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

Aditya Sharma[a], **Vinti Agarwal**[a], **Rajesh Kumar**[b],

[a]BITS Pilani, India [b]Bucknell University, USA

{p20200470, vinti.agarwal}@pilani.bits-pilani.ac.in, rk042@bucknell.edu

Abstract

Traditional loss functions, including cross-entropy, contrastive, triplet, and supervised contrastive losses, used for fine-tuning pre-trained language models such as BERT, operate only within local neighborhoods and fail to account for the global semantic structure. We present *G-Loss*, a graph-guided loss function that incorporates semi-supervised label propagation to use structural relationships within the embedding manifold. *G-Loss* builds a document-similarity graph that captures global semantic relationships, thereby guiding the model to learn more discriminative and robust embeddings. We evaluate *G-Loss* on five benchmark datasets covering key downstream classification tasks: MR (sentiment analysis), R8 and R52 (topic categorization), Ohsumed (medical document classification), and 20NG (news categorization). In the majority of experimental setups, *G-Loss* converges faster and produces semantically coherent embedding spaces, resulting in higher classification accuracy than models fine-tuned with traditional loss functions.

1 Introduction

Language models, ranging from BERT [1] and RoBERTa [2], with millions of parameters, to large-scale models such as GPT [3] and LLaMA [4], with parameters in billions, have transformed natural language processing (NLP). These models follow a two-stage approach: unsupervised pre-training on large unlabeled corpora to learn general representations, followed by supervised fine-tuning with labeled data using a task-specific loss function. Fine-tuning, however, presents challenges, including the need for large labeled datasets, high computational cost, and sensitivity to task-specific objectives. Traditional fine-tuning losses, such as cross-entropy, primarily minimize prediction error while overlooking the semantic structure within the data.

Several strategies have been explored to improve fine-tuning effectiveness. Classical approaches, such as BERT and RoBERTa, employ cross-entropy loss with a classification head attached to the encoder [1, 2], which enforces class separation in the embedding space. This approach, however, ignores relationships between individual samples and relies solely on label supervision [5]. Pairwise and triplet-based methods, such as SBERT [6], address this limitation by optimizing similarity objectives that pull semantically related samples closer while pushing unrelated samples apart. These methods improve local neighborhood structure but are computationally expensive because they require generating many positive and negative pairs. More importantly, they still optimize local relationships independently, without enforcing global structural alignment. For example, ensuring that A is close to B and far from C does not guarantee a coherent global positioning of B and C unless the pair (B, C) is explicitly optimized. This locality constraint often limits generalization to unseen data.

To address these limitations, we propose a shift from local pairwise optimization to global structural alignment by modeling semantic relationships across all pairs through a graph. We introduce *G-Loss*, a graph-based fine-tuning strategy that enforces both local and multi-hop consistency within a unified framework. While the approach can be applied to any encoder-based model, we focus here on transformer-based encoders for NLP tasks. *G-Loss* integrates the semi-supervised Label Propagation

A Sharma et al., G-Loss: Graph-Guided Fine-Tuning of Language Models. *Proceedings of the Fourth Learning on Graphs Conference (LoG 2025)*, PMLR 338, Hybrid Event, December 10–12, 2025.

Algorithm (LPA) [7] into the fine-tuning process, allowing label information to diffuse from labeled to unlabeled nodes within a similarity graph. This enables the model to benefit from both explicit labels and implicit relational information, consistent with the manifold assumption in semi-supervised learning [8], which posits that proximity in feature space correlates with label similarity.

A key advantage of using LPA is its parameter-free nature, which ensures scalability and efficiency across model sizes and datasets. Another distinguishing feature of our framework is its self-reinforcing nature: the graph structure and the embeddings co-evolve during fine-tuning. As embeddings improve, they reshape the graph structure, which in turn guides further embedding refinement, yielding better global alignment.

Unlike earlier works that use LPA with static, full-dataset graphs for vision tasks [9] or employ similarity graphs as soft targets for image classification [10], *G-Loss* introduces three key techniques. First, it dynamically constructs mini-batch graphs using evolving embeddings, supporting inductive learning while reducing memory cost. Second, it integrates LPA directly into the loss function, eliminating the need for separate pseudo-labeling and retraining. Third, it maintains embedding consistency throughout training while streamlining the fine-tuning pipeline.

The major contributions of this work are summarized as follows:

- We propose a fine-tuning framework that integrates a graph-based loss derived from dynamically evolving graphs and the semi-supervised Label Propagation Algorithm (LPA). Unlike traditional losses that optimize only local relationships, our approach enforces global semantic consistency—the coevolution of the graph structure and the embedding space during fine-tuning yields robust, discriminative representations.
- We implement and evaluate the proposed framework using `BERT`, `RoBERTa`, and `DistilBERT`, demonstrating its applicability across model scales. The code scripts are publicly available at [1].
- We conduct experiments on five benchmark datasets covering binary and multi-class classification tasks with varying class distributions. Performance is evaluated using accuracy, macro F1-score, silhouette score, and convergence time.
- We compare our method with standard loss functions, including supervised contrastive, cosine similarity, triplet, and cross-entropy losses.

The remainder of this paper is organized as follows. Section 2 reviews related literature. Section 3 details the fine-tuning process with *G-Loss*. Section 4 describes the datasets, language models, loss baselines, and evaluation criteria. Section 5 presents and discusses the results. Section 7 concludes the paper and outlines future directions.

2 Related work

Loss functions play a crucial role in fine-tuning language models (LMs) by shaping how representation learning adapts to downstream tasks. Fine-tuning models such as `BERT` [1], `RoBERTa` [2], and `ALBERT` [11] typically involves appending a classification head and optimizing the cross-entropy (CE) loss. Although effective for classification, CE treats training samples independently and fails to preserve structural consistency within the embedding space.

To address this limitation, similarity-based learning methods focus on relational structure. Sentence-BERT (`SBERT`) [6] uses a Siamese network with triplet and cosine similarity losses for pairwise alignment of semantically related sentences. `SimCSE` [12] introduces contrastive learning with dropout-based augmentation, promoting more discriminative and stable embeddings through positive and negative pair comparison. `SimCSE++` [13] improves upon this by reducing negative-pair noise and adding a dimension-wise contrastive objective to prevent feature collapse. Despite these advances, contrastive and triplet-based losses remain locally optimized, capturing only sample-level relationships and failing to represent global semantic structure [14]. Gunel et al. [15] proposed the supervised contrastive learning (SCL) objective to complement cross-entropy loss, improving class-level separation but still without explicit structural modeling.

Graph-based approaches, in contrast, explicitly encode relational dependencies among samples. Graph Neural Networks (GNNs) extend representation learning by propagating contextual information

[1] https://github.com/saditya13/G-Loss-LoG

across graph nodes [16]. Graph Convolutional Networks (GCN) [16] first demonstrated feature aggregation through message passing, and `TextGCN` [17] extended this idea to document classification via heterogeneous word-document graphs. `TensorGCN` [18] further incorporated syntactic and sequential dependencies to enrich the textual graph representations.

Hybrid language model–graph frameworks have emerged to combine contextual embeddings with graph reasoning. For instance, `BertGCN` integrates BERT embeddings within a GCN to improve text classification [19]. Other approaches include: (1) cascading models, where the language model produces embeddings subsequently processed by a GNN for classification [20–22]; (2) co-training models, where LMs and GNNs are optimized jointly with shared objectives [22–24]; and (3) frozen LM strategies, where the language model remains fixed while only the GNN parameters are updated [25, 26].

While these methods show promise, they face two key limitations: (1) *static graph construction*, where the graph structure is fixed and does not adapt to evolving embeddings, and (2) *high computational overhead* from joint GNN–LM optimization. For example, `BertGCN` relies on a static, full-dataset graph and a memory bank, which significantly increase training time and memory usage.

We propose *G-Loss* to address these limitations through dynamic graph construction during fine-tuning. *G-Loss* enables the language model to learn and refine semantic structure adaptively, unlike traditional fine-tuning objectives (cross-entropy, triplet, contrastive) that overlook global dependencies, or graph-based methods (`TextGCN`, `BertGCN`) that require costly static, precomputed graphs. *G-Loss* bridges this gap by combining dynamic graph adaptation with efficient label propagation, allowing seamless integration of graph-structured learning into language model fine-tuning.

3 Proposed fine-tuning framework

3.1 Task description

The goal is to improve document-level representation learning for supervised text classification by integrating global semantic structure into the fine-tuning process of pre-trained language models. Formally, given a document set $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ and a corresponding label set $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ spanning $\mathcal{C}$ classes, the objective is to learn a mapping function

$$f : d_i \mapsto y_i, \quad f(d_i) = \arg\max_{c \in \mathcal{C}} P(y_i = c \mid d_i; \theta),$$

where θ denotes the parameters of the fine-tuned language model.

Unlike conventional fine-tuning, which optimizes sample-level prediction loss, we reformulate the task also to preserve semantic relationships among samples. This is achieved by representing each minibatch of documents as a graph $\mathcal{G} = (\mathcal{V}_k, \mathcal{E}_k, \mathcal{X}_k)$, where nodes correspond to document embeddings, edges encode semantic similarity, and $\mathcal{X}_k \in \mathbb{R}^{B \times d}$ contains the language model representations. The model is fine-tuned on the training set to minimize classification error and graph-based relational inconsistency jointly. The validation set is used to monitor convergence and for hyperparameter tuning $(\lambda, \gamma, \sigma)$, and the held-out test set is used only for final evaluation.

3.2 G-Loss: Graph-driven loss computation

Given the training set $\mathcal{T} \subset \mathcal{D}$ and corresponding labels $\mathcal{Y}$, fine-tuning proceeds over minibatches $\{b_1, b_2, \ldots, b_m\}$. For each minibatch b_k, documents are encoded into dense representations using a pre-trained language model $\Phi(.)$, and a similarity-based document graph is constructed in which nodes represent documents and edge weights w_{ij} capture pairwise semantic affinity. To incorporate global structural consistency into the fine-tuning process, a subset of node labels is masked according to a masking ratio γ, and the semi-supervised Label Propagation Algorithm (LPA) [7] is applied to infer these hidden labels while keeping known labels fixed. The discrepancy between the inferred and true labels defines the graph-based loss $\mathcal{L}_{\mathcal{G}}$. The overall objective combines this graph-based loss with the standard cross-entropy loss $\mathcal{L}_{\mathcal{CE}}$ using a weighting factor λ:

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

This composite loss is optimized via backpropagation, allowing the model to jointly refine both the embeddings and the evolving graph structure during fine-tuning.

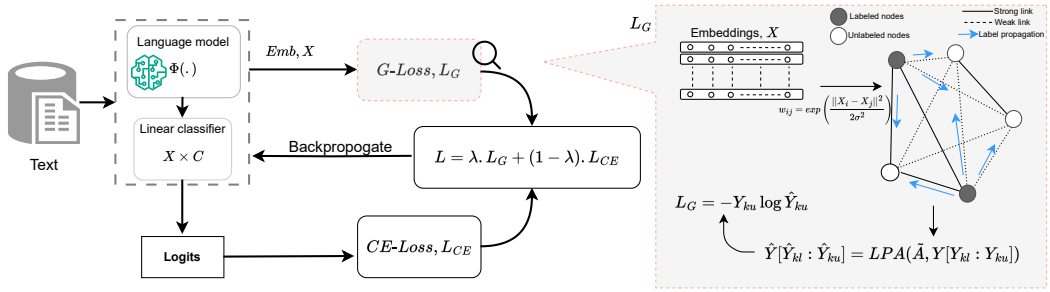


Figure 1: The language model $\Phi(.)$ encodes input text into embeddings, which are used by both a linear classifier (producing logits) and a similarity-based document graph. In the graph, nodes represent documents and edge weights w_{ij} capture pairwise semantic similarity. A subset of node labels (Y_{ku}) is masked and inferred using the LPA, yielding predicted labels $\hat{Y}_{ku}$. The resulting graph-based loss $\mathcal{L}_{\mathcal{G}}$ is combined with the cross-entropy loss $\mathcal{L}_{\mathcal{CE}}$ using weighting factor λ. The composite loss is then backpropagated to update both the language model and classifier parameters.

Figure 1 illustrates this process for a single minibatch. The key stages of the proposed framework are described below.

(Step 1) Embedding extraction. Each document in minibatch b_k, $k \in \{1, 2, \ldots, m\}$, is encoded by the language model $\Phi(.)$ into a d-dimensional embedding space:

$$X_k = \Phi(\text{Text in } b_k),$$

where $X_k \in \mathbb{R}^{B \times d}$ represents contextual embeddings for the batch of size B. These embeddings serve as node features for graph construction. The quality of these embeddings directly affects label propagation and downstream classification performance.

(Step 2) Graph construction. A fully connected weighted graph $\mathcal{G}_k = (\mathcal{V}_k, \mathcal{E}_k, X_k)$ is constructed for minibatch b_k. The edge weight between nodes i and j is computed using a Gaussian kernel:

$$w_{ij} = \exp\left(-\frac{\|X_i - X_j\|^2}{2\sigma^2}\right), \quad w_{ii} = 0,$$

where X_i and X_j are document embeddings, and σ controls neighborhood sensitivity in kernel space.

We evaluate two approaches for choosing σ: (1) *G-Loss-SQRT*, which analytically estimates $\sigma = \sqrt{d_1/3}$, where d_1 is the median Euclidean distance across all pairs (see Appendix D); and (2) *G-Loss-O*, which optimizes σ through hyperparameter search. *G-Loss-SQRT* provides a computationally efficient alternative by avoiding hyperparameter tuning overhead. The Gaussian kernel ensures that highly similar documents are strongly connected while dissimilar ones have weaker connections.

To stabilize propagation, the adjacency matrix is symmetrically normalized:

$$\tilde{A} = D^{-1/2} W D^{-1/2},$$

where $D_{ii} = \sum_{j=1}^{B} w_{ij}$ is the degree matrix. This ensures that label propagation distributes information effectively without amplifying numerical instabilities.

(Step 3) Label propagation. A hyperparameter γ splits the minibatch node set $\mathcal{V}_k$ into two disjoint subsets:

- $\mathcal{V}_{kl}$(Labeled subset) : a γ fraction of nodes with known labels Y_{kl};
- $\mathcal{V}_{ku}$ (Evaluation subset): unlabeled nodes whose labels Y_{ku} are masked.

The column-stochastic transition matrix T is defined as:

$$T_{ij} = P(j \to i) = \frac{\tilde{A}_{ij}}{\sum_{m=1}^{B} \tilde{A}_{mj}}.$$

Using the closed-form solution of LPA [7], the inferred labels for the evaluation subset are:

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

where I is the identity matrix, T_{uu} is the submatrix for unlabeled nodes, and T_{ul} captures transitions from labeled to unlabeled nodes.

(Step 4) Loss computation. The graph-based loss $\mathcal{L}_{\mathcal{G}}$ measures the discrepancy between the inferred labels $\hat{\mathbf{Y}}_{ku}$ and the true labels $\mathbf{Y}_{ku}$ over the evaluation subset of size $B_e = (1-\gamma)B$:

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

where $y_{ku,j}^{c}$ and $\hat{y}_{ku,j}^{c}$ denote the ground-truth and predicted probabilities for class c of the jth evaluation node, and C is the number of classes.

(Step 5) Loss integration. The final objective unifies the previously defined components by combining the graph-based loss $\mathcal{L}_{\mathcal{G}}$ and the cross-entropy loss $\mathcal{L}_{\mathcal{CE}}$:

$$\mathcal{L} = \lambda\, \mathcal{L}_{\mathcal{G}} + (1-\lambda)\, \mathcal{L}_{\mathcal{CE}}, \tag{2}$$

where $\lambda \in [0,1]$ balances the contribution of both terms. The cross-entropy component is expressed as:

$$\mathcal{L}_{\mathcal{CE}} = -\frac{1}{B} \sum_{i=1}^{B} \sum_{c=1}^{C} y_{i,c} \log \hat{y}_{i,c}. \tag{3}$$

Here, B denotes the minibatch size.

(Step 6) Model optimization and dynamic graph updating. The language model and classifier parameters are jointly optimized via backpropagation using the composite loss $\mathcal{L}$:

$$\mathbf{X}'_k \leftarrow \mathbf{X}_k - \eta \nabla \mathcal{L}, \tag{4}$$

where η denotes the learning rate. As embeddings evolve during fine-tuning, the graph structure is recomputed after each update:

$$\mathbf{W}'_{ij} = \exp\left(-\frac{\|\mathbf{X}'_i - \mathbf{X}'_j\|^2}{2\sigma^2}\right). \tag{5}$$

This dynamic adaptation ensures that the similarity graph remains consistent with the evolving embedding space, enabling more coherent representations and improved classification performance.

This completes one iteration of fine-tuning under the *G-Loss* framework. A detailed algorithm of the approach is provided in section A of the appendix. The next section describes the experimental setup, benchmark datasets, and evaluation protocol used to assess its effectiveness.

4 Experimental setup

4.1 Datasets and downstream tasks

We evaluate G-Loss on five widely used English text classification benchmarks. *Movie Review (MR)* [27]: binary sentiment classification (positive/negative). *R8* and *R52* [28]: topic classification derived from Reuters-21578 with 8 and 52 categories, respectively. *Ohsumed* [29]: medical text classification using MEDLINE abstracts labeled under 23 MeSH categories. *20 Newsgroups (20NG)* [30]: news articles grouped into 20 discussion topics. For consistency, we adopt the data splits released by Lin et al. [31], ensuring identical data partitions. Dataset statistics are summarized in Table 1.

4.2 Language models and traditional losses

Language Models. *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 hidden units, 110M parameters) [1], `RoBERTa-large` (24 layers, 1024 hidden units, 356M parameters) [2], and `DistilBERT-base-uncased` (6 layers, 768 hidden units, 66M parameters) [32].

Table 1: Dataset statistics used for evaluation. The Train, Validation, and Test columns denote 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 assess *G-Loss* under two configurations: integrated and standalone.

In the **integrated setup**, the hybrid loss jointly optimizes the language model and classifier parameters, following the framework in Figure 1. This setting aligns with standard fine-tuning approaches such as BERT, RoBERTa, and BertGCN [15, 19].

In the **standalone setup**, only the language model parameters are updated using the computed loss, while the classifier is trained separately on the resulting embeddings.

- **Integrated:** *G-Loss* (with CE) is compared against cross-entropy alone, and the hybrid Supervised Contrastive + CE loss (SCL+CE) [15].
- **Standalone:** *G-Loss* is compared against triplet loss [33], Supervised Contrastive Loss (SCL) [5], and cosine-similarity loss.

Mathematical formulations for all baselines are provided in Appendix B.

4.3 Fine-tuning convergence and hyperparameter tuning

Evaluation strategies and early stopping. In the integrated configuration, training is monitored using the macro F1-score on the validation set. Early stopping is applied if no improvement occurs within a fixed patience window. The classifier is a linear layer of size $E \times C$, where E is the embedding dimension and C the number of classes.

In the standalone configuration, convergence is monitored using the macro-silhouette score [34] computed on validation labels (formula in Appendix C). Unlike the standard silhouette score, this variant accounts for class imbalance. After fine-tuning, a linear classifier is trained on the frozen embeddings, and test predictions are obtained from this downstream model. This setup avoids training a classification head jointly with the language model, reducing parameter count and computation.

Hyperparameter selection and final classifier. Hyperparameter tuning is performed using `Optuna`. For *G-Loss*, we tune: Gaussian kernel width $\sigma \in [0.1, 10]$, learning rate $\eta \in \{1\text{e}{-5}, 2\text{e}{-5}, 3\text{e}{-5}, 4\text{e}{-5}, 5\text{e}{-5}\}$, label masking ratio $\gamma \in [0.1, 0.9]$, and weighting factor $\lambda \in [0.1, 0.9]$. Baseline losses use comparable optimization: temperature $\tau \in [0.01, 1]$ and learning rate η for SCL, and learning rate η for cross-entropy, cosine-similarity, and triplet losses. Details on the hyperparameter search are provided in Appendix G. All experiments were conducted on a single NVIDIA A100 80GB GPU.

Evaluation metrics and protocol. We report test accuracy and macro F1-score as primary metrics, ensuring comparability with prior work such as TextGCN, TensorGCN, and BertGCN. Macro F1 is particularly relevant for imbalanced datasets (e.g., R52 and Ohsumed). To assess computational efficiency, we also record fine-tuning time to convergence and per-epoch training time for *G-Loss* and all baseline losses, as shown in Table 3.

5 Results and discussion

5.1 *G-Loss* with integrated pipeline

We experimentally compare the proposed *G-Loss* with strong baselines: CE loss and SCL+CE loss (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
	SCL + CE[15]	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
	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
RoBERTa -large	CE	89.92±1.07 / 89.92±1.07	97.75±0.20 / 93.46±0.37	95.91±0.56 / 83.71±1.09	84.89±0.36 / 84.28±0.39	75.03±0.24 / 67.95±0.59
	SCL + CE [15]	90.21±0.91/ 90.20±0.92	97.24±0.19 / 93.11±0.60	95.95±0.45 / 82.33±1.37	85.31±0.26 / 84.90±0.25	73.67±0.33 / 65.55±0.42
	G-Loss-SQRT + CE	90.53±0.51 / 90.53±0.51	97.26±0.12 / 93.02±0.21	96.14±0.47 / 84.31±1.02	85.06±0.28 / 84.61±0.58	74.82±0.52 / 66.55±0.74
	G-Loss-O + CE	90.87±0.62 / 90.87±0.62	97.49±0.05 / 93.04±0.12	96.44±0.63 / 84.34±0.85	85.19±0.31 / 84.23±0.73	75.35±0.42 / 68.58±1.02
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
	SCL + CE [15]	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
	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

Table 2 shows classification accuracy and macro F1-score across all datasets and language model variants. *G-Loss* variants closely match or marginally surpass other baselines regardless of the underlying language model, achieving improvements of $0.03\% - 1.02\%$ in accuracy and $0.16\% - 1.35\%$ in macro F1-score compared to the second-best-performing loss, across all datasets. Notably, *G-Loss* variants generalize across different language model families, indicating its effectiveness in enhancing a variety of language models. While SCL+CE remains competitive on some datasets, it underperforms on large models such as RoBERTa-large. These findings confirm that our graph-based loss function consistently improves language model fine-tuning for text classification.

5.2 *G-Loss* as a standalone Objective

Table 3 presents the standalone evaluation of *G-Loss* compared with traditional loss functions (detailed in section 4.2). We report results for *G-Loss-O*, the best-performing variant. Overall, *G-Loss-O* delivers performance that is competitive or superior to, traditional objectives. In particular, it achieves the highest Macro F1 on challenging datasets, such as Ohsumed (61.43%), and attains the highest silhouette scores across most benchmarks. These results demonstrates that beyond improving predictive accuracy, *G-Loss-O* enforces the formation of more semantically coherent embedding spaces.

Building on these performance gains, *G-Loss* also demonstrates notable training efficiency. It converges faster, requiring fewer epochs to reach optimal performance (e.g., 23 epochs on R8). Importantly, despite incorporating graph-based components, the average per-epoch training time remains comparable to existing baselines, leading to faster overall convergence and improved training time. We further analyze the computational efficiency of *G-Loss*. Figure 6 in the appendix provides a detailed timing breakdown per epoch for *G-Loss* and competing baselines. *G-Loss* requires 9.2 seconds per epoch (BERT forward pass: 8.95 sec; graph construction: 0.18 sec; LPA operations: 0.07 sec), compared to Supervised Contrastive Loss (10.41 sec), Triplet Loss (9.15 sec), and Cosine Similarity Loss (9.09 sec). This breakdown confirms that additional graph-based components - graph construction, normalization, and label propagation - introduce negligible computational overhead during training.

Adding to this, table 4 provides a quantitative comparison between *G-Loss-O* and SCL (the closest competitor) across five benchmark datasets, evaluating both representation quality and training efficiency. Complementing this analysis, Appendix I reports paired t-test results comparing *G-Loss* against all baseline losses. These statistical tests confirm the robustness of the improvements delivered by *G-Loss*, with all comparisons yielding $p < 0.05$.

5.3 GLUE benchmark performance

To assess the scalability and generalization of *G-Loss* on large-scale natural language understanding tasks, we conduct experiments on two GLUE benchmark datasets: SST-2 and QNLI (Table 6). SST-2 is a binary sentiment classification task from the Stanford Sentiment Treebank with approximately $67k$ training examples. QNLI, derived from the Stanford Question Answering Dataset, is a large-scale natural language inference task with approximately $105k$ sentence pairs in training set and requires sentence-level semantic reasoning. Across both tasks and different language models, *G-Loss-O* outperforms both cross-entropy (CE) and supervised contrastive loss (SCL). For example, on SST-2

Table 3: Comparison of *G-Loss-O* with traditional losses using `BERT-base-uncased`. *G-Loss-O* achieves superior or competitive accuracy and macro-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	86.18±0.49	86.18±0.49	0.5725	1258.41	33.04	35
	Triplet	86.30±0.24	86.29±0.24	0.5331	941.55	27.50	29
	Cos-sim	86.08±0.45	86.08±0.45	0.4479	654.56	27.70	20
	GLoss-O	86.22±0.25	86.22±0.25	0.5507	810.85	27.77	25
R8	SCL	97.80±0.07	93.54±0.52	0.4869	1088.58	25.56	38
	Triplet	97.30±0.05	93.35±0.98	0.6581	727.85	21.19	30
	Cos-sim	97.74±0.12	93.97±0.11	0.7383	4960.64	21.65	200
	GLoss-O	97.83±0.13	93.95±0.31	0.7870	572.29	21.77	23
R52	SCL	95.11±0.74	79.72±1.02	0.4561	1170.50	30.41	34
	Triplet	95.08±0.60	77.78±0.63	0.4662	5496.14	25.05	189
	Cos-sim	95.21±1.11	79.57±2.01	0.3972	1925.82	25.79	64
	GLoss-O	95.79±0.74	79.92±1.71	0.4824	1095.93	25.87	39
Ohsumed	SCL	68.54±0.69	57.97±0.93	0.1373	691.03	15.70	39
	Triplet	68.68±0.40	58.34±0.55	0.1222	1576.44	13.09	105
	Cos-sim	69.33±0.38	59.58±0.40	0.1393	3064.31	13.42	200
	GLoss-O	70.25±0.37	61.43±1.16	0.1468	633.50	13.50	41
20NG	SCL	84.45±0.79	83.66±0.57	0.5507	11671.10	52.76	200
	Triplet	84.23±0.30	83.80±0.28	0.4538	8516.99	43.46	171
	Cos-sim	84.11±0.30	83.67±0.31	0.5524	10155.44	45.10	200
	GLoss-O	84.94±0.16	84.26±0.22	0.5849	8596.50	46.10	180

Table 4: Comparison of *G-Loss-O* and SCL (the closest competitor) across silhouette scores and training efficiency. Higher silhouette scores indicate better cluster separability; lower time per epoch indicates faster optimization.

Dataset	Silhouette Score (↑)		Average Time per Epoch (s, ↓)		Observations
	G-Loss-O	SCL	G-Loss-O	SCL	
MR	0.5507	0.5725	27.77	33.04	−3.8% Sil ↓ / **16.0% faster**
R8	0.7870	0.4869	21.77	25.56	**+61.6% Sil** ↑ / 14.8% faster
R52	0.4824	0.4561	25.87	30.41	**+5.8% Sil** ↑ / 14.9% faster
Ohsumed	0.1468	0.1373	13.50	15.70	**+6.9% Sil** ↑ / 14.0% faster
20NG	0.5849	0.5507	46.10	52.76	**+6.2% Sil** ↑ / 12.6% faster

with `BERT-base-uncased`, *G-Loss-O+CE* improves accuracy from 92.10% to 93.32% (+1.22%). Similar improvements are observed for RoBERTa and DistilBERT, demonstrating that *G-Loss* scales effectively to larger datasets and maintains strong performance across diverse architectures.

5.4 Comparison with SOTA models

Table 5 provides a 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 baselines, accuracy is reported as the primary evaluation metric.

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

Table 5: Performance comparison (accuacy in %) with SOTA across benchmark datasets. *G-Loss-O* consistently improves over graph-based methods and BERT, 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

Table 6: Accuracy (%) across large-scale GLUE benchmarks SST-2 and QNLI, highlighting the strong generalization ability of *G-Loss*.

Model	Loss	SST2	QNLI
BERT-base -uncased	CE	91.20	89.08
	SCL+CE	92.10	90.06
	GLoss-O+CE	93.32	89.62
RoBERTa -large	CE	94.84	93.71
	SCL+CE	95.83	94.01
	GLoss-O+CE	95.88	94.27
DistilBERT -base	CE	90.14	87.86
	SCL+CE	90.36	87.18
	GLoss-O+CE	91.06	88.34

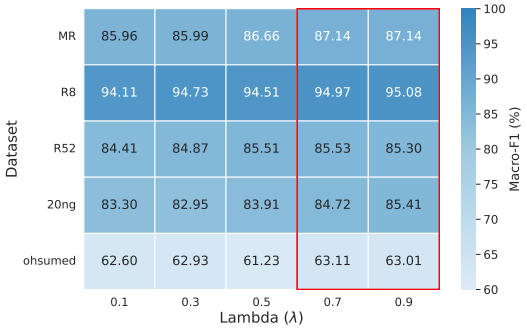


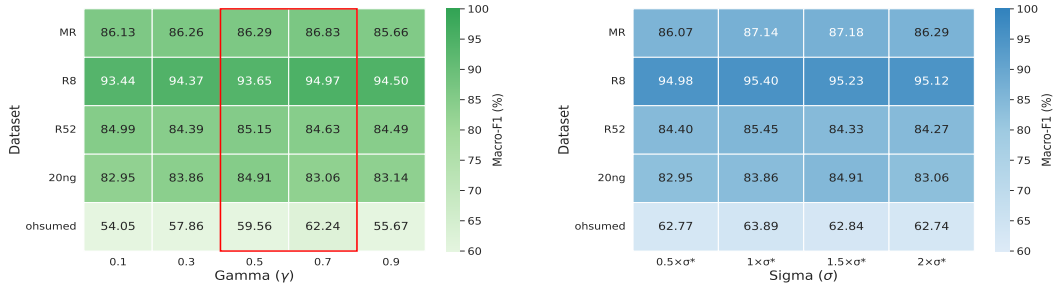
Figure 2: Loss weighting factor Lambda (λ) vs performance of *G-Loss* on `BERT-base-uncased`

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 remaining two datasets. 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 substantially increases computational resource requirements. Above results demonstrate that *G-Loss* not only strengthens language model fine-tuning but also bridges the gap between graph-based and transformer-based paradigms more efficiently.

6 Ablation study

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 (γ) $\in [0, 1]$, plays an important role in *G-Loss* function, by controlling the proportion of labels that remain visible during the LPA iterations. Specifically, a fraction of $(1 - \gamma)$ of the labels is masked and LPA is used to infer these hidden labels. Figure 3a 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.



(a) **(b)**

Figure 3: Sensitivity of *G-Loss* (on `BERT-base-uncased`) to variations in (a) the label-hiding ratio γ and (b) the Gaussian kernel width σ.

Impact of sigma in Gaussian similarity on performance of *G-Loss*. The parameter σ in the Gaussian similarity function controls graph connectivity and affects *G-Loss* performance. Small σ value induce sparse, localized neighborhoods that preserve fine-grained class distinctions, while larger values generate denser graphs that may blur class boundaries. Figure 3b shows performance across σ multipliers relative to `Optuna`-optimized value (σ^*). We observe dataset-dependent sensitivity: MR and R8 datasets demonstrate remarkable stability, with performance varying by less than 1% across all σ values. In contrast, 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 and (2) the optimal σ balances local precision with global connectivity, underscoring the importance of proper hyperparameter tuning to maximize G-Loss effectiveness.

Additionally, figure 2 presents an ablation study on the weighting factor λ in *G-Loss-O*. We further compliment our analysis with a hyperparameter sensitivity study of *G-Loss* using tornado plot in section J. To illustrate the qualitative impact of different objectives, we provide t-SNE visualizations of learned embeddings for MR and R8 datasets in section F of the appendix. These visual analyses offer additional insight into how *G-Loss* influences representation structure compared to traditional loss functions. Finally, we note that evaluating *G-Loss* on extremely large encoder-based models such as DeBERTa-xxlarge (1.5B parameters) was not feasible due to computational limitations. Training such models requires GPU memory far exceeding our available resources, and thus remained outside the scope of this study.

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 across five benchmark datasets and three transformer architectures show *G-Loss's* consistent effectiveness in achieving performance improvements over 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.

Acknowledgements

We thank Saurabh Rahul Bhandari, Apoorva Gulati, and Henil Shalin Panchal for their helpful discussions and valuable feedback.

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, 14

[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. 2, 3, 5, 17

[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. 3

[17] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. *CoRR*, 2018. 3, 8

[18] Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. Tensor graph convolutional networks for text classification, 2020. 3, 8

[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, 8

[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, 14

[34] John Pavlopoulos, Georgios Vardakas, and Aristidis Likas. Revisiting silhouette aggregation, 2024. 6, 15

[35] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE (CVPR'06)*, 2006. 14

[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. 17

Appendix Contents

Section	Page
A Algorithm for the G-Loss Based Fine-Tuning	14
B Traditional Loss Functions — Mathematical Formulations	14
Supervised-Contrastive Loss	14
Triplet Loss	14
Cosine-Similarity Loss	14
C Macro-Silhouette Coefficient-Based Evaluation Details	15
D Sigma Value Computation for G-Loss-SQRT	15
First and Second Partial Derivatives	15
Points of Inflection	15
Median Pairwise Distance as Optimal Choice	15
E Singularity and Stability in the Closed-form LPA Solution	17
F Visualizations	18
G Hyperparameter Selection	18
H Detailed Per-Epoch Timing Breakdown	19
I Statistical Significance Test on G-Loss	20
J Hyperparameter Sensitivity Analysis of G-Loss	21
Figures	
Document Similarity and Demo Graph Evolution During Fine-Tuning	18
t-SNE Visualization of Learned Embeddings (G-Loss vs. SCL, R8)	19
Training Time Breakdown per Epoch (MR, BERT-base-uncased)	20
Hyperparameter Sensitivity (Tornado Plot, MR)	21
Tables	
List of Hyperparameters for G-Loss and Baseline Losses	19
G-Loss Hyperparameter Selection and Optimal Range	19
Statistical Significance — Paired t-Test Results	20
Algorithms	
Fine-Tuning Process at Minibatch k	14

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 constructing a similarity matrix using a Gaussian kernel function to capture 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 with known labels and the other with masked labels (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: $Y_{kl}, Y_{ku} \leftarrow \gamma\text{-split}(Y_k)$ {Partition label set}
10: **Label Propagation**
$\quad$ 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 that represents the probability of transitioning from node i to node j (step 10). The soft labels for label-masked nodes are computed using a closed-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 returned to be combined with the CE loss and further backpropagation (Step 14). This approach effectively leverages both supervised label information and unsupervised document-similarity relationships, thereby improving 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:

$$\mathcal{L}_{SCL}(z_i) = -\frac{1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p/\tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a/\tau)} \tag{6}$$

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 maximizes cosine similarity between positive pairs and minimizes it between negative pairs within 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 the learned model. The goal is for the model to learn to discriminate between the embeddings of same-class/different-class documents.

Specifically, at each fine-tuning epoch, we calculate the macro-Silhouette score using embeddings generated from the models' most recent parameters and the true labels of 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 the 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. The Macro-Silhouette score provides greater robustness and reflects the quality of the embedding and the language model's learning progress. 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

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

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.

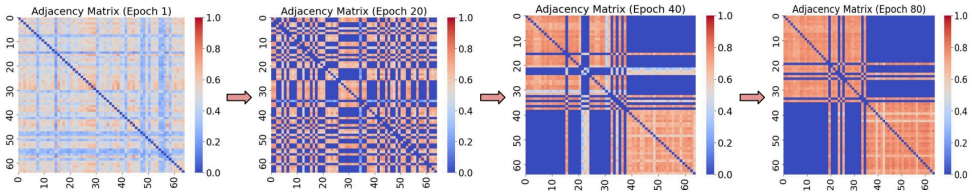
Furthermore, the condition $\rho(T_{uu}) < 1$ also guarantees *convergence* of iterative propagation schemes. When this condition holds, the matrix inverse can be equivalently expressed via the Neumann series expansion:

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

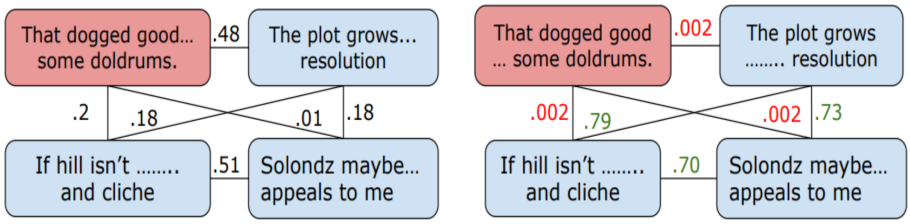
This infinite series converges exactly when the spectral radius of T_{uu} is less than one. This property provides the theoretical foundation for the convergence of iterative graph-based methods, such as Label Propagation (LPA) and PageRank, as both involve repeated multiplication by a transition matrix 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 4a shows the temporal evolution of adjacency heatmaps, capturing how the coherence among same-class data points increases across epochs. Figure 4b 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.

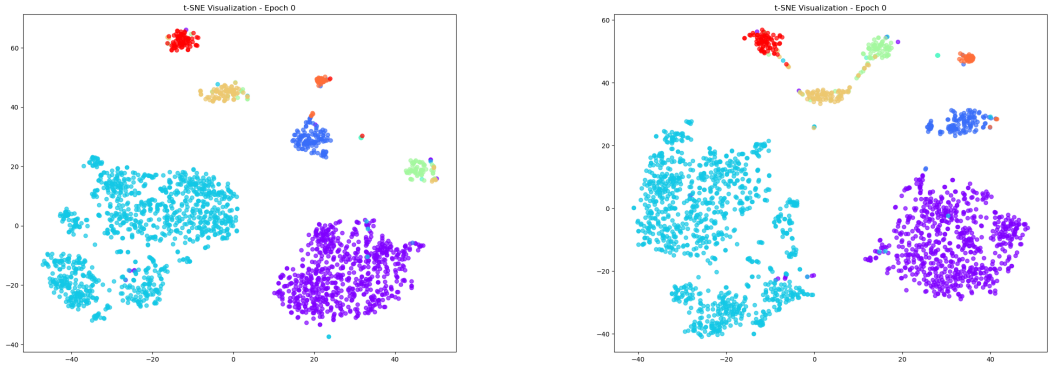


(b) Demo graph evolution during training: the left graph depicts the initial connectivity, while the right graph highlights the strengthened intra-class (green) and weakened inter-class (red) connections, reflecting enhanced semantic coherence and cluster formation.

Figure 5 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 7 and Table 8 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,



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

Figure 5: 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.

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 2 presents an ablation study on the weighting factor λ, which effectively modulates the trade-off between local label supervision via the CE loss and global alignment of the embedding structure 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 optimally integrates supervised signals and graph-structured relational information, thereby enhancing both predictive accuracy and embedding coherence.

Table 7: List of hyperparameters for *G-Loss* and other baseline losses.

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 8: 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.

	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)	$0.1 - 0.9$	$0.7 - 0.9$

H Detailed per epoch timing breakdown

The figure 6 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

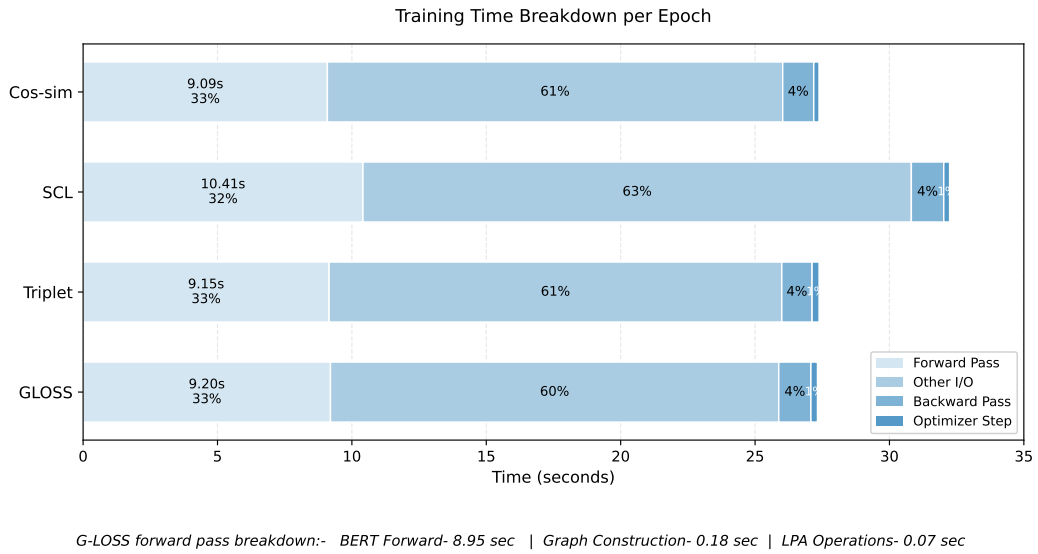


Figure 6: Training time breakdown per epoch across different loss functions on the MR dataset with `BERT-base-uncased` model. Each horizontal bar represents the total epoch time (in seconds) decomposed into four components: forward pass, other I/O operations, backward pass, and optimizer step. The percentages indicate each component's contribution to the total epoch time. Despite incorporating graph construction and label propagation operations, *G-Loss* maintains comparable computational efficiency to baseline methods.

Table 9: Mean accuracies of *G-Loss* across benchmark datasets compared with alternative losses (SCL, Cosine Similarity, and Triplet). The table reports absolute performance differences and paired t-test p-values, demonstrating that *G-Loss* offers statistically significant improvements ($p < 0.05$) across all comparisons, highlighting its robustness and consistent advantage. Significance stars represent: ** - $p < 0.05$, *** - $p < 0.01$, **** - $p < 0.001$.

Comparison	μ(G-Loss)	μ(Baseline)	$\Delta\mu$	p-value	Observations
G-Loss vs SCL	87.05	86.41	0.63	0.006239	***
G-Loss vs Cos-Sim	87.05	86.32	0.73	0.000493	****
G-Loss vs Triplet	87.05	86.50	0.55	0.019647	**

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 integrating structural loss introduces minimal computational burden while maintaining efficiency comparable to that of conventional training pipelines.

I Statistical significance test on *G-Loss*

Table 9 demonstrates that *G-Loss* outperforms baselines across all benchmark datasets. Relative to SCL, Cosine Similarity, and Triplet objectives, *G-Loss* yields absolute accuracy gains of 0.63, 0.73, and 0.55 points, respectively. The statistical significance of these improvements, as indicated by paired t-tests ($p < 0.05$ in each case), underscores the robustness of *G-Loss* in leveraging graph-based label propagation for enhanced classification performance. The significance-star patterns (marked with *** for $p < 0.01$ and **** for $p < 0.001$) confirm that the performance gains are not merely due to random variation. This consistent performance advantage, coupled with strong statistical significance, validates *G-Loss* as an effective training objective for fine-tuning language models.

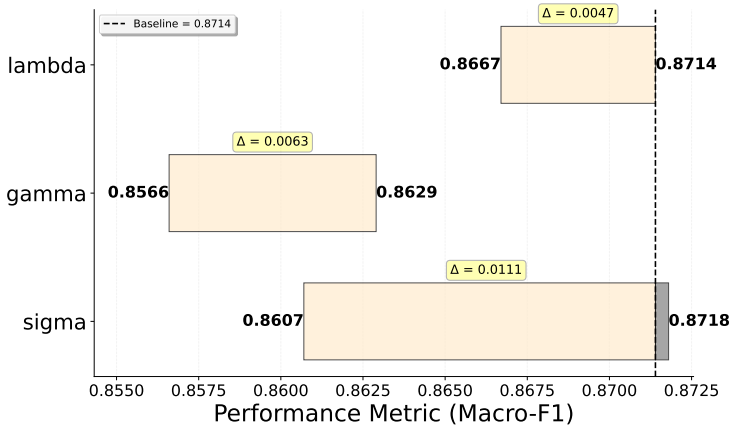


Figure 7: Hyperparameter sensitivity analysis of *G-Loss* on MR dataset with `BERT-base-uncased`. Each bar represents the maximum macro-F1 deviation when perturbing a single hyperparameter around its optimal value.

J Hyperparameter sensitivity analysis of *G-Loss*

We further evaluate the robustness of *G-Loss* through a targeted hyperparameter sensitivity analysis on the MR dataset using `BERT-base-uncased` (Figure 7). Taking the optimal configuration as a reference point (Macro-F1 = 0.8714), we perturb each hyperparameter within a small neighborhood while holding the others fixed. The Gaussian kernel width (σ) emerges as the dominant factor, inducing a performance variation of ± 0.0111. In comparison, the loss-weighting coefficient (λ) and label-hiding ratio (γ) produce substantially smaller deviations of ± 0.0063 and ± 0.0047, respectively, indicating limited sensitivity to these parameters. Overall, the analysis shows that *G-Loss* is robust to small variations in the hyperparameter values, with only σ exhibiting moderate influence on downstream performance.