G-Loss: Graph-Guided Fine-Tuning of Language Models

Proceedings Track Submission

Anonymous Author(s)
Anonymous Affiliation
Anonymous Email

Abstract

Traditional loss functions (cross-entropy, contrastive, triplet, and supervised contrastive) for fine-tuning pre-trained language models, such as BERT, operate in the local neighborhood, overlooking the holistic structure of semantic relationships. Thus, we propose *G-Loss*, a novel graph-guided loss function that uses semi-supervised label propagation and leverages the structural relationships in the embedding manifold using graphs. *G-Loss* constructs a document similarity graph to capture global semantic relationships, guiding the language model to learn more discriminative and robust embeddings. We evaluated *G-Loss* on five benchmark datasets: MR (sentiment), R8 and R52 (topic), Ohsumed (medical), and 20NG (news). *G-Loss* converges faster and produces a semantically coherent embedding space, consequently improving classification performance compared to the language models trained with traditional losses across diverse classification tasks.

1 Introduction

Language models ranging from BERT [1], RoBERTa[2], with millions of parameters to large language models like GPT [3] and LLaMA [4], etc., with parameters in billions, have transformed natural language processing (NLP). These models follow a *two-stage* approach: unsupervised pre-training on large unlabeled text corpora to learn general representations, followed by supervised fine-tuning using labeled data and a task-specific loss function. However, fine-tuning presents challenges, including the need for large amounts of labeled data and substantial computational resources. Additionally, traditional fine-tuning losses, such as cross-entropy loss, focus on minimizing prediction error, potentially overlooking the semantic structure within the data.

Over time, various fine-tuning strategies have emerged. For instance, BERT and RoBERTa employ cross-entropy loss with a classification head to a fully connected layer [1, 2], ensuring different classes remain separated in the embedding space. However, this approach overlooks the relationships between samples, focusing only on individual labels [5]. Pairwise or triplet-based approaches, such as in SBERT [6], address this by optimizing cosine similarity or triplet loss, aligning semantically similar document pairs while separating dissimilar ones in embedding space. However, these losses become computationally expensive, as their effectiveness depends on sampling many positive and negative pairs to form pairs/triplets. Moreover, a fundamental limitation persists: these losses optimize local relationships independently without enforcing global structural alignment. For instance, learning that A is close to B and distinct from C does not ensure appropriate positioning of B and C relative to each other unless pair (B, C) is explicitly optimized. This local optimization limits generalization to unseen data.

To overcome these limitations, we propose a shift from local pairwise optimization to global structural alignment by modeling semantic relationships across all pairs through a graph. This alignment is enforced not just for the immediate neighbors (local consistency), but for multi-hop relationships that capture the global structure of the graph. Specifically, we propose a *graph-based fine-tuning strategy* by introducing *G-Loss*. While *G-Loss* can be applied wherever encoder-based embeddings are available, we focus on NLP tasks with transformer-based encoders. *G-Loss* is a graph-driven loss function that integrates the semi-supervised Label Propagation Algorithm (LPA) [7] into language

Submitted to the Fourth Learning on Graphs Conference (LoG 2025, Proceedings Track). Do not distribute.

model (LM) fine-tuning by leveraging LPA; *G-Loss* diffuses label information from labeled nodes to unlabeled nodes through the graph structure. This enables the model to leverage both supervised labels and implicit relational information from the graph. This aligns with the manifold assumption in semi-supervised learning [8], which posits that proximity in feature space corresponds to label similarity.

Furthermore, LPA is parameter-free, ensuring computational efficiency and scalability across model sizes and datasets. Another key feature of our approach is the self-reinforcing system, where the graph structure and LM embeddings co-evolve during fine-tuning. Improved embeddings dynamically reshape the graph structure, which in turn guides further refinement of the embeddings, leading to better structural alignment.

Unlike prior work that either use LPA with static, full-dataset graphs for vision tasks [9] or constructs similarity graphs as soft targets for image classification [10], *G-Loss* offers key innovations. *G-Loss* dynamically constructs mini-batch graphs using evolving LM embeddings, enabling inductive learning with reduced memory and better generalization. Furthermore, *G-Loss* directly integrates LPA into the loss function, eliminating the need for separate pseudo-labeling/retraining, and streamlining training while preserving embedding consistency. To this end, we summarize the major contributions of this work as follows:

- We propose a fine-tuning approach that integrates a graph-based loss derived from dynamically evolving graphs and the semi-supervised Label Propagation Algorithm (LPA). Unlike conventional losses that optimize local relationships (e.g., pairwise and triplet losses), our method enforces global semantic consistency among documents. The co-evolution of the graph structure and embedding space across successive fine-tuning epochs yields robust and more discriminative representations.
- We implement and evaluate the proposed fine-tuning framework using `BERT`, `RoBERTa`, and `DistilBERT`, demonstrating its applicability across models of various sizes.
- We conduct experiments on five benchmark datasets, spanning from binary to complex multi-class (up to 52 classes), with varying class distribution and different classification tasks. Performance is reported using accuracy and macro F1-score metrics.
- We also compare the proposed fine-tuning framework with other traditional loss functions, including supervised contrastive, cosine similarity, triplet, and cross-entropy losses.

The rest of the paper is organized as follows: Section 2 reviews related literature. Section 3 describes the fine-tuning process with *G-Loss*. Section 4 describes the benchmark datasets, language model variants considered, comparable loss functions, downstream tasks, and the evaluation criteria. Section 5 discusses the results, followed by Section 7, which concludes the paper with a list of future work.

2 Related work

Loss functions play a crucial role in fine-tuning language models (LMs) to enhance the models' representation capabilities. Fine-tuning `BERT` [1], `RoBERTa` [2], and `ALBERT` [11] for a downstream task typically involves appending a classification head and optimizing the cross-entropy (CE) loss. While effective in classification, fine-tuning with cross-entropy loss treats training instances independently and ignores structural consistency. Similarity-based learning methods address this: Sentence-BERT (`SBERT`) [6] uses a Siamese network architecture with triplet and cosine similarity loss for fine-grained pairwise comparisons.

`SimCSE` [12] proposed a contrastive learning via dropout augmentation, aligning the embedding space by contrasting positive/negative pairs. `SimCSE` aligns the embedding space, thereby promoting more effective sentence representations. `SimCSE++` [13] further refined this approach by reducing negative pair noise and introducing a dimension-wise contrastive objective to prevent feature collapse. However, these methods remain locally optimized, focusing only on relationships between individual samples but failing to capture global semantic structures [14]. Gunel et al. [15] introduces supervised contrastive learning (SCL) objective to complement cross-entropy loss for fine-tuning language models.

Graph-based models, such as graph neural networks (GNN), incorporate structural dependencies by leveraging relational information across datasets [16]. Graph Convolutional Networks (GCN) [16]

pioneered contextual aggregation via feature propagation. `TextGCN` [17] extended this to document classification with a heterogeneous word-document graph, and `TensorGCN` [18] integrates syntactic and sequential dependencies across words in text.

Hybrid models, such as `GNN-LM`, have shown promise. `BertGCN` integrates BERT embeddings into `GCN`, enhancing text classification performance [19]. Other approaches include Cascading models: `LM` generates embeddings, which are then processed by a `GNN` for classification [20–22], Co-training methods: `LMs` and `GNNs` train jointly with a shared objective [22–24], and Frozen LM strategies: Frozen `LM` while only the `GNN` is trained [25, 26].

Despite success, the above-mentioned graph-based methods suffer from two key limitations: (1) static graph construction (fixed, non-adaptive graphs) and (2) high computational overhead from joint `GNN-LM` training. For instance, `BertGCN`'s static, full-dataset graph incurs high memory usage and computational overhead, as well as prolonged training, due to the use of a memory bank and a small learning rate.

We introduce *G-Loss*, which addresses these limitations through dynamic graph construction. *G-Loss* enables models to learn and refine semantic structures during fine-tuning, unlike prior methods that either (1) ignore global structural dependencies (cross-entropy, triplet loss, `SimCSE`), or (2) require expensive, static graphs (`BertGCN`, `TextGCN`). *G-Loss* bridges this gap by leveraging dynamic graph adaptation alongside efficient label propagation, thereby facilitating the direct integration of graph-structured learning into language model fine-tuning.

3 Proposed fine-tuning framework

3.1 Task description

For a multi-class classification problem, we are given a document set $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$, belonging to $\mathcal{C}$ distinct classes. We partition $\mathcal{D}$ into three disjoint sets: training, validation, and testing. The goal is to fine-tune a pre-trained language model on the training set by optimizing a task-specific loss function, and evaluate the fine-tuned language models' performance on the testing set by classifying each test document into one of the $\mathcal{C}$ classes.

During fine-tuning, we select a minibatch of size B from the training set $\mathcal{V}$ and represent it as a graph $\mathcal{G} = (\mathcal{V}_k, \mathcal{E}_k, \mathcal{X}_k)$. Here, nodes $\mathcal{V}_k$ represent documents in current minibatch, edges $\mathcal{E}_k$ represent semantic relationships between documents, and $\mathcal{X}_k \in \mathbb{R}^{B\times d}$ represents d-dimensional document embeddings from language model.

3.2 G-Loss: Graph-driven loss computation

Given the training document set $\mathcal{T} \in \mathcal{D}$ and corresponding labels $\mathcal{Y}$, fine-tuning is performed iteratively in m minibatches (b_1, b_2....b_m). For each minibatch b_k, the process follows these steps: we encode each document into dense representations using a language model, $\Phi(.)$. Then, we construct a similarity graph where nodes represent documents and edges are weighted using a similarity measure to capture the semantic relationships between them. Next, we hide a subset of labels based on a masking ratio gamma (γ). LPA [7] infers these hidden labels while keeping known labels fixed. The loss computed from the discrepancy between inferred and true labels of masked nodes gives us the value of *G-Loss*. Final loss is computed using a weighted average of *G-Loss* with the CE loss and minimized via backpropagation. This iterative process refines both the embeddings and the graph structure to align with the global semantic relationships across the document set. Figure 1 presents the *G-Loss* based fine-tuning framework for one minibatch. Each step is explained in detail as follows:

(Step 1) Embedding extraction . Text in each document in minibatch b_k, $k \in (1, 2, ...m)$ is encoded and mapped by language model $\Phi(.)$ into a d-dimensional semantic space, capturing semantic information through contextual embeddings. The resulting embedding matrix $\mathbf{X}_k \in \mathbb{R}^{B\times d}$ is represented as:

$$\mathbf{X}_k = \Phi(\text{Text in } b_k). \tag{1}$$

where $\Phi(.)$ denotes a language model. These embeddings act as node features in the subsequent graph construction step. The quality of embeddings directly impacts the effectiveness of label propagation and the performance of downstream classification.

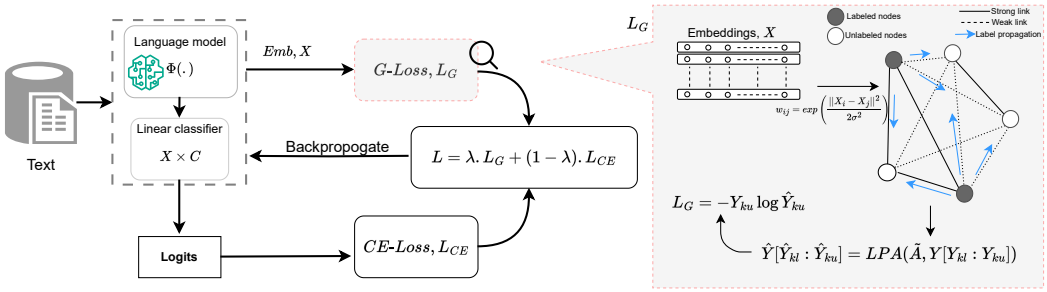


Figure 1: The language model $\Phi(.)$ generates embeddings from input text, which is feed into both a linear classifier (producing logits) and a similarity-based document graph where nodes represent documents and weight w_{ij} represents similarity between documents. Labels for few nodes (Y_{ku}) are hidden and inferred using label propagation algorithm (LPA) producing ($\hat{Y}_{ku}$). This gives graph-based loss (L_G), which is combined with cross-entropy loss L_{CE} using weighting factor λ, and composite loss backpropagates to update both language model parameters and linear classifier weights.

(Step 2) Graph construction. Next step is to construct a fully connected weighted graph $\mathcal{G} = (\mathcal{V}_k, \mathcal{E}_k, \mathcal{X}_k)$ for current minibatch b_k. The weight of an edge between two nodes i and j is computed using a Gaussian kernel: $\mathbf{W}_{ij} = \exp\left(-\frac{\|\mathbf{X}_i - \mathbf{X}_j\|^2}{2\sigma^2}\right), \quad \mathbf{W}_{ii} = 0$, where X_i represents embeddings of document i, and σ controls the scale of neighborhood sensitivity in kernel space. We investigate two approaches for selecting σ and propose two variants of *G-Loss* function. First, *G-Loss-SQRT* derives σ analytically, as detailed in section D of appendix, by using double partial differentiation of the Gaussian kernel function, which yields $\sigma = \sqrt{d_1/3}$ where d_1 is the median Euclidean distance across all data points. Second, *G-Loss-O* determines optimal σ through systematic hyperparameter optimization. *G-Loss-SQRT* offers a computationally efficient alternative for environments with limited resources, bypassing resource-intensive hyperparameter tuning.

The Gaussian kernel ensures that highly similar documents are strongly connected while dissimilar ones have weaker connections. Further, we normalize the adjacency matrix using symmetric normalization to stabilize information propagation in the graph: $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, where $\mathbf{D}_{ii} = \sum_{j=1}^{B} \mathbf{W}_{ij}$ represents the degree matrix. This ensures that label propagation distributes information effectively without amplifying numerical instabilities.

(Step 3) Label propagation. We use a hyperparameter gamma (γ) to randomly split the minibatch node set $\mathcal{V}_k$ into two disjoint subsets to enable semi-supervised learning:

- $\mathcal{V}_{kl}$ (Labeled subset): A γ fraction of nodes with true labels (Y_{kl}) that guide propagation.
- $\mathcal{V}_{ku}$ (Evaluation subset): The remaining $(1-\gamma)$ fraction whose labels (Y_{ku}) are masked and must be inferred.

We compute a column-stochastic transition matrix $\mathbf{T}$:

$$\mathbf{T}_{ij} = P(j \to i) = \frac{\tilde{A}_{ij}}{\sum_{m=0}^{B} \tilde{A}_{mj}} \tag{2}$$

Using the closed-from label propagation solution of LPA [7], predicted labels for the evaluation set are computed as:

$$\hat{Y}_{ku} = (I - T_{uu})^{-1}.T_{ul}Y_{kl} \tag{3}$$

where I is identity matrix, T_{uu} is transition sub-matrix of evaluation nodes, T_{ul} captures transitions from labeled to evaluation nodes.

(Step 4) Loss computation. To compute the loss from graph, we compute the discrepancy between inferred labels ($\hat{\mathbf{Y}}_{ku}$) and true labels, ($\mathbf{Y}_{ku}$) on the evaluation subset of size $B_e = (1-\gamma) * B$:

$$\mathcal{L}_{\mathcal{G}} = -\frac{1}{B_e} \sum_{j=1}^{B_e} \sum_{c=1}^{C} y_{ku,j}^{c} \log\left(\hat{y}_{ku,j}^{c}\right), \tag{4}$$

where: for the class c, $y^c_{ku,j}$ and $\hat{y}^c_{ku,j}$ are the ground-truth and predicted labels of j-th evaluation node, B_e is the number of samples in evaluation set of k^{th} minibatch. C is the number of classes.

(Step 5) Loss Function Integration: The final loss function combines the standard cross-entropy (CE) loss with the graph-based loss via weighted average. A scalar hyperparameter λ controls the trade-off between the two loss components:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{\mathcal{G}} + (1 - \lambda) \cdot \mathcal{L}_{\mathcal{CE}} \tag{5}$$

where $\mathcal{L}_{\mathcal{CE}}$ is the cross-entropy loss over all data points in the current training batch, $\mathcal{L}_{\mathcal{G}}$ is the graph-based propagation loss over the evaluation subset, $\lambda \in [0, 1]$ controls the influence of both losses.

The CE loss is defined as:

$$\mathcal{L}_{\mathcal{CE}} = -\frac{1}{B} \sum_{i=1}^{B} \sum_{c=1}^{C} y_{i,c} \log \hat{y}_{i,c} \tag{6}$$

Here, B denotes the batch size, C represents the number of classes, $y_{i,c}$ is the ground truth label for sample i and class c, and $\hat{y}_{i,c}$ is the corresponding predicted probability from the linear classifier.

(Step 6) Model optimization via backpropagation. Both the language model parameters and the linear classifier weights are jointly optimized via gradient descent backpropagation, using the composite loss defined in equation 5. loss signal $\mathcal{L}$:

$$\mathbf{X}^{'}_{k} \leftarrow \mathbf{X}_{k} - \eta \nabla \mathcal{L}, \tag{7}$$

Where η represents the learning rate of the model. This step ensures that the embeddings evolve in a manner that enhances the effectiveness of label propagation.

Dynamic graph updating. Since embeddings continuously improve throughout fine-tuning, the graph structure must be updated accordingly. After each backpropagation step, the adjacency matrix $\mathbf{A}$ is recomputed using the newly refined embeddings:

$$\mathbf{W}^{'}_{ij} \leftarrow \exp\left(-\frac{\|\mathbf{X}^{'}_{i} - \mathbf{X}^{'}_{j}\|^2}{2\sigma^2}\right). \tag{8}$$

This iterative adaptation ensures that similarity-based relationships remain aligned with the evolving representation space, leading to more accurate label propagation and improved classification performance. A detailed algorithm of approach is provided in section A of the appendix.

4 Experimentation setup

4.1 Datasets and downstream tasks

We experimented on five diverse, widely used English language benchmark datasets for text classification. *Movie Review (MR)* [27]: binary sentiment classification, where each sample is labeled as either positive or negative. *R8* and *R52* [28]: hierarchical news categorization datasets derived from Reuters-21578, with 8 and 52 topic categories, respectively. *Ohsumed* [29]: medical text classification dataset comprising MEDLINE abstracts, categorized into 23 medical subject headings (MeSH). Lastly, *20 Newsgroups (20NG)* [30]: documents classified into 20 distinct newsgroups representing various discussion topics. For a fair comparison, we adopted BertGCN's [31] data splits, ensuring consistent training and test partitions. Table 1 provides the train/validation/test split details for each dataset.

4.2 Language models and traditional losses

Language Models. While *G-Loss* is compatible with any encoder-based transformer, we evaluate it using three models of varying size and complexity: `BERT-base-uncased` (12 layers, 768 dim, 110M parameters) [1], `RoBERTa-large` [2] (24 layers, 1024 dim, 356M parameters), and `DistilBERT-base-uncased` (6 layers, 768 dim, 66M parameters) [32].

Table 1: Dataset statistics used for evaluation. The Train, Validation, and Test column represents the number of documents in each split.

Dataset	# Docs	Train	Validation	Test	# Classes
MR	10662	6397	711	3554	2
R8	7674	4936	549	2189	8
R52	9100	5865	667	2568	52
20NG	18846	10182	1132	7532	20
Ohsumed	7400	3021	336	4043	23

Traditional loss functions. We evaluate our proposed *G-Loss* under two configurations: integrated and standalone. In **integrated approach**, we implement the framework shown in Figure 1, where the hybrid loss jointly optimizes both the language model parameters and linear classifier weights. The integrated approach is in sync with approach followed by prior works such as BERT, RoBERTa, BertGCN [19], Gunel et al. [15] etc. In **standalone** approach, only the language model weights are updated using the computed loss.

- **Integrated:** We compare *G-Loss* augmented with cross-entropy loss against standard cross-entropy alone, and hybrid supervised contrastive and cross-entropy loss (SCL+CE) [15].
- **Standalone:** *G-Loss* is evaluated against triplet loss [33], supervised contrastive loss (SCL) [5], and cosine-similarity loss.

Mathematical formulations for all traditional losses are provided in Appendix B.

4.3 Fine-tuning convergence & hyperparameter tuning

Evaluation strategies and early stopping. Following our dual evaluation framework, in integrated approach, we implement a classification-based monitoring system that tracks macro F1-score on the validation set throughout fine-tuning and applies early stopping when performance plateaus for a predetermined patience window. Performance evaluation utilizes a linear classifier with dimensions $E \times C$ (where E represents embedding size and C denotes the number of classes) attached to the final layer of language model.

In standalone approach, we fine-tune the language model using only a single loss objective, with training convergence monitored via macro-silhouette score [34] on true labels of the validation set (the mathematical formula is provided in appendix C). Unlike the standard silhouette score, macro-silhouette takes into account the imbalance in the dataset and gives a balanced measure. After fine-tuning, a linear classifier is trained independently on the updated embeddings. Final predictions on the test set are then generated using this downstream classifier. This strategy eliminates the need for a supervised classification head to be trained alongside the language model (as in BERT, RoBERTa, and BertGCN) during fine-tuning, thereby reducing the number of parameters and computational overhead.

Hyperparameter selection & final classifier. Hyperparameter optimization is conducted using `Optuna`. For *G-Loss*, we tune: Gaussian kernel width $\sigma \in [0.1, 10]$, learning rate $\eta \in \{1e-05, 2e-05, 3e-05, 4e-05, 5e-05\}$, label hiding ratio $\gamma \in [0.1, 0.9]$, and loss weighting factor $\lambda \in [0.1, 0.9]$. Traditional losses undergo similar tuning: temperature $\tau \in [0.01, 1]$ and learning rate η for Supervised Contrastive Loss (SCL), and learning rate η for cosine-similarity and triplet losses. We add details about hyperparameter selection in section G of the appendix. All the experiments in this study were conducted on a single NVIDIA A100 80GB GPU.

Evaluation metrics and protocol. We report the test accuracy and macro F1-score as primary metrics, ensuring comparability with prior work such as TextGCN, TensorGCN, BertGCN. Macro F1-score provides a more reliable evaluation due to class imbalance in some datasets (e.g., R52, Ohsumed). In addition to classification performance, we measure the computational efficiency of *G-Loss* by tracking the fine-tuning time till convergence and the average time taken per epoch for *G-Loss* and other loss functions in table 3.

5 Results and discussion

5.1 Classification performance analysis

We experimentally compare the proposed *G-Loss* with a strong baseline of CE loss and SCL+CE loss functions (Section 4.2). For robust assessment, we used three distinct language model architectures (section 4.2).

Table 2: Accuracy(%)/macro F1-score (%) obtained while fine-tuning different language model variants across multiple datasets. The values highlighted in gray and yellow denote the best and second-best performance, respectively. We report the mean and variance of three different seeds.

Model	Loss	MR	R8	R52	20NG	Ohsumed
BERT-base -uncased	CE	85.64±0.81 / 85.64±0.81	97.57±0.16 / 93.90±0.84	96.29±0.15 / 84.42±0.60	83.98±0.12 / 83.49±0.18	71.09±0.31 / 63.40±0.83
	GLoss-SQRT + CE	86.17±0.35 / 86.17±0.35	97.67±0.19 / 94.34±0.66	96.06±0.18 / 83.78±0.48	84.01±0.56 / 83.52±0.62	70.55±0.18 / 62.49±0.73
	GLoss-O + CE	86.63±0.12 / 86.61±0.12	97.91±0.20 / 94.53±0.76	96.37±0.40 / 84.55±0.38	84.71±0.24 / 84.17±0.26	71.25±0.59 / 63.80±0.82
	SCL + CE	85.97±0.50 / 85.95±0.50	97.88±0.17 / 93.96±0.57	96.18±0.14 / 83.75±0.77	84.39±0.10 / 83.99±0.13	71.28±0.17 / 63.64±0.20
RoBERTa -large	CE	90.04 / 90.04	97.85 / 94.15	96.04 / 83.29	84.68 / 84.04	73.77 / 65.88
	G-Loss-SQRT + CE	90.53±0.51 / 90.53±0.51	97.69 / 93.78	96.41 / 84.43	84.26 / 84.01	74.58 / 66.75
	G-Loss-O + CE	90.87±0.62 / 90.87±0.62	97.49±0.05 / 93.04±0.12	96.65 / 84.82	85.19±0.31 / 84.23±0.73	75.35±0.42 / 68.58±1.02
	SCL + CE [15]	90.82/ 90.81	97.81 / 93.80	96.22 / 83.19	84.53 / 83.97	75.76 / 68.53
DistilBERT -base	CE	84.85±0.23 / 84.85±0.23	97.48±0.11 / 93.02±0.48	95.78±0.04 / 83.64±0.69	83.15±0.61 / 82.69±0.56	69.50±0.34 / 61.06±0.29
	GLoss-SQRT + CE	85.14±0.33 / 85.14±0.34	97.46±0.17 / 93.62±0.69	96.13±0.34 / 83.93±0.35	83.61±0.23 / 83.16±0.20	69.93±0.38 / 61.39±0.21
	GLoss-O + CE	85.10±0.45 / 85.10±0.45	97.71±0.12 / 93.61±0.22	96.07±0.04 / 84.44±0.43	83.64±0.28 / 83.23±0.27	70.16±0.21 / 62.41±0.34
	SCL + CE	84.16±0.46 / 84.16±0.46	97.65±0.25 / 93.79±0.43	96.17±0.10 / 83.82±1.62	83.49±0.20 / 83.18±0.12	70.02±0.35 / 60.56±0.80

Table 2 shows classification accuracy and macro F1-score across all datasets and language model variants. *G-Loss* variants closely match and marginally surpass other loss baselines regardless of the underlying language model, with improvements of $0.03\% - 1.06\%$ in accuracy and $0.05\% - 1.04\%$ in macro F1-score over the second-best-performing loss, across all datasets. Notably, *G-Loss* variants generalize across different language model families, indicating its ability to enhance a variety of transformer backbones. While SCL+CE remains competitive on some datasets, it underperforms on small models such as DistilBERT. These results confirm the effectiveness of our graph-based loss function in fine-tuning language models for classification, demonstrating consistent improvements over baselines.

Table 3 presents the standalone evaluation of *G-Loss* against traditional loss functions (detailed in Section 4.2). For comparison, we considered the best-performing *G-Loss-O* variant. *G-Loss-O* consistently achieves competitive or superior performance relative to traditional objectives. In particular, it delivers the highest Macro F1 on challenging datasets, such as Ohsumed (62.52%), and attains the highest silhouette scores across most benchmarks, indicating better structural alignment of embeddings. This demonstrates that beyond improving predictive accuracy, *G-Loss-O* produces semantically coherent embedding spaces. Notably, *G-Loss-O* converges in fewer epochs (e.g., 23 on R8) compared to alternative loss functions, highlighting its superior training efficiency. The average per-epoch training time and total time to convergence show that *G-Loss* maintains comparable per-epoch computational cost to existing baselines while achieving faster overall convergence.

We also provide a timing breakdown for per epoch time across *G-Loss* and other baselines in Figure 5 of the appendix. *G-Loss* records a total of 9.2 seconds for forward pass (BERT forward: 8.95 sec, graph construction: 0.18 sec, LPA operations: 0.07 sec) in one epoch, compared to Supervised Contrastive Loss (10.41 sec), Triplet Loss (9.15 sec), and Cosine Similarity Loss (9.09 sec). This confirms that the additional steps involved in graph construction, normalization, and LPA operations introduce negligible computational overhead during training.

6 Comparison with SOTA models

Table 4 provides a comprehensive comparison of our *G-Loss* fine-tuned language models against several state-of-the-art baselines. The baselines include three major paradigms in text classification: (i) graph-based methods (TextGCN [17], TensorGCN [18]), (ii) transformer-based language models (BERT (base-uncased), RoBERTa (large) language model, reported by Lin et al. [19]), and (iii) hybrid models that integrate language models with graph classification (Bert-GCN and RoBERTa-GCN versions of BertGCN [19]). All baseline results are taken from their respective published scores on the benchmark datasets used in this study. To ensure fairness across diverse approaches, accuracy is reported as the primary evaluation metric.

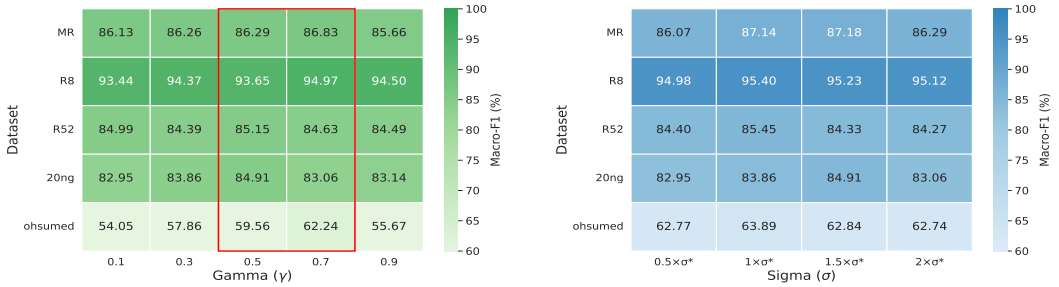
Table 3: Comparison of *G-Loss-O* with traditional losses using `BERT-base-uncased`. *G-Loss-O* achieves superior or competitive accuracy and F1, produces well-separated embeddings (higher silhouette scores), and converges in fewer epochs, demonstrating both effectiveness and efficiency. The highlighted colors are the same as in Table 2

Dataset	Loss	Accuracy (%)	Macro F1 (%)	Test macro silhouette score	Total train time (sec)	Avg. time per epoch (sec)	Early stopping epoch
MR	SCL	86.36	86.36	0.5725	1258.41	33.04	35
	Triplet	86.66	86.66	0.5331	941.55	27.50	29
	Cos-sim	86.14	86.14	0.4479	654.56	27.70	20
	GLoss-O	86.63	86.62	0.5507	810.85	27.77	25
R8	SCL	97.85	93.64	0.4869	1088.58	25.56	38
	Triplet	97.69	93.96	0.6581	727.85	21.19	30
	Cos-sim	97.94	94.10	0.7383	4960.64	21.65	200
	GLoss-O	97.99	94.25	0.7870	572.29	21.77	23
R52	SCL	96.10	81.24	0.4561	1170.50	30.41	34
	Triplet	96.53	82.54	0.4662	5496.14	25.05	189
	Cos-sim	95.94	80.90	0.3972	1925.82	25.79	64
	GLoss-O	96.24	81.99	0.4824	1095.93	25.87	39
Ohsumed	SCL	69.85	57.12	0.1373	691.03	15.70	39
	Triplet	68.56	57.86	0.1222	1576.44	13.09	105
	Cos-sim	70.49	59.65	0.1393	3064.31	13.42	200
	GLoss-O	70.67	62.52	0.1468	633.50	13.50	41
20NG	SCL	84.93	84.47	0.5507	11671.10	52.76	200
	Triplet	84.46	83.91	0.4538	8516.99	43.46	171
	Cos-sim	84.20	83.72	0.5524	10155.44	45.10	200
	GLoss-O	85.02	84.78	0.5849	8596.50	46.10	180

Table 4: Performance comparison (accuacy in %) with SOTA across benchmark datasets. G-Loss consistently improves over graph-based and BERT, while achieving competitive results with BertGCN, while avoiding full-graph overhead through dynamic mini-batch graph construction.

Model	MR	R8	R52	Ohsumed	20NG
TextGCN	76.74	97.07	93.56	68.36	86.34
TensorGCN	77.91	98.04	95.05	70.11	87.74
BERT-base	85.30	97.80	96.40	70.50	85.70
RoBERTa-large	89.40	97.80	96.20	70.70	83.80
Bert-GCN	86.00	98.10	96.60	72.80	89.30
RoBerta-GCN	89.70	98.20	96.10	72.80	89.50
G-Loss + BERT-base	87.14	98.04	96.48	71.48	85.13
G-Loss + RoBERTa-large	90.82	98.18	96.65	75.76	85.33

G-Loss delivers substantial gains over both standard pre-trained models and graph-based baselines. When combined with BERT-base, *G-Loss* consistently outperforms vanilla BERT across all datasets, achieving up to (+1.84) improvement on MR and (+0.98) on Ohsumed, highlighting the benefits of graph-driven structural supervision. Notably, *G-Loss* paired with RoBERTa-large attains new state-of-the-art results on MR (90.82), R52 (96.65), and Ohsumed (75.76), while remaining competitive on the other two. In contrast to BertGCN and RoBERTa-GCN, which require full-graph construction and incur significant memory costs, *G-Loss* achieves similar or superior accuracy with a lightweight, mini-batch dynamic graph, ensuring scalability and inductive generalization. Additionally, BertGCN's simultaneous co-training of BERT and GCN part substantially increases computational resource requirements. These results demonstrate that *G-Loss* not only strengthens language model fine-tuning but also bridges the gap between graph-based and transformer-based paradigms in a more efficient manner.



(a) Label hiding ratio γ vs Macro-F1 **(b)** Sigma parameter σ vs Macro-F1

Figure 2: Hyperparameter sensitivity analysis of *G-Loss* on `BERT-base-uncased`: (a) label hiding ratio γ and (b) Gaussian similarity parameter σ.

6.1 Ablation study results

We conduct an ablation study to explore the effectiveness of our proposed graph-based loss function, utilizing the `Bert-base-uncased` language model due to its lightweight architecture and reduced computational overhead.

Effect of label hiding ratio gamma (γ) on *G-Loss* performance. The label hiding ratio, denoted by gamma $(\gamma) \in [0, 1]$, plays an important role in *G-Loss* function, by maintaining a proportion of labels masked and unmasked during the LPA iterations. Specifically, a proportion of $(1 - \gamma)$ labels are hidden during fine-tuning, and LPA is applied to predict those hidden labels. Figure 2a presents an ablation study on γ, revealing that intermediate values - particularly in the range $\gamma \sim \{0.5 - 0.7\}$ - consistently lead to better performance across all the datasets. This suggests that revealing approximately $50 - 70\%$ of the labels encourages the model to generalize effectively by maintaining a balance: enough labeled nodes for stable propagation and enough unlabeled nodes to guide meaningful learning.

Impact of sigma in Gaussian similarity on performance of *G-Loss*. The parameter σ in the Gaussian similarity function determines graph connectivity and *G-Loss* performance. Small σ values create sparse, localized neighborhoods that preserve fine-grained distinctions, while larger values generate denser graphs that may blur class boundaries. Figure 2b shows performance across σ multipliers around the `Optuna`-optimized value (σ^*). The results reveal dataset-dependent sensitivity: MR and R8 datasets demonstrate remarkable stability, with performance varying by less than 1% across all σ values. Conversely, R52 shows a clear optimum at σ^* (85.45%), with notable degradation at both extremes (84.40% at $0.5\times$ and 84.27% at $2\times$). The 20NG and Ohsumed datasets exhibit intermediate sensitivity, with 20NG peaking at $1.5 \times \sigma^*$ (84.91%) rather than the optimal point. This analysis reveals two key insights: (1) G-Loss maintains robust performance within a reasonable σ range for most datasets, validating its practical applicability, and (2) the optimal σ balances local precision with global connectivity, confirming the importance of proper hyperparameter tuning for maximizing G-Loss effectiveness.

Additionally, Figure 6 in the appendix presents an ablation study on the weighting factor λ in *G-Loss-O*. We also plot t-SNE visualizations of learned embeddings across different loss functions on the MR and R8 datasets in section F of the appendix.

7 Conclusion

We proposed a novel graph-guided loss, *G-Loss* that shifts language model fine-tuning from the local pairwise optimization to global structure alignment. *G-Loss* utilizes a dynamically constructed semantic graph and semi-supervised LPA to capture global document similarity. Experiments on five benchmark datasets and three transformer architectures shows *G-Loss's* consistent effectiveness in achieving performance improvements compared to traditional losses. The dynamic mini-batch graph construction ensures computational efficiency and scalability. *G-Loss* highlights the potential of integrating graphs into language model fine-tuning. Future research directions include: (1) Scaling

G-Loss to large-language models (e.g., GPT, LLaMa), (2) evaluating on larger, non-textual complex datasets, and (3) extending to multi-label and multi-modal tasks.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, 2018. 1, 2, 5

[2] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, 2019. 1, 2, 5

[3] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018. 1

[4] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. 1

[5] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *CoRR*, 2020. 1, 6, 12

[6] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, 2019. 1, 2

[7] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002. 1, 3, 4, 15

[8] Yassine Ouali, Céline Hudelot, and Myriam Tami. An overview of deep semi-supervised learning. *CoRR*, 2020. 2

[9] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. *CoRR*, abs/1904.04717, 2019. 2

[10] Vlad Sobal, Mark Ibrahim, Randall Balestriero, Vivien Cabannes, Diane Bouchacourt, Pietro Astolfi, Kyunghyun Cho, and Yann LeCun. $\mathbb{X}$-sample contrastive loss: Improving contrastive learning with sample similarity graphs, 2024. 2

[11] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, 2019. 2

[12] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *CoRR*, abs/2104.08821, 2021. 2

[13] Jiahao Xu, Wei Shao, Lihui Chen, and Lemao Liu. Simcse++: Improving contrastive learning for sentence embeddings from two perspectives. In *Proceedings of the 2023 Conference on EMNLP*, 2023. 2

[14] Zhuoning Yuan, Yuexin Wu, Zi-Hao Qiu, Xianzhi Du, Lijun Zhang, Denny Zhou, and Tianbao Yang. Provable stochastic optimization for global contrastive learning: Small batch does not harm performance. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *ICML*, Proceedings of Machine Learning Research, 2022. 2

[15] Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. Supervised contrastive learning for pre-trained language model fine-tuning. *CoRR*, abs/2011.01403, 2020. 2, 6, 7

[16] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, 2016. 2

[17] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. *CoRR*, 2018. 3, 7

[18] Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. Tensor graph convolutional networks for text classification, 2020. 3, 7

[19] Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. Bertgcn: Transductive text classification by combining GCN and BERT. *CoRR*, 2021. 3, 6, 7

[20] Zhibin Lu, Pan Du, and Jian-Yun Nie. VGCN-BERT: augmenting BERT with graph embedding for text classification. *CoRR*, 2020. 3

[21] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning, 2024.

[22] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Guangzhong Sun, and Xing Xie. Graphformers: Gnn-nested language models for linked text representation. *CoRR*, 2021. 3

[23] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference, 2023.

[24] Rui Xue, Xipeng Shen, Ruozhou Yu, and Xiaorui Liu. Efficient end-to-end language model fine-tuning on graphs, 2024. 3

[25] Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. Efficient tuning and inference for large language models on textual graphs. In Kate Larson, editor, *Proceedings of the Thirty-Third IJCAI-24*, 2024. 3

[26] Qi Zhu, Da Zheng, Xiang Song, Shichang Zhang, Bowen Jin, Yizhou Sun, and George Karypis. Parameter-efficient tuning large language models for graph representation learning, 2024. 3

[27] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd ACL*, 2004. 5

[28] David D. Lewis. Reuters-21578 text categorization test collection. 1997. Distribution 1.0. 5

[29] William Hersh, Chris Buckley, T. J. Leone, and David Hickam. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994. 5

[30] Ken Lang. Newsweeder: Learning to filter netnews, 1995. 5

[31] Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. Github: Bertgcn-transductive text classification by combining gcn and bert, 2021. URL https://github.com/ZeroRin/BertGCN. 5

[32] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. 5

[33] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *CoRR*, 2017. 6, 12

[34] John Pavlopoulos, Georgios Vardakas, and Aristidis Likas. Revisiting silhouette aggregation, 2024. URL https://arxiv.org/abs/2401.05831. 6, 13

[35] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE (CVPR'06)*, 2006. 12

[36] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003. 15

A Algorithm for the *G-Loss* based fine-tuning

The algorithm below presents the outline of our proposed *G-Loss* based fine-tuning approach for language models in the document classification task, executed per minibatch k. The process begins with embedding extraction from the language model (Step 3), followed by normalization (Step 4). A crucial step is the construction of a similarity matrix using a Gaussian kernel function, which captures the semantic relationships between document embeddings (Step 5). The similarity matrix is then transformed into a normalized adjacency matrix (Step 6) and subsequently partitioned into two sets: one for which labels are known, and the other for which labels are masked (Step 7).

Algorithm 1 Fine-tuning process at minibatch k

1: **Input:** $\mathcal{V}_k$: Document set, Y_k: Label set, $\Phi(.)$: language model (LM),
2: **Hyperparameters:** γ, η, σ
3: **Output:** Optimized weights of LM, $\Phi(.)$
4: $X_k \leftarrow \Phi(\mathcal{V}_k)$ {Extract embeddings from LM}
5: $X_k \leftarrow X_k / \|X_k\|_2$ {Normalize embeddings}
6: **Compute similarity matrix:**
7: $\quad W_{ij} \leftarrow \exp\left(-\frac{\|X_{ki} - X_{kj}\|^2}{2\sigma^2}\right) - \text{diag}(W)$
8: $\quad \tilde{A} \leftarrow D^{-1/2} W D^{-1/2}$ {Normalize W}
9: $Y_{kl}, Y_{ku} \leftarrow \gamma$-split$(Y_k)$ {Partition label set}
10: **Label Propagation**
Transition matrix $\mathbf{T}$, where $\mathbf{T_{ij}} \leftarrow \frac{\tilde{A}_{ij}}{\sum_{m=0}^{B} \tilde{A}_{mj}}$
11: $\quad \hat{Y}_{ku} \leftarrow (I - \mathbf{T}_{uu})^{-1}.\mathbf{T}_{ul} Y_{kl}$
12: **Compute cross-entropy loss:**
13: $\quad \mathcal{L}_{\mathcal{G}} \leftarrow -\frac{1}{B_e} \sum_{j=1}^{B_e} \sum_{c=1}^{C} Y^c_{ku,j} \log\left(\hat{Y}^c_{ku,j}\right)$
14: **return** Loss $\mathcal{L}_{\mathcal{G}}$

A transition matrix is computed which represents the probability of transition from node i to node j (step 10). The soft labels for label-masked nodes are computed using close-form solution of LPA. The resulting soft labels (Step 11) serve as targets for the cross-entropy loss computation (Step 13), which specifically evaluates the model's predictions for instances with previously masked labels against their true labels. The loss is then return to be combined with CE loss and further backpropagation (Step 14). This approach effectively leverages both supervised label information and unsupervised document similarity relationships, enabling improved classification performance.

B Traditional loss functions-mathematical formulations

Supervised-contrastive loss

Supervised contrastive loss (SCL) [5] enhances traditional contrastive learning [35] by incorporating class labels to optimize embeddings. Given a minibatch of samples (x_i, y_i) and their representations $(z_i \in \mathbb{R}^d))$, the loss function minimizes intra-class and maximizes inter-class distances in the $(\mathbb{R}^d)$ embedding space. Mathematically, supervised contrastive loss can be expressed as:

$$\mathcal{L}_{SCL}(z_i) = -\frac{1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a / \tau)} \tag{9}$$

where z represents sample embeddings, $P(i)$, positive samples for the anchor z_i, $A(i)$,all other samples excluding anchor, and τ, scalar temperature parameter controlling class separation.

Triplet Loss Triplet Loss [33] constructs triplets (*anchor, positive, negative*) within each minibatch, encouraging the anchor-positive pair to be closer while pushing the negative further away. The loss is defined as:

$$\mathcal{L}_{\text{Triplet}} = \frac{1}{N_{bt}} \sum_{i=1}^{N_{bt}} \max\left(0, \|\mathbf{z}_i - \mathbf{z}_i^+\|_2^2 - \|\mathbf{z}_i - \mathbf{z}_i^-\|_2^2 + \alpha\right)$$

where N_{bt} is the number of triplets formed from datapoints in minibatch, and z_i, z_i^+, z_i^- are the anchor, positive, and negative sample embeddings, respectively, and α is a margin parameter enforcing separation.

Cosine-Similarity Loss Cosine-Similarity loss operates by maximizing cosine similarity between positive pairs and minimizing the similarity between negative pairs in a batch. Given a minibatch of size B, the loss is defined as:

$$\mathcal{L}_{\text{Cos-sim}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \left(\frac{z_1^i \cdot z_2^i}{\|z_1^i\| \cdot \|z_2^i\|} - label^i \right)^2$$

where N_p is the number of generated pairs in minibatch, $\frac{z_1^i \cdot z_2^i}{||z_1^i|| \cdot ||z_2^i||}$ is cosine similarity between embeddings z_1 and z_2, and $label \in \{0, 1\}$ denotes different-class (0) or same-class (1).

C Macro-Silhouette coefficient based evaluation details

In *Macro-Silhouette coefficient* based evaluation strategy, we monitor the cohesiveness of generated embeddings from learned model. The motive is that the model should learn to discriminate the embeddings of same/different class documents.

Specifically, at each fine-tuning epoch, we calculate the macro-Silhouette score from embeddings generated from models' most recent parameters and the true labels for the validation set. The macro-Silhouette coefficient [34] for a data point x_i is given by:

$$s(x_i) = \frac{b(x_i) - a(x_i)}{\max(a(x_i), b(x_i))}$$

where $a(x_i)$ is the average intra-class distance, while $b(x_i)$ is the average nearest-class distance. Silhouette-coefficient for each class, C_i is computed as follow:

$$S_C = \frac{1}{|C|} \sum_{x_i \in C_i} s(x_i)$$

And overall macro-Silhouette coefficient is:

$$S_{macro} = \frac{1}{K} \sum_{i=1}^{K} S_C(C_i)$$

where K is the number of classes.

Traditionally, Silhouette score is aggregated using micro-averaging (averaging over all data points), however it can be highly sensitive to cluster imbalance and noise. Macro-Silhouette score provides more robustness and reflects the quality of embedding and learning progress of the language model. We implement early stopping when this score plateaus to capture optimal embedding representations.

D Sigma value computation for *Gloss-SQRT*

The **Gaussian kernel** is defined as:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right).$$

Let

$$d = \|\mathbf{x}_i - \mathbf{x}_j\|^2$$

(a fixed constant for given $\mathbf{x}_i, \mathbf{x}_j$), so the kernel simplifies to:

$$k(\sigma) = \exp\left(-\frac{d}{2\sigma^2} \right).$$

We compute partial derivatives with respect to σ (assuming $\sigma > 0$).

1. First Partial Derivative $\frac{\partial k}{\partial \sigma}$

Rewrite

$$k = \exp\left(-\frac{d}{2}\sigma^{-2}\right).$$

Using the chain rule:

$$\frac{\partial k}{\partial \sigma} = \exp\left(-\frac{d}{2\sigma^2}\right) \cdot \frac{\partial}{\partial \sigma}\left(-\frac{d}{2}\sigma^{-2}\right).$$

Compute the derivative inside:

$$\frac{\partial}{\partial \sigma}\left(-\frac{d}{2}\sigma^{-2}\right) = -\frac{d}{2} \cdot (-2)\sigma^{-3} = \frac{d}{\sigma^3}.$$

Thus:

$$\boxed{\frac{\partial k}{\partial \sigma} = \frac{d}{\sigma^3}\exp\left(-\frac{d}{2\sigma^2}\right)}.$$

2. Second Partial Derivative $\frac{\partial^2 k}{\partial \sigma^2}$

Differentiate $\frac{\partial k}{\partial \sigma}$ using the product rule:

$$\frac{\partial}{\partial \sigma}\left(\frac{d}{\sigma^3}\exp\left(-\frac{d}{2\sigma^2}\right)\right) = d \cdot \frac{\partial}{\partial \sigma}\left(\sigma^{-3}\exp\left(-\frac{d}{2}\sigma^{-2}\right)\right).$$

Set $u = \sigma^{-3}$ and $v = \exp\left(-\frac{d}{2}\sigma^{-2}\right)$. Then:

$$\frac{\partial u}{\partial \sigma} = -3\sigma^{-4}, \quad \frac{\partial v}{\partial \sigma} = \exp\left(-\frac{d}{2\sigma^2}\right) \cdot \frac{d}{\sigma^3}.$$

Apply the product rule:

$$\frac{\partial^2 k}{\partial \sigma^2} = d\left[\sigma^{-3} \cdot \left(\frac{d}{\sigma^3}\exp\left(-\frac{d}{2\sigma^2}\right)\right) + \exp\left(-\frac{d}{2\sigma^2}\right) \cdot (-3\sigma^{-4})\right].$$

Simplify:

$$\frac{\partial^2 k}{\partial \sigma^2} = d\exp\left(-\frac{d}{2\sigma^2}\right)\left[\frac{d}{\sigma^6} - \frac{3}{\sigma^4}\right].$$

$$\boxed{\frac{\partial^2 k}{\partial \sigma^2} = \frac{d(d-3\sigma^2)}{\sigma^6}\exp\left(-\frac{d}{2\sigma^2}\right)}.$$

3. Points of Inflection

Points of inflection occur where $\frac{\partial^2 k}{\partial \sigma^2} = 0$ or is undefined, and the **concavity changes**.

Critical Points

$$\frac{\partial^2 k}{\partial \sigma^2} = 0 \implies d(d-3\sigma^2) = 0$$

(since $\exp(\cdot) > 0$ and $\sigma^6 > 0$ for $\sigma > 0$).

Given $d = \|\mathbf{x}_i - \mathbf{x}_j\|^2 > 0$ (assuming $\mathbf{x}_i \neq \mathbf{x}_j$), solve:

$$d - 3\sigma^2 = 0 \implies \sigma^2 = \frac{d}{3} \implies \sigma = \sqrt{\frac{d}{3}} \quad \text{(valid since } \sigma > 0\text{)}.$$

Concavity Change

- For $\sigma < \sqrt{d/3}$: $d - 3\sigma^2 > 0 \Rightarrow \frac{\partial^2 k}{\partial \sigma^2} > 0$ (concave up). - For $\sigma > \sqrt{d/3}$: $d - 3\sigma^2 < 0 \Rightarrow \frac{\partial^2 k}{\partial \sigma^2} < 0$ (concave down). - At $\sigma = \sqrt{d/3}$, concavity changes from up to down.

Undefined Point: $\sigma = 0$ is not in the domain (kernel undefined).

Conclusion

$$\boxed{\text{Point of inflection at } \sigma = \sqrt{\frac{d}{3}} \quad \text{where } d = \|\mathbf{x}_i - \mathbf{x}_j\|^2}$$

Median Pairwise Distance as Optimal Choice

The median pairwise distance is often chosen as the kernel bandwidth σ because:

- It is **robust to outliers**.
- Reflects the **dominant scale** of the data.
- Maximizes **kernel sensitivity** for typical pairs.

E Singularity and Stability in the Closed-form LPA Solution

The closed-form solution employed in this work follows the classical formulation of graph-based semi-supervised learning [7, 36]. When pairwise similarities are computed using the Gaussian kernel, the resulting similarity matrix W is positive semi-definite. This property ensures that the corresponding normalized transition matrix

$$T = D^{-1}W$$

is row-stochastic or, in the case of unlabeled data, *substochastic*. The submatrix T_{uu}, representing transitions among unlabeled nodes, therefore satisfies the substochastic property.

Since T_{uu} is substochastic, its spectral radius $\rho(T_{uu})$ is strictly less than one, i.e.,

$$\rho(T_{uu}) < 1.$$

This directly implies that the matrix $(I - T_{uu})$ is **non-singular**, ensuring the existence and stability of the closed-form solution.

Formally, the spectral radius of a square matrix A is defined as

$$\rho(A) = \max_i |\lambda_i|,$$

where λ_i denotes the eigenvalues of A. A matrix $(I - A)$ is invertible if and only if 1 is not an eigenvalue of A. Consequently, if all eigenvalues of A satisfy $|\lambda_i| < 1$, the matrix $(I - A)$ is guaranteed to be invertible (i.e., non-singular).

Therefore, the closed-form label propagation solution,

$$F = (I - T_{uu})^{-1} T_{ul} Y_l,$$

is well-defined and numerically stable under the practical assumptions governing the mini-batch graph construction used in GLOSS.

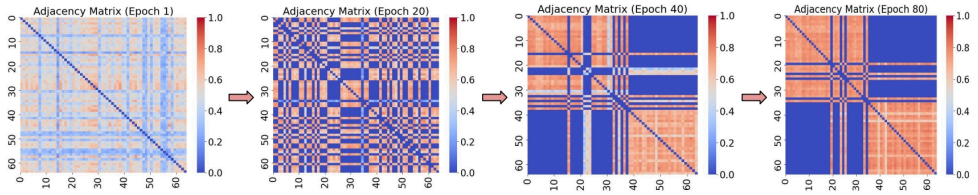
Furthermore, the condition $\rho(T_{uu}) < 1$ also guarantees *convergence* of iterative propagation schemes. When this condition holds, the matrix inverse can be equivalently expressed via the Neumann series expansion:

$$(I - T_{uu})^{-1} = I + T_{uu} + T_{uu}^2 + T_{uu}^3 + \cdots$$

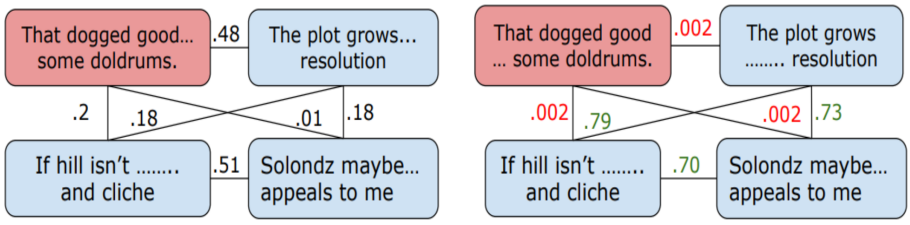
This infinite series converges exactly when the spectral radius of T_{uu} is less than one. This property provides the theoretical foundation for the convergence of iterative graph-based methods, such as Label Propagation (LPA) and PageRank, as both involve repeated multiplication by a transition matrix whose spectral radius is less than unity.

F Visualizations

We visualize the evolution of the document graph during fine-tuning with *G-Loss* using a sample subset from the MR dataset to illustrate how the model progressively refines the underlying semantic structure. Figure 3a shows the temporal evolution of adjacency heatmaps, capturing how the coherence among same-class data points increases across epochs. Figure 3b provides a schematic example of this process: the initial graph (left) appears noisy and poorly aligned with class boundaries, indicating that pre-trained embeddings lack task-specific separation. After fine-tuning (right), the graph exhibits stronger intra-class (green) and weaker inter-class (red) connections. This structural transformation highlights how *G-Loss* promotes the emergence of coherent, class-consistent, and structure-aware representations aligned with the ground-truth labels.



(a) Document similarity evolution for `Bert-base-uncased` with *G-Loss* fine-tuning. Heatmaps show single minibatch similarity matrices over the epochs.



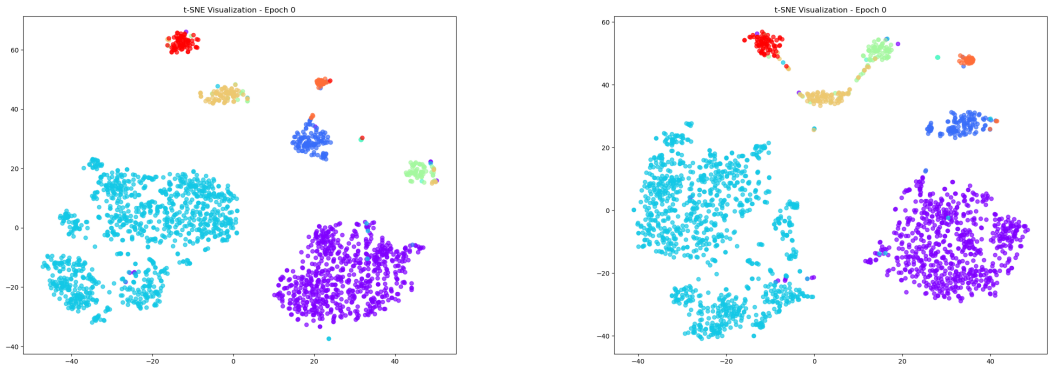
(b) Demo graph evolution during training: the left graph depicts the initial connectivity, while the right graph highlights the strengthened intra-class (green) and weakened inter-class (red) connections, reflecting enhanced semantic coherence and cluster formation.

Figure 4 presents t-SNE visualizations of the learned embeddings from `BERT-base-uncased` model, demonstrating the effectiveness of *G-Loss* in creating well-separated cluster representations. The visualizations reveal distinct class boundaries and improved intra-class cohesion compared to baseline methods.

G Hyperparameter selection

Table 5 and Table 6 summarize the search ranges and empirically optimal intervals obtained via Optuna-based tuning and ablation studies across benchmark datasets. The *G-Loss* framework is controlled by four key hyperparameters: learning rate (η), label-hiding ratio (γ), weighting coefficient (λ), and Gaussian kernel bandwidth (σ). Ablation studies show that $\gamma \in [0.5, 0.7]$—hiding about 30-50% of labels during label propagation—yields the best performance. For the weighting coefficient λ, experimental results demonstrate that prioritizing the structural regularization term with $\lambda \in [0.7, 0.9]$ enhances geometric alignment in the embedding space. The Gaussian kernel bandwidth σ is dataset-dependent and lacks a universal optimal range, necessitating data-specific tuning. To address this challenge, we introduce two *G-Loss* variants: *G-Loss-SQRT* employs an analytical approximation for σ derived from the methodology presented in Section D, offering computational efficiency for resource-constrained deployments; conversely, *G-Loss-O* uses `Optuna` to identify the optimal σ, for maximal performance at the cost of increased hyperparameter tuning overhead. This dual design allows practitioners to balance computational cost and accuracy as needed.

Additionally, Figure 6 in the appendix presents an ablation study on the weighting factor λ, which effectively modulates the trade-off between local label supervision via CE loss and global embedding structure alignment using *G-Loss*. Our results show that a range of $0.7 - 0.9$ consistently yields strong performance across benchmark datasets, indicating that this weighting achieves an optimal



(a) *G-Loss* **(b)** SCL

Figure 4: t-SNE visualization of learned embeddings with *G-Loss* and SCL on R8 dataset with `BERT-base-uncased`. *G-Loss* shows a clear separation as compared to SCL.

integration of supervised signals and graph-structured relational information, thereby enhancing both predictive accuracy and embedding coherence.

Loss	**Hyperparameters**
Cross Entropy	Learning rate (η)
Cosine-Similarity	Learning rate (η)
Triplet	Learning rate (η)
CE + SCL	Learning rate (η), Temperature (τ), Weight factor(λ)
CE + GLoss-SQRT	Learning rate (η), Label-hiding ratio (γ), Weight-factor (λ)
CE + GLoss-O	Learning rate (η), Label-hiding ratio (γ), Weight-factor (λ), Gaussian-kernel width (σ)

Table 5: List of hyperparameters for *G-Loss* and other baseline losses.

	Optuna search range	**Optimal range**
(η)	$[1e-05, 2e-05, 3e-05, 4e-05, 5e-05]$	-
(γ)	$0.1 - 0.9$	$0.5 - 0.7$
(σ)	$0.01 - 10.0$	Data-specific
(λ)	$0.1 - 0.9$	$0.7 - 0.9$

Table 6: G-Loss Hyperparameter Selection and Optimal Range: For the hyperparameter σ, we propose a resource-constrained approach, CE+G-Loss-SQRT, where the value of σ can be determined mathematically, eliminating the need for Optuna-based tuning.

H Detailed per epoch timing breakdown

The figure 5 presents a detailed per-epoch timing breakdown for different training configurations — *G-Loss*, Triplet, SCL, and Cos-sim — highlighting the relative computational cost across major components: forward pass, backward pass, optimizer step, and I/O operations. *G-Loss* requires 27.74 seconds per epoch, nearly identical to Triplet loss (27.58 sec) and Cosine similarity (27.76 sec), while being 15% faster than SCL (32.62 sec). The forward pass accounts for around $32 - 33\%$ of the computation across all loss functions. Notably, *G-Loss's* graph-based components add minimal overhead: graph construction consumes only 0.18 seconds (0.7% of total time) and LPA operations require 0.07 seconds (0.2% of total time). These results demonstrate that the structural loss integration introduces minimal computational burden while maintaining efficiency comparable to conventional training pipelines.

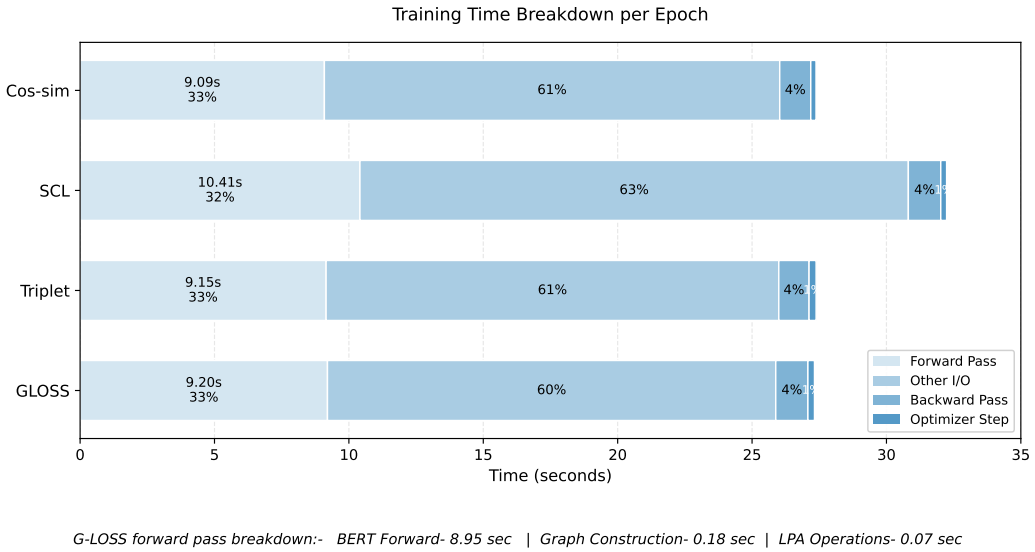


Figure 5: Training time breakdown per epoch across different loss functions on the MR dataset with `BERT-base-uncased` model. Each horizontal bar represents the total epoch time (in seconds) decomposed into four components: forward pass, other I/O operations, backward pass, and optimizer step. The percentages indicate the contribution of each component to the total epoch time. Despite incorporating graph construction and label propagation operations, *G-Loss* maintains comparable computational efficiency to baseline methods.

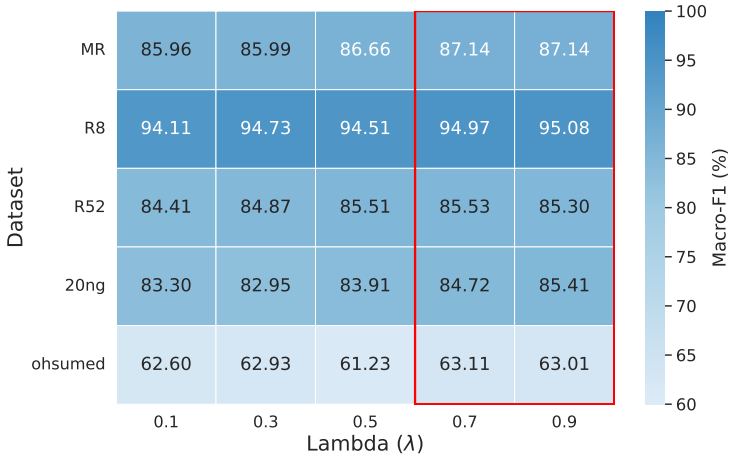


Figure 6: Hyperparameter sensitivity analysis of G-Loss on `BERT-base-uncased`: loss weighting factor Lambda (λ) vs performance