How do loss functions impact the performance of graph neural networks?

Gabriel Jonas da S. Duarte
Dept. Computer Science
Federal Institute of Ceará
Maracanaú, Brazil
g.jonas.duarte@gmail.com

Tamara Arruda Pereira
Dept. Computer Science
Federal Institute of Ceará
Fortaleza, Brazil
tamaraarrudap@gmail.com

Erik Jhones F. Nascimento
Dept. Computer Science
Federal Institute of Ceará
Fortaleza, Brazil
erikjhonesf@gmail.com

Diego Mesquita
Dept. Computer Science
Aalto University
Espoo, Finland
diego.mesquita@aalto.fi

Amauri Holanda de Souza Jr.
Dept. Computer Science
Federal Institute of Ceará
Fortaleza, Brazil
amauriholanda@ifce.edu.br

***Abstract*—Graph neural networks (GNNs) have become the *de facto* approach for supervised learning on graph data. To train these networks, most practitioners employ the categorical cross-entropy (CE) loss. We can attribute this largely to the probabilistic interpretation of CE, since it corresponds to the negative log of the categorical/softmax likelihood. Nonetheless, recent works have shown that deep learning models can benefit from adopting other loss functions. For instance, neural networks trained with symmetric losses (e.g., mean absolute error) are robust to label noise. Perhaps surprisingly, the effect of using different losses on GNNs has not been explored. In this preliminary work, we gauge the impact of different loss functions to the performance of GNNs for node classification under i) noisy labels and ii) different sample sizes. In contrast to findings on Euclidean domains, our results for GNNs show that there is no significant difference between models trained with CE and other classical loss functions on both aforementioned scenarios.**

***Index Terms*—Graph neural networks, semi-supervised node classification, loss functions, noise-tolerant networks.**

I. Introduction

Graph neural networks (GNNs) [1], [2] have become the main workhorse for analyzing graph-structured data. Similarly to convolutional neural networks, GNNs architectures interleave convolutional [3], [4], [5] and pooling layers [6], [7], [8] to extract meaningful graph representations. Remarkably, these models have led to breakthroughs in many tasks such as antibiotic design [9], recommender systems [10], machine translation [11], and physical systems simulation [12].

Similarly to classical deep learning models, the learning phase of GNNs consists of minimizing a loss function using gradient-based optimization (e.g., Adam [13]). Notably, the cross-entropy (CE) loss function is the ubiquitous choice for classification tasks. While the CE is arguably an intuitive choice, due to its direct relationship to classical logistic and multinomial regression models [14], there are many other options to choose from — e.g., mean squared-error (MSE), mean absolute error (MAE) and hinge-losses [15]. In fact, there is evidence from research in traditional neural networks that choosing the CE loss can lead to sub-optimal accuracy [16], [17].

For instance, Manwani and Sastry [18] show that the 0-1 loss is robust to symmetric label noise, i.e., when the probability of observing label i, given that the true label is j, is the same for all $i \neq j$. Additionally, Ghosh *et al.* [16] show that, in the context of multi-class classification, neural networks trained using MAE are robust to symmetric label noise, while networks trained using CE are not.

Despite the central role played by loss functions, to the best of our knowledge, there is no work exploring the impact of different loss functions on the performance of GNNs. In this work, we focus on semi-supervised node classification and evaluate the importance of different loss functions in two distinct scenarios: i) varying amount of labeled data; and ii) noisy training labels. The former reflects the reality of many semi-supervised settings, in which labeled data are scarce. The latter commonly applies when data are labeled by humans, and are therefore inherently noisy. More specifically, we run experiments using four loss functions: CE, MSE, MAE and the Weston-Watkins hinge loss [15]. Our results show that all loss functions result in similar accuracy for any amount correctly-labeled data. In addition, our experiments suggest that MAE, MSE and large-margin losses are not robust to symmetric label noise on semi-supervised node classification tasks.

The remainder of this work is organized as follows. Section II provides a brief review of GNNs and common loss functions. Section III evaluates the impact of different loss functions when node labels are noisy. Section IV gauges the performance of choosing distinct losses when few labeled nodes are available. Section V discusses related works. Finally, Section VI draws conclusions and points directions for future works.

II. Background

A. Notation and problem definition

We represent a graph G with nodes $V = \{1, 2, \dots, n\}$ as a pair $(\boldsymbol{A}, \boldsymbol{X})$ where $\boldsymbol{A} \in \{0,1\}^{n \times n}$ denotes a symmetric adjacency matrix and $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ is a matrix whose rows comprise d-dimensional node features. Additionally, we define the diagonal degree matrix $\boldsymbol{D}$ of G such that $D_{ii} := \sum_j A_{ij}$. Also, we denote the normalized graph Laplacian matrix of G as $\boldsymbol{\Delta} := \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$.

In node classification problems, we are given a partially labeled graph G with labeled nodes V_l and unlabeled nodes V_u, such that $V_u \cup V_l = V$. The goal is to correctly predict the labels of V_u.

B. Graph Neural Networks

Although graph neural networks (GNNs) can come in different flavors, they are usually introduced either from a message-passing (spatial) or a spectral perspective. Here, we briefly describe three representative GNNs: graph convolutional networks (GCN, [19]), simplified graph convolutions (SGC, [20]), and graph attention networks (GAT, [21]).

From a spectral perspective, we can define convolutions in graph domains via the eigendecomposition of the graph Laplacian matrix [22]. Let $\boldsymbol{U}$ and $\boldsymbol{\Lambda}$ denote the eigenvectors and eigenvalues of the graph Laplacian $\boldsymbol{\Delta}$, respectively. The graph Fourier transform of a d-channel signal $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ on the vertices of the graph is given by $\hat{\boldsymbol{X}} = \boldsymbol{U}^\intercal \boldsymbol{X}$, and its inverse is $\boldsymbol{X} = \boldsymbol{U}\hat{\boldsymbol{X}}$. Using these operations, we define the graph convolution between a signal $\boldsymbol{X}$ and a filter $\boldsymbol{g}$ as

$$\boldsymbol{g} \star \boldsymbol{X} = \boldsymbol{U}\left((\boldsymbol{U}^\intercal \boldsymbol{g}) \odot (\boldsymbol{U}^\intercal \boldsymbol{X})\right) = \boldsymbol{U}\hat{\boldsymbol{G}}\boldsymbol{U}^\intercal \boldsymbol{X} \tag{1}$$

where $\hat{\boldsymbol{G}} = \mathrm{diag}(\hat{g}_1, \hat{g}_2, \dots, \hat{g}_n)$ comprises the spectral filter coefficients $\hat{g}_i$.

In practice, the non-parametric approach in Equation 1 is undesirable, since it comprises a large number of parameters and does not produce localized filters [3]. A solution is to approximate the filter with polynomials of the Laplacian eigenvalues, leading to polynomials of the Laplacian since $\boldsymbol{U}\boldsymbol{\Lambda}^k\boldsymbol{U}^\intercal = \boldsymbol{\Delta}^k$.

Let $\boldsymbol{H}^{(0)} = \boldsymbol{X}$; at layer ℓ, a generic polynomial spectral GNN computes

$$\boldsymbol{H}^{(\ell)} = \mathrm{ReLU}\left(\sum_{k=0}^{K} \boldsymbol{\Delta}^k \boldsymbol{H}^{(\ell-1)}\boldsymbol{\Theta}_k^{(\ell)}\right), \tag{2}$$

where $\boldsymbol{\Theta}^{(\ell)} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$ are the coefficients of the spectral filters. The output $\boldsymbol{H}^{(L)} \in \mathbb{R}^{n \times d_L}$ after L layers comprises representations for each node in G.

Graph Convolutional Networks (GCN). Building upon the spectral GNN in Equation 2, Kipf and Welling [19] propose using a first-order polynomial approximation, i.e, $K = 1$. In addition, GCNs apply two modifications: ii) adopting a single parameter matrix per layer such that $\boldsymbol{\Theta}^{(\ell)} = \boldsymbol{\Theta}_0^{(\ell)} = -\boldsymbol{\Theta}_1^{(\ell)}$; ii) employing the normalized adjacency matrix with added self-loops $\widetilde{\boldsymbol{A}} = (\boldsymbol{D}+\boldsymbol{I})^{-1/2}(\boldsymbol{A}+\boldsymbol{I})(\boldsymbol{D}+\boldsymbol{I})^{-1/2}$. As a result, the GCN layer recursively computes

$$\boldsymbol{H}^{(\ell)} = \mathrm{ReLU}\left(\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(\ell-1)}\boldsymbol{\Theta}^{(\ell)}\right). \tag{3}$$

Simplified Graph Convolutions (SGC). Wu *et al.* [20] further simplify the GCN model. In particular, SGC removes the nonlinearities and collapses the parameters of the resulting stacked linear layers into a single matrix $\boldsymbol{\Theta}$. After L feature propagation steps (or layers), SGC obtains node embeddings $\boldsymbol{H}$ given by

$$\boldsymbol{H} = \widetilde{\boldsymbol{A}}^L \boldsymbol{X}\boldsymbol{\Theta}. \tag{4}$$

Notably, SGC has proven to perform on par with GCNs on many node classification tasks while being much faster and more interpretable [20].

Graph Attention Networks (GAT). From a message-passing perspective, Velivckovic *et al.* [21] introduce an attention-based mechanism for graph data. Given a query node i, GATs compute attention weights over the neighbors of i. The representation of the query node i is then updated with the weighted average of the embeddings of its neighbors. Formally, we first compute the attention weight matrix $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{n \times n}$ at layer ℓ. More specifically, for each i, j such that $A_{ij} = 1$, we set

$$W_{ij}^{(\ell)} = \mathrm{LeakyReLU}\left(\boldsymbol{a}_\ell^\intercal \left[\boldsymbol{\Theta}^{(\ell)\intercal}\boldsymbol{h}_i^{(\ell-1)} || \boldsymbol{\Theta}^{(\ell)\intercal}\boldsymbol{h}_j^{(\ell-1)}\right]\right) \tag{5}$$

and the remaining entries are set to zero. The matrix $\boldsymbol{\Theta}^{(\ell)}$ and the vector $\boldsymbol{a}_\ell$ denote the model parameters of each layer. We then apply the softmax function to each row $W_{i:}$ of $\boldsymbol{W}$ to normalize the attention across the neighbors of each node $i \in V$. Denoting the normalized attention matrix by $\widetilde{\boldsymbol{W}}^{(\ell)}$, the output of the ℓ-th GAT layer is

$$\boldsymbol{H}^{(\ell)} = \mathrm{ReLU}\left(\widetilde{\boldsymbol{W}}^{(\ell)}\boldsymbol{H}^{(\ell-1)}\boldsymbol{\Theta}^{(\ell)}\right). \tag{6}$$

Note that Equation 5 and Equation 6 describe a single head, but in practice GATs employ a multi-head attention scheme. We refer to [21] for further details.

C. Loss functions

Loss functions are crucial components in supervised learning settings. It consists of the measure a learner minimizes to obtain a suitable hypothesis (model) for the problem at hand. Formally, let $\mathcal{Y}$ denote the space of targets (e.g., $\mathcal{Y} = \{+1, -1\}$ for binary classification) and $\mathcal{P}_{|\mathcal{Y}|}$ denote the $|\mathcal{Y}|$-dimensional probability simplex. In a typical classification problem, a loss function $\mathcal{L} : \mathcal{P} \times \mathcal{Y} \to \mathbb{R}^+$ maps a pair *target-prediction* to a non-negative scalar. We want to minimize over a given training dataset $\mathcal{D}$. In this minimization problem, the decision variable consists of model parameters (in the parametric case).

In the following, we briefly review the loss functions considered in this paper. Without loss of generality, we assume the target labels are represented as one-hot vectors of length C — the number of classes. For simplicity of notation, we

TABLE I

COMPARISON OF DIFFERENT LOSS FUNCTIONS FOR 30% OF MISLABELED SAMPLES. THE BEST RESULTS FOR EACH METHOD AND DATASET ARE IN BOLDFACE. THE LOSS EMPLOYED BY THE BEST PERFORMING MODELS CHANGE WITH THE METHOD AND DATASETS. THEREFORE, THERE IS NO CLEAR BEST LOSS FUNCTION FOR THIS SETTING.

Model	Loss	Cora	Citeseer	Pubmed	DBLP	WikiCs
SGC	CE	67.0±3.2	58.5±4.2	64.6±4.0	75.9±0.4	70.1±0.5
	HINGE	70.6±2.9	59.3±3.3	67.6±3.5	**76.6**±0.8	**70.9**±0.5
	MSE	**70.9**±3.5	**61.2**±7.5	**69.0**±3.4	75.9±0.9	70.0±0.7
	MAE	68.3±4.0	58.7±4.8	66.0±6.1	75.8±0.9	61.6±3.8
GCN	CE	**68.5**±3.0	58.9±3.8	63.9±6.3	**76.5**±0.4	68.3±0.7
	HINGE	67.9±2.2	**59.2**±2.8	64.2±5.7	76.4±0.4	**68.9**±0.6
	MSE	66.7±3.6	58.9±3.9	**65.0**±5.5	75.9±0.5	68.1±1.6
	MAE	61.0±5.9	48.5±9.5	59.8±15.8	75.6±0.9	55.0±3.3
GAT	CE	**73.4**±2.0	**64.6**±3.6	67.3±4.2	**76.8**±0.3	71.7±0.4
	HINGE	72.0±2.9	64.1±3.1	67.4±5.8	76.7±0.3	71.1±0.5
	MSE	73.4±3.2	59.7±2.2	**67.9**±2.4	75.7±0.6	**72.1**±0.6
	MAE	70.9±3.9	60.2±2.3	66.5±6.0	75.9±0.5	57.1±1.1

omit the explicit dependency of the predictions on the model parameters and illustrate the loss equations for a single data point.

Cross-entropy. By far, cross-entropy (CE) is the most commonly used loss function for classification problems. This loss is the negative logarithm of a categorical likelihood parameterized by the output of a softmax. Assume $\boldsymbol{p}$ is the softmax-transformed prediction of our model for a training sample whose true label has one-hot encoding $\boldsymbol{y}$. The categorical CE can be expressed as:

$$\mathcal{L}_{CE}(\boldsymbol{y}, \boldsymbol{p}) = -\sum_{i=1}^{C} y_i \log p_i. \tag{7}$$

Mean squared error. Albeit more commonly used for regression, the MSE can also be used for classification tasks. Adopting a one-hot representation $\boldsymbol{y}$ for the target label, the MSE loss can be written as:

$$\mathcal{L}_{MSE}(\boldsymbol{y}, \boldsymbol{p}) = \sum_{i=1}^{C} (y_i - p_i)^2. \tag{8}$$

Mean absolute error. Similarly to the MSE, the MAE is often employed in regression tasks. While the MSE is the L2 norm between predictions and targets, the MAE corresponds to the L1 norm:

$$\mathcal{L}_{MAE}(\boldsymbol{y}, \boldsymbol{p}) = \sum_{i=1}^{C} |y_i - p_i|. \tag{9}$$

Hinge-loss. Hinge loss functions play an important role in large-margin methods for binary classification. Here we consider the multiclass extension by Weston and Watkins [15]:

$$\mathcal{L}_{hinge}(\boldsymbol{y}, \boldsymbol{p}) = \sum_{i=1}^{C} \max(0, \gamma + p_i - p_{j^*}), \tag{10}$$

where $j^* = \arg\max_j y_j$ and γ is a hyper-parameter. Importantly, we follow Gosh *et al.* [16] and assume that, for all loss functions, the prediction $\boldsymbol{p}$ is the output of a softmax activation function.

III. IS ANY LOSS BETTER FOR NOISY LABELS?

In this section, we assess the impact of different loss functions under the noisy label regime. To simulate this setting, for each labeled example: i) we sample a value z from Uniform$(0, 1)$, and ii) if z is greater than a constant $p \in (0, 1)$, which governs the amount of noise, we switch the node label by a different one sampled uniformly. This regime is known as symmetric label noise and has been thoroughly studied outside graph domains [16], [17].

Datasets. We consider data from five popular benchmarks: Cora, Citeseer, Pubmed [23], DBLP [24] and Wiki-CS [25]. While the first four datasets represent classic citation networks, Wiki-CS comprises Wikipedia posts as nodes and hyper-links (edges) between these posts. All datasets have textual node features. We provide further details and dataset statistics in the Appendix.

Experimental Setup. We evaluate three representative GNNs in our experiments: GAT, GCN and SGC. Following guidelines from their original works, all models act on a 2-hop neighborhood in the graph. We train all models for 10^4 epochs using Adam [13] and repeated the experiments 10 times, using different seeds. To simulate different levels of label noise, we randomly re-assign the labels of a fraction of the training nodes. This type of noise is often referred to as symmetric, since the flipping probability does not depend on the original label nor the re-assigned one. We analyze the performance of each model using four loss functions: CE, MSE, MAE and the Weston-Watkins hinge-loss. We provide further implementation details in the Appendix.

Results. Table I presents the average test accuracy and standard deviation for $p = 0.3$. Notably, no loss clearly outperforms the others. For instance, most loss functions (CE, Hinge and MSE) are optimal for at least one method/dataset

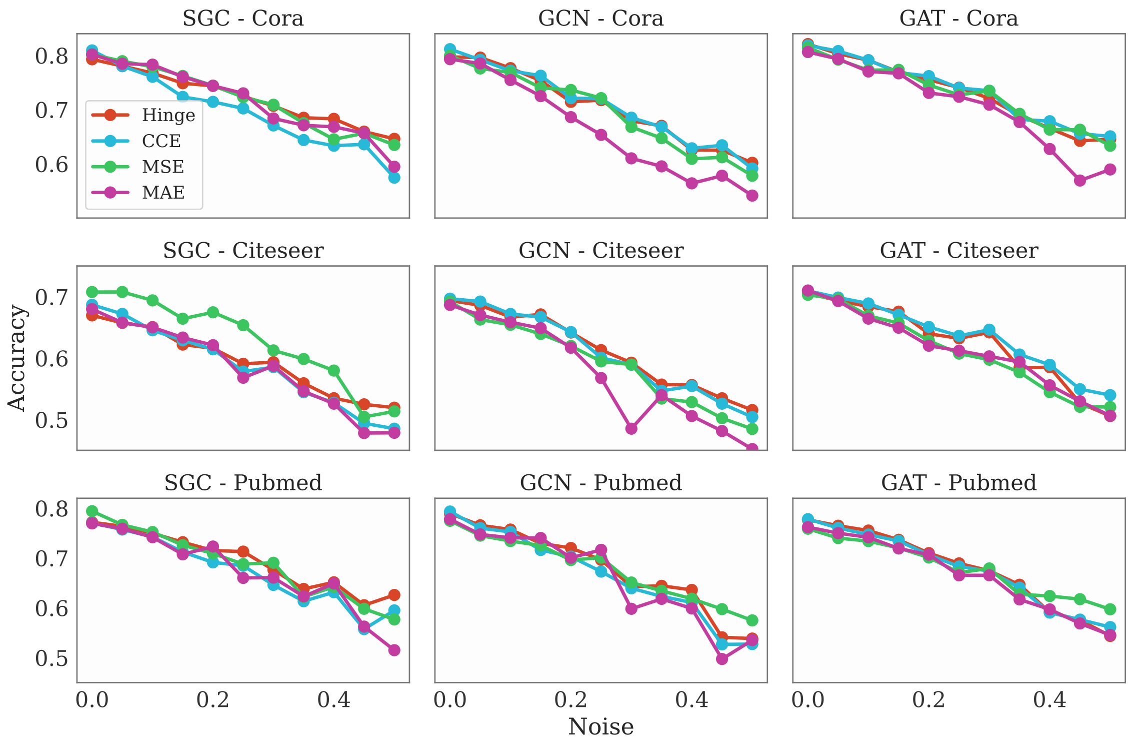

Fig. 1. Average test accuracy for SGC, GCN and GAT using different losses as a function of label noise (proportion of misslabeled nodes). Overall, there is no clear best loss function. As expected, the test accuracy drops as the noise increases.

pair. Additionally, Figure 1 corroborates these findings with results for different values of p.

IV. Is any loss better when labels are scarce?

In this section, we assess the impact of different loss functions when labeled data is scarce, a defining feature of semi-supervised node classification settings.

Again, we consider three GNNs: SGC, GCN and GAT; and four loss functions: MSE, MAE, CE, and Hinge loss. We train these models using different loss functions on Cora, Citeseer and Pubmed, for a varying number of labeled nodes per class in $\{5, 10, 20, 30, 60\}$.

Results. Figure 2 shows the average test accuracy for SGC, GCN and GAT trained with different loss functions and amounts of labeled data. Again, no loss is cearly better than the others. For instance, using only five labeled nodes, the Hinge loss achieves the highest performance for SGC on Core. However, the same method performs better on Pubmed when trained using MSE, given the same amount of data.

V. Related Works

Learning from noisy labels. While learning from noisy labels has been extensively studied on regular domains (e.g.,image classification), the topic has been less explored on graphs. Hoang *et al.* [26] propose using a modified loss function which takes into account an estimate of the (symmetric) noise distribution, also learned by the model. Yayong *et al.* [27] propose UnionNET-GCN , which leverages random walks to learn importance weights for the labeled nodes and to correct possibly corrupted labels.

Large-margin losses for GNNs. Wang *et al.* [28] employ large-margin losses functions to learn efficient initializations for GNNs in few-shot classification tasks, when only a small amount of data is available. Recently, Zhao *et al.* [29] used large-margin loss function to extract node representations which can be used for anomaly detection when paired with global patterns, learned using graph mining algorithms.

VI. Conclusions and future works

This work evaluates the impact of training GNNs using different loss functions when labels are noisy and when labeled data are scarce. Among the classical loss functions, we find there is no choice that clearly outperforms the others in any of these scenarios.

In particular, our results for noisy labels directly contrast with prior findings for non-graph data [16], [17], [18]. We believe this opens a venue to explore theoretical aspects of semi-supervised node classification that lead to this discrepancy, which we will explore in future works.

We also plan to assess the impact of losses on different graph tasks (e.g. link prediction) and other predictive aspects of GNNs, such as calibration.

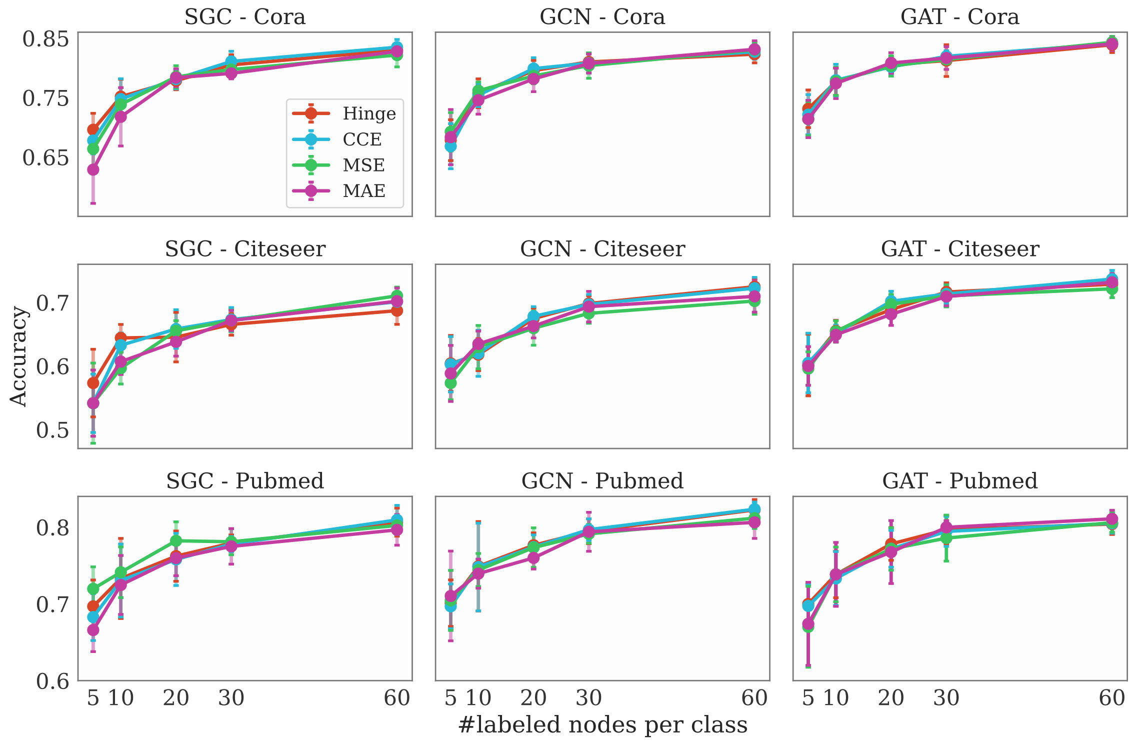

Fig. 2. Mean and standard deviation of the accuracy for SGC, GCN, and GAT. We compare different losses as a function of the number of labeled nodes per class. Overall, there is no significant difference between the loss functions.

References

[1] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, p. 61–80, 2009.

[2] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[3] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations (ICLR)*, 2014.

[4] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Neural Information Processing Systems (NeurIPS)*, 2016.

[5] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2019.

[6] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[7] A. H. Khasahmadi, K. Hassani, P. Moradi, L. Lee, and Q. Morris, "Memory-based graph networks," in *International Conference on Learning Representations (ICLR)*, 2020.

[8] H. Yuan and S. Ji, "StructPool: Structured graph pooling via random fields," in *International Conference on Learning Representations (ICLR)*, 2020.

[9] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, V. M. Tran, A. Chiappino-Pepe, A. H. Badran, I. W. Andrews, E. J. Chory, G. M. Church, E. D. Brown, T. S. Jaakkola, R. Barzilay, and J. J. Collins, "A deep learning approach to antibiotic discovery," *Cell*, vol. 180, no. 4, pp. 688 – 702, 2020.

[10] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *International Conference on Knowledge Discovery & Data Mining (KDD)*, 2018.

[11] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[12] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *International Conference on Machine Learning (ICML)*, 2020.

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

[14] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[15] J. Weston, C. Watkins *et al.*, "Support vector machines for multi-class pattern recognition." in *Esann*, vol. 99, 1999, pp. 219–224.

[16] A. Ghosh, H. Kumar, and P. S. Sastry, "Robust loss functions under label noise for deep neural networks," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17. AAAI Press, 2017, p. 1919–1925.

[17] H. Song, M. Kim, D. Park, and J. Lee, "Learning from noisy labels with deep neural networks: A survey," *arXiv preprint arXiv:2007.08199v2*, 2020.

[18] R. Manwani and P. Sastry, "Noise tolerance under risk minimization," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, p. 1146–1151, 2013.

[19] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *International Conference on Learning Representations*, 2017.

[20] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, “Simplifying graph convolutional networks,” in *International conference on machine learning*. PMLR, 2019, pp. 6861–6871.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph Attention Networks,” *International Conference on Learning Representations*, 2018.

[22] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs,” *IEEE Transactions on Signal Processing*, vol. 61, pp. 1644–1656, 2013.

[23] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, “Collective classification in network data,” *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[24] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, “Tri-party deep network representation,” in *IJCAI International Joint Conference on Artificial Intelligence*, 2016.

[25] P. Mernyei and C. Cangea, “Wiki-cs: A wikipedia-based benchmark for graph neural networks,” *arXiv preprint arXiv:2007.02901*, 2020.

[26] H. NT, C. J. Jin, and T. Murata, “Learning graph neural networks with noisy labels,” *CoRR*, vol. abs/1905.01591, 2019. [Online]. Available: http://arxiv.org/abs/1905.01591

[27] Y. Li, J. Yin, and L. Chen, “Unified robust training for graph neural networks against label noise,” *Arxiv e-print*, vol. abs/2103.03414, 2021.

[28] Y. Wang, X.-M. Wu, Q. Li, J. Gu, W. Xiang, L. Zhang, and V. O. Li, “Large margin meta-learning for few-shot classification,” in *Proc. 2nd Workshop Meta-Learn. NeurIPS*, 2018, pp. 1–8.

[29] T. Zhao, C. Deng, K. Yu, T. Jiang, D. Wang, and M. Jiang, “Gnn-based graph anomaly detection with graph anomaly loss,” in *International Conference on Knowledge Discovery & Data Mining (KDD)*, 2020.

[30] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *arXiv preprint arXiv:1511.07289*, 2015.

[31] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch Geometric,” in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

APPENDIX

A. Datasets

We used five datasets for node classification tasks, four of which are citation networks: Cora, Citeseer and Pubmed, DBLP, where nodes correspond to documents and the undirected edges are citations. Wiki-CS is a dataset based on Wikipedia in which nodes are Computer Science articles, edges are hyperlinks, and classes represent different branches of the field. All datasets have been divided into training, validation and testing as specified in Table II. For the noisy label setup on Cora, Citeseer, Pubmed and Wiki-CS, we use the public split training set. For DBLP, we use random splits. For the experiment with different numbers of nodes labeled per class, we use random splits. Table II shows summary statistics for each dataset.

B. Implementation details

For SGC, we apply a 2-hop neighborhood model. We train SGC for 500 epochs. Regarding the GAT model, we employ 2 layers. The first layer comprises 8 attentions heads, followed by exponential linear unity (ELU) [30] non-linearity. We train GAT models for 10000 epochs. Regarding GCNs, apply a 2-layer model, trained for 500 epochs. For all methods, we use the Adam optimizer with learning rate and weight decaying as specified in the Table III for each loss function. In addition, we use early stop with patience 100 epochs and all experiments were repeated 10 times to obtain mean and standard deviation results. We implement all models using the PyTorch Geometric Toolkit [31].

TABLE II
SUMMARY STATISTICS FOR THE DATASETS USED IN OUR EXPERIMENTS.

	#Nodes	#Edges	#Features	#Classes	#Train	#Val	#Test
Cora	2708	5429	1433	7	140	500	1000
Citeseer	3327	4732	3703	6	120	500	1000
Pubmed	19717	44338	500	3	60	500	1000
DBLP	17716	52867	1639	4	800	400	17516
Wiki-CS	11701	297110	300	10	580	1769	5847

C. Additional results

Figure 3 depicts the performance of the loss functions with noisy labels for the Wiki-CS and DBLP datasets. On the DBLP dataset, the loss functions obtains results similar to those from the cross entropy loss. On the Wiki-CS dataset, the MAE function obtains poor performance.

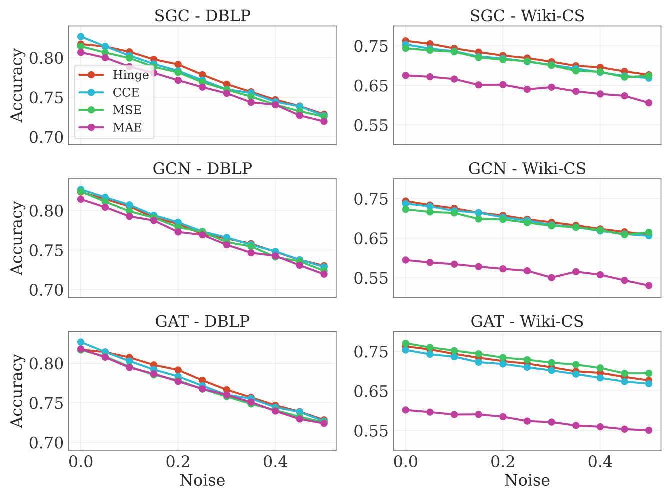

Fig. 3. Results (average accuracy) for SGC, GCN, and GAT using cross-entropy, hinge-loss, MSE, and MAE loss functions on the DBLP and Wiki-CS datasets. Except for MAE on Wiki-CS, all losses achieve very similar performance as we increase the amount of noisy labels.

D. Hyper-parameters

Table III shows the hyper-parameters for all models and datasets considered in this paper. We note that, for the experiments with variable number of nodes labeled per class, we use dropout as given in the original implementations of the GCN and GAT methods.

TABLE III
HYPER-PARAMETERS FOR EACH MODEL AND DATASET. LR STANDS FOR LEARNING RATE; WD STANDS FOR WEIGHT DECAY.

			GAT		
Model	**Loss**	**LR**	**WD**	**Heads**	**Hidden**
Cora CiteSeer Pubmed	Hinge	$5e-3$	$5e-4$	8	8
	CE	$5e-3$	$5e-4$	8	8
	MSE	$1e-2$	$5e-6$	8	8
	MAE	$1e-2$	$5e-6$	8	8
DBLP	Hinge	$5e-3$	$5e-4$	8	8
	CE	$5e-3$	$5e-4$	8	8
	MSE	$5e-3$	$5e-6$	8	8
	MAE	$5e-3$	$5e-6$	8	8
WikiCS	Hinge	$7e-3$	$5e-4$	5	14
	CE	$7e-3$	$5e-4$	5	14
	MSE	$5e-3$	$-$	5	14
	MAE	$5e-3$	$-$	5	14

			GCN	
Model	**Loss**	**LR**	**WD**	**Hidden**
Cora CiteSeer Pubmed	Hinge	$1e-2$	$5e-4$	16
	CE	$1e-2$	$5e-4$	16
	MSE	$1e-2$	$5e-6$	16
	MAE	$1e-2$	$5e-6$	16
DBLP	Hinge	$1e-2$	$5e-4$	16
	CE	$1e-2$	$5e-4$	16
	MSE	0.1	$5e-6$	16
	MAE	0.1	$5e-6$	16
WikiCS	Hinge	$2e-2$	$-$	33
	CE	$2e-2$	$-$	33
	MSE	$5e-3$	$-$	33
	MAE	$5e-3$	$-$	33

			SGC	
Model	**Loss**	**LR**	**WD**	**Hidden**
Cora CiteSeer Pubmed	Hinge	0.2	$5e-6$	-
	CE	0.2	$5e-6$	-
	MSE	0.1	-	-
	MAE	0.1	-	-
DBLP	Hinge	0.2	$5e-6$	-
	CE	0.2	$5e-6$	-
	MSE	0.3	-	-
	MAE	0.3	-	-
WikiCS	Hinge	0.2	$5e-6$	-
	CE	0.2	$5e-6$	-
	MSE	5e-3	-	-
	MAE	5e-3	-	-