ego network synthesis for class-imbalanced node classification. In *International Conference on Learning Representations*, 2022.

Qian, Y., Zhang, C., Zhang, Y., Wen, Q., Ye, Y., and Zhang, C. Co-modality graph contrastive learning for imbalanced node classification. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 15862–15874. Curran Associates, Inc., 2022.

Raymond, C., Chen, Q., Xue, B., and Zhang, M. Learning symbolic model-agnostic loss functions via meta-learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):13699–13714, 2023.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 33, pp. 4780–4789, 2019.

Real, E., Liang, C., So, D., and Le, Q. AutoML-zero: Evolving machine learning algorithms from scratch. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8007–8019. PMLR, 13–18 Jul 2020.

Ren, J., Yu, C., sheng, s., Ma, X., Zhao, H., Yi, S., and Li, h. Balanced meta-softmax for long-tailed visual recognition. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 4175–4186. Curran Associates, Inc., 2020.

Song, J., Park, J., and Yang, E. TAM: Topology-aware margin loss for class-imbalanced node classification. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 20369–20383. PMLR, 17–23 Jul 2022.

Sun, F., Liu, Y., Wang, J.-X., and Sun, H. Symbolic physics learner: Discovering governing equations via monte carlo tree search. In *The Eleventh International Conference on Learning Representations*, 2023.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wang, X., Wang, S., Chi, C., Zhang, S., and Mei, T. Loss function search for face recognition. In *International Conference on Machine Learning*, pp. 10029–10038. PMLR, 2020.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386.

Xu, H., Zhang, H., Hu, Z., Liang, X., Salakhutdinov, R., and Xing, E. Autoloss: Learning discrete schedule for alternate optimization. In *International Conference on Learning Representations*, 2019a.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019b.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 40–48, New York, New York, USA, 20–22 Jun 2016. PMLR.

Zeng, L., Li, L., Gao, Z., Zhao, P., and Li, J. Imgcl: Revisiting graph contrastive learning on imbalanced node classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11138–11146, Jun. 2023.

Zhao, T., Zhang, X., and Wang, S. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, pp. 833–841, New York, NY, USA, 2021a. Association for Computing Machinery.

Zhao, X., Liu, H., Fan, W., Liu, H., Tang, J., and Wang, C. Autoloss: Automated loss function search in recommendations. In Zhu, F., Ooi, B. C., and Miao, C. (eds.), *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pp. 3959–3967. ACM, 2021b.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. ISSN 2666-6510.

Algorithm 1 AutoLINC

Input: Grammar $\mathcal{G} = (U, \Sigma, R, O)$, node feature matrix X, node label Y, GNN model f_θ, measure σ, number of trails P, number of episode EP, number of simulation B, t_{max}
Output: Optimal loss function $\mathcal{L}^\star$
Initialize $M = \emptyset$
for $p \leftarrow 1, P$ **do**
  for $e \leftarrow 1, EP$ **do**
    Selection: Initialize $s_0 = \emptyset, t = 0, NT = [O]$
    while s_t expandable and $t < t_{max}$ **do**
      $a_{t+1} \leftarrow \arg\max_{a \in \mathcal{A}} UCT(s_t, a)$
      Obtain s_{t+1}, NT after a_{t+1}; $t \leftarrow t + 1$
    end while
    Expansion: Randomly take an unvisited path with a Obtain s_{t+1}, NT after a; $t \leftarrow t + 1$
    if $NT = \emptyset$ **then**
      Validate the Legitimacy of $\mathcal{L}$ and check $\mathcal{L}$ via the Loss Check Strategy
      Train the proxy task with $\mathcal{L}$
      $r \leftarrow \sigma(f_\theta(X_{val}, Y_{val}))$
      Record $\mathcal{L}, r$ in M, **Backpropagate** and finish the episode
    end if
    Simulation: Fix the state s_t and NT Initialize $r = 0$
    for $b \leftarrow 1, B$ **do**
      while s_t non-terminal and $t < t_{max}$ **do**
        $a \leftarrow Random(\mathcal{A})$
        Obtain s_{t+1}, NT after a_{t+1}; $t \leftarrow t + 1$
      end while
      if $NT = \emptyset$ **then**
        Validate the Legitimacy of $\mathcal{L}$ and check
        $\mathcal{L}$ via the Loss Check Strategy
        Train the proxy task with $\mathcal{L}$
        $r \leftarrow \sigma(f_\theta(X_{val}, Y_{val}))$
        Record $\mathcal{L}, r$ in M
      end if
    end for
    Backpropagate: Back-update parent nodes based on simulation results all the way back to the root node
  end for
end for
Get the top-10 loss functions M_{top} from M based on reward
Initialize $best = 0$, $\mathcal{L}^* = \emptyset$
for $l \leftarrow 1, 10$ **do**
  $\mathcal{L} \leftarrow M_{top}(l)$
  Train the complete task with $\mathcal{L}$
  $r \leftarrow \sigma(f_\theta(X_{val}, Y_{val}))$
  if $r > best$ **then**
    $best \leftarrow r$;
    $\mathcal{L}^\star \leftarrow \mathcal{L}$
  end if
end for

A. Experimental Setup

Datasets We validate AutoLINC on well-known citation networks (Yang et al., 2016), comprising three datasets: Cora, CiteSeer, and PubMed, as well as Amazon's co-purchase networks (McAuley et al., 2015), which consist of two datasets: Computers and Photo. In the case of citation networks, we employ training, validation, and testing set splits as described in (Yang et al., 2016). For Amazon networks, the data set is divided into five distinct partitions using five different random seeds, following the methodology in (Chen et al., 2021). A label ratio of 0.01 is maintained. To generate class-imbalanced datasets, we adopt the step imbalance method as detailed in (Park et al., 2022; Zhao et al., 2021a). The minority class contains an equal number of instances, denoted as n, while the majority class consists of $n \times \rho$ instances, where ρ represents the class imbalance ratio. For this experiment, we set ρ to 5 or 10.

Baseline To evaluate the effectiveness of the loss functions learned by AutoLINC, we compare them with several baseline methods, including cross-entropy, re-weight (Japkowicz & Stephen, 2002), balanced softmax (Ren et al., 2020), PC softmax

Table 11: The searched loss functions, represented in the table, exemplify their effectiveness. For instance, the entry labeled ID A showcases the evaluation of node classification accuracy using GCN on the Cora training set, highlighting the high-performance loss function discovered by AutoLINC. Here, $\rho = 10$.

ID	Dataset	GNN	Discovered Loss Function
A	Cora	GCN	$\exp(\tanh^2\left(\frac{1}{N} \times (-y) + \hat{y}\right))$
B	CiteSeer	GCN	$\left(-\left(N \times \hat{y}\right) + y\right)^2$
C	PubMed	GCN	$\hat{y} + \left(\frac{1}{N} \times (-y) + \hat{y}\right)^2$
D	Cora	GAT	$\left(-\tanh\left(\frac{1}{N} \times (-y) + \hat{y}\right)\right)^2$
E	CiteSeer	GAT	$\left(\tanh\left(-\hat{y}\right) + \tanh\left(y\right) \times \frac{1}{N}\right)^2$
F	PubMed	GAT	$\left(\exp(\frac{1}{N} \times (-y) \times 2) + \hat{y}\right)^2$
G	Cora	SAGE	$\tanh^2\left(\tanh\left(\frac{1}{N} \times (-y) + \hat{y}\right)\right)$
H	CiteSeer	SAGE	$\left(\hat{y} + \left(-\frac{1}{\log(\tanh(y)) + N}\right)\right)^2$
I	PubMed	SAGE	$\left(-\left(\tanh\left(y\right) + \hat{y} \times (-N)\right)\right) \times \hat{y}$
J	Amazon-computers	GCN	$\left(\left(\hat{y} + N\right) + \log\left(\log\left(\frac{1}{\sqrt{y}}\right)\right)\right)^2$
K	Amazon-photo	GCN	$\left(\hat{y} + \sqrt{\sqrt{N \times log^2\left(y\right)}}\right)^2$
L	Amazon-computers	GAT	$\log\left(\left(y + \left(-\hat{y}\right)\right)^2 + \log\left(N\right)\right)$
M	Amazon-photo	GAT	$\left\lvert -y + \left(\hat{y} + \tanh^2\left(N\right)\right)\right\rvert$
N	Amazon-computers	SAGE	$\left(\hat{y} + \left(N + \log\left(\log\left(\frac{1}{\sqrt{y}}\right)\right)\right)\right)^2$
O	Amazon-photo	SAGE	$\hat{y} \times \left(\hat{y} + \left(N + \log\left(-\log\left(y\right)\right)\right)\right)$

(Hong et al., 2021), ReNode (Chen et al., 2021), and TAM (Song et al., 2022). Cross-entropy and re-weight serve as fundamental baseline approaches, while balanced softmax and PC softmax are designed to address long-tail issues in computer vision. On the other hand, ReNode and TAM are specifically tailored to tackle imbalanced node classification problems. For ReNode, we enhance it by combining it with the focal loss, with a focal hyperparameter set to 2.0, and topology imbalance bounds set to 0.5 and 1.5. In the case of TAM, we carefully select parameters based on the average of accuracy and F1 score from the recommended settings in its original paper. These parameters include the Anomalous Connectivity Margin term coefficients ($\{0.25, 0.5, 1.5, 2.5\}$), the Anomalous Distribution-aware Margin ($\{0.125, 0.25, 0.5\}$), and the minimum temperature of class-wise temperature ($\{0.8, 1.2\}$). For GraphSMOTE, we choose the GraphSMOTE$_{\text{O}}$ version which is without pretraining, as it shows excellent performance among multiple versions. For GraphENS, we set the feature masking hyperparameter k at 0.01 and the temperature τ at 2. For GraphSHA, we keep the default setting in the code.

Experimental Settings To address the sparsity of the search space and the tendency for simulated formulas to exceed depth limits and include nonterminal nodes, we employ a larger number of simulations, specifically 100. We consider the maximum value simulated during these 100 simulations as the reward score for the state. In the context of citation network datasets, one trial comprises 100,000 episodes, and we conduct a single trial. For the Amazon co-purchase network, one trial involves 200,000 episodes, and we perform 10 trials. In the case of citation networks, all experiments are conducted ten times to calculate bAcc and F1 scores, while for the Amazon networks, each partition is executed three times. It's important to note that all experiments in this paper are performed using a single NVIDIA GeForce RTX 3090 GPU.

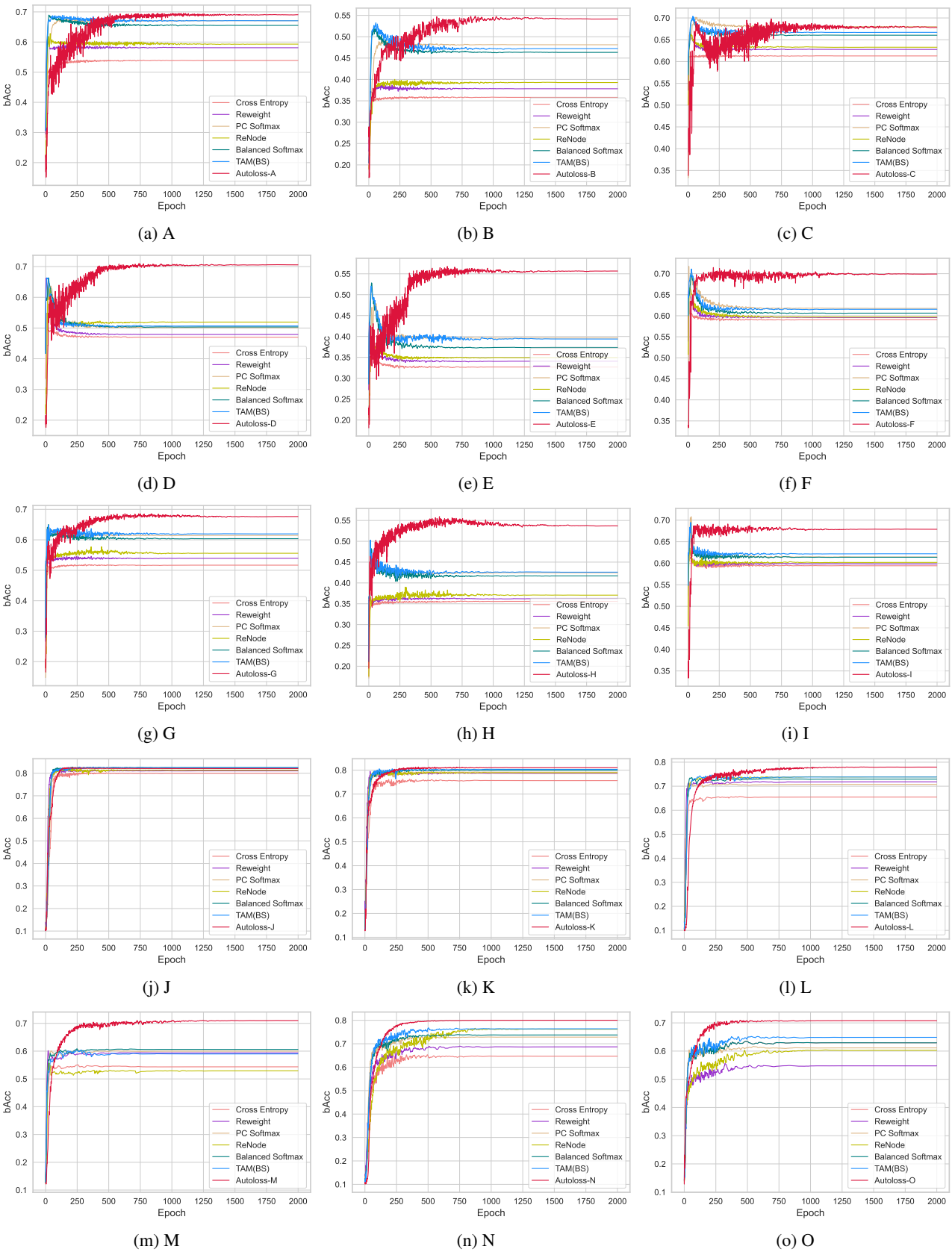
GNN Settings We perform experiments utilizing three prominent GNN models: GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and SAGE (Hamilton et al., 2017). All three GNNs consist of a 2-layer neural network with a hidden layer dimension of 256. Other hyperparameters align with those detailed in (Song et al., 2022). This includes employing the Adam (Kingma & Ba, 2014) optimizer, training the model for 2000 epochs, and selecting the optimal model parameters based on the average accuracy and F1 scores on the validation set. The initial learning rate is configured at 0.1 and undergoes halving when there is no improvement in the validation set loss for 100 consecutive generations. Additionally, weight decay is set to 0.0005 and is applied to all learnable parameters except for the last convolutional layer.

Table 12: The searched loss functions by AutoLINC combined with GraphSHA. The evaluation results of these loss functions are presented in Table 7.

Frame	Dataset	GNN	Discovered Loss Function
AUTOLINC(GraphSHA)	Cora	GCN	$\hat{y} \times ((\hat{y} - y) \times N + \hat{y})$
AUTOLINC(GraphSHA)	CiteSeer	GCN	$(y - N \times \hat{y}^3)^2$
AUTOLINC(GraphSHA)	PubMed	GCN	$\hat{y} + \frac{1}{\exp(N \times \hat{y} \times y)}$
AUTOLINC(GraphSHA)	Cora	GAT	$N + (y + \sqrt{-\hat{y}})^2$
AUTOLINC(GraphSHA)	CiteSeer	GAT	$N \times (y - \hat{y})^4$
AUTOLINC(GraphSHA)	PubMed	GAT	$\hat{y} + y \times \exp(N \times \lvert\hat{y}\rvert^2)$
AUTOLINC(GraphSHA)	Cora	SAGE	$N \times \lvert\hat{y} - \tanh(\hat{y} + y)\rvert$
AUTOLINC(GraphSHA)	CiteSeer	SAGE	$N \times \tanh^2(\tanh(y^2 - \hat{y}))$
AUTOLINC(GraphSHA)	PubMed	SAGE	$\frac{\lvert\hat{y}\rvert}{N \times \sqrt{\hat{y}+y}}$

Table 13: The searched loss functions by AutolossZero and AutoLINC. The evaluation results of these loss functions are presented in Table 9.

Frame	Dataset	GNN	Discovered Loss Function
AUTOLOSS-ZERO-A	Cora	GCN	$((\hat{y} + \log y) \times (\tanh 1 + \hat{y} + y))^2$
AUTOLOSS-ZERO-A	CiteSeer	GCN	$\tanh(\tanh(-\hat{y})) + \tanh\sqrt{\hat{y} + y}$
AUTOLOSS-ZERO-A	PubMed	GCN	$\lvert 1 + y\rvert \times (\hat{y} - y) \times \hat{y}^8$
AUTOLOSS-ZERO-A	Cora	GAT	$\tanh\left(\sqrt{\hat{y}} + y\right) + (\lvert\hat{y}\rvert \times \exp(y))^2$
AUTOLOSS-ZERO-A	CiteSeer	GAT	$\frac{\left(\left(y^2 - y\right) \times \left(-y \times \sqrt{y}\right) \times \log(\log(1))\right)}{\exp}$
AUTOLOSS-ZERO-A	PubMed	GAT	$((y + 1) \times \hat{y})^2 + \left(\exp + \sqrt{y}\right)$
AUTOLOSS-ZERO-A	Cora	SAGE	$\left(\left(\tanh \hat{y} + \hat{y}^2\right) - 2 \times y\right)^2$
AUTOLOSS-ZERO-A	CiteSeer	SAGE	$\left(y + \sqrt{\hat{y} + y} + \exp\sqrt{-\hat{y}}\right)$
AUTOLOSS-ZERO-A	PubMed	SAGE	$\lvert y \times \hat{y} - 2\rvert$
AUTOLOSS-ZERO-B	Cora	GCN	$\left(\exp(y) \times \hat{y}^2 + \tanh \hat{y}\right) \times (y \times \lvert N\rvert + y + N + \hat{y})$
AUTOLOSS-ZERO-B	CiteSeer	GCN	$\tanh((-y + \hat{y}) \times (N + 2 \times \hat{y}))$
AUTOLOSS-ZERO-B	PubMed	GCN	$-\sqrt{y} \times (\tanh \hat{y} + \hat{y}) + (\log N + N) \times \hat{y}^2$
AUTOLOSS-ZERO-B	Cora	GAT	$(\tanh(\exp(\hat{y})) + 2 \times y \times \hat{y}) \times (N + y + \hat{y}) \times \hat{y}$
AUTOLOSS-ZERO-B	CiteSeer	GAT	$\sqrt{\hat{y} + N} \times \tanh \hat{y} + \hat{y}^2 \times y \times \sqrt{N}$
AUTOLOSS-ZERO-B	PubMed	GAT	$\frac{1}{N \times \tanh \hat{y} + \hat{y} + \frac{1}{y}}$
AUTOLOSS-ZERO-B	Cora	SAGE	$2 \times \hat{y} \times (\tanh(N \times \hat{y}) - y + \tanh \hat{y})$
AUTOLOSS-ZERO-B	CiteSeer	SAGE	$\lvert\hat{y}\rvert + \hat{y} + (y + \hat{y}) \times y \times \hat{y} + 2 \times N \times \sqrt{\exp(\hat{y})}$
AUTOLOSS-ZERO-B	PubMed	SAGE	$\hat{y}^4 \times \tanh \hat{y} \times N - \tanh(y \times \hat{y})$
AUTOLINC(1 HOUR)	Cora	GCN	$(\tanh(N \times \hat{y}) - y)^2$
AUTOLINC(1 HOUR)	CiteSeer	GCN	$(y - N \times \hat{y})^2$
AUTOLINC(1 HOUR)	PubMed	GCN	$N \times \hat{y}^2 - y \times \hat{y}$
AUTOLINC(1 HOUR)	Cora	GAT	$\sqrt{\hat{y}} \times (\hat{y} - \frac{y}{N})$
AUTOLINC(1 HOUR)	CiteSeer	GAT	$\hat{y}^2 - \frac{y \times \hat{y}}{N}$
AUTOLINC(1 HOUR)	PubMed	GAT	$\exp(-\hat{y} \times (y + N)) \times y$
AUTOLINC(1 HOUR)	Cora	SAGE	$(\tanh(N \times \hat{y}) + \sqrt{-y})^2$
AUTOLINC(1 HOUR)	CiteSeer	SAGE	$N \times \hat{y}^2 - y \times \hat{y}$
AUTOLINC(1 HOUR)	PubMed	SAGE	$N \times \hat{y}^2 - y \times \hat{y}$



(a) A (b) B (c) C

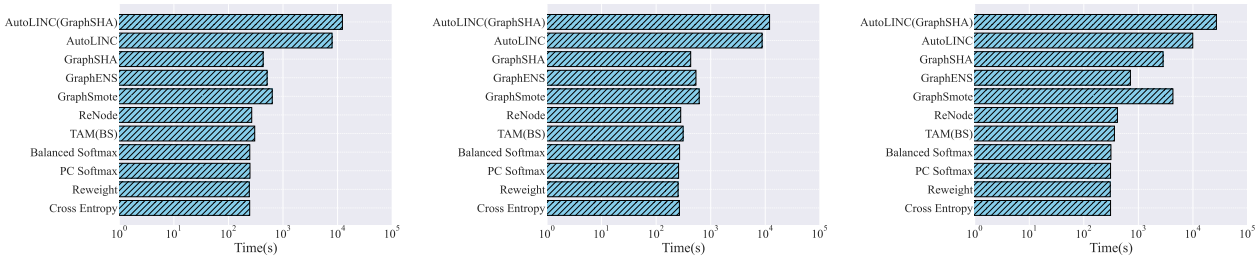
(d) D (e) E (f) F

(g) G (h) H (i) I

(j) J (k) K (l) L

(m) M (n) N (o) O

Figure 4: The convergence of loss function discovered in Table 11.



(a) Cora (b) CiteSeer (c) PubMed

Figure 5: The runtime on GCN across three citation network datasets using 10 random seeds.