Weed Out, Then Harvest: Dual Low-Rank Adaptation is an Effective Noisy Label Detector for Noise-Robust Learning

Bo Yuan, Yulin Chen, Yin Zhang*
Zhejiang University, Hangzhou, China
{byuan,yulinchen,yinzh}@zju.edu.cn

Abstract

Parameter-efficient fine-tuning (PEFT) large language models (LLMs) have shown impressive performance in various downstream tasks. However, in many real-world scenarios, the collected training data inevitably contains noisy labels. To learn from noisy labels, most solutions select samples with small losses for model training. However, the selected samples, in turn, impact the loss computation in the next iteration. An inaccurate initial selection can create a vicious cycle, leading to suboptimal performance. To break this cycle, we propose Delora, a novel framework that decouples the sample selection from model training. For sample selection, Delora establishes a noisy label detector by introducing clean and noisy LoRA. Benefiting from the memory effect, the clean LoRA is encouraged to memorize clean data, while the noisy LoRA is constrained to memorize mislabeled data, which serves as a learnable threshold for selecting clean and noisy samples. For model training, Delora can use carefully selected samples to fine-tune language models seamlessly. Experimental results on synthetic and real-world noisy datasets demonstrate the effectiveness of Delora in noisy label detection and text classification.

1 Introduction

LLMs are extremely powerful, yet they are expensive to train. By balancing performance with practicality, PEFT has become a popular technique for adapting the LLM for downstream applications. Notable PEFT methods include Low-Rank Adaptation (LoRA) (Hu et al., 2022), adaptors (Houlsby et al., 2019), and prompts (Liu et al., 2022). Instead of fine-tuning all weights, PEFT fixes the backbone model parameters while adding a few learnable parameters for adaptation. While promising, PEFT techniques rely on perfectly labeled datasets, which may not be readily available in many real-world applications, limiting their broader application.

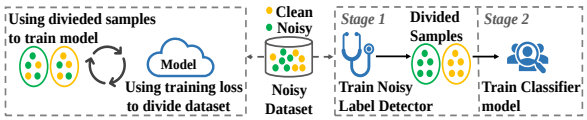


Figure 1: A comparison between other sample selection methods (left) and our method (right) for LNL tasks. Our method decouples sample selection (stage 1) from model training (stage 2) by training a noisy label detector and classifier model separately.

To tackle this issue, a recent work (Kim et al., 2024) explores PEFT methods on imperfectly labeled datasets (*i.e.*, datasets with noisy labels) to learn with noisy labels (LNL). It uses training losses to select clean data and suggests a routing-based PEFT method that preferentially trains PEFT modules on clean data. Similar to previous LNL methods (Han et al., 2018; Shu et al., 2019; Qiao et al., 2022; Yuan et al., 2024) based on sample selection, they both adopt the "small-loss" mechanism to select clean data because the model tends to fit clean samples earlier than noisy samples during training. However, these methods are inherently affected by the label noise, as losses used for sample selections are extracted from the model that is being trained. Specifically, the sample selection process affects subsequent training, and training loss in turn influences the sample selection. If the initial sample selection is poor, it leads to an inescapable vicious cycle. We contend that in such a strategic feedback loop, a good sample selection can itself fall into a new paradox, akin to: ***It's a catch-22 situation: A good sample selection requires generalizability, while generalizability requires sample selection.***

Given this, let us leave the existing framework and revisit the LNL task. The main difficulty here is that we need to select clean samples to train a strong model while avoiding the effects of noisy

*Corresponding Author

Findings of the Association for Computational Linguistics: ACL 2025, pages 15292–15311

samples. *Can we, perhaps, decouple sample selection from the model training, making them independent of each other?* With this question in mind, we propose a new framework that firstly leverages PEFT modules to construct a noisy label detector for sample selection and then trains the model using selected samples, as shown in Figure 1.

In the first stage, we introduce two distinct PEFT modules (*i.e.*, clean LoRA and noisy LoRA) to construct a noisy label detector. The parameters in clean LoRA are termed the ideal parameters used to memorize clean data, while parameters in noisy LoRA are called noise parameters used to memorize mislabeled data. Our noisy label detector can be derived from the cross-entropy between the predictions of clean/noisy LoRA on the training text and its corresponding labels. If the text exhibits a higher cross-entropy value with the noisy LoRA than the clean LoRA, it is considered a correctly labeled sample, *i.e.*, a clean sample. This design draws inspiration from a recent study (Liu et al., 2024) that decomposes an image into its constituent subject and style, represented as two distinct LoRAs. This insight has inspired us to explore different LoRAs to memorize clean and noisy samples separately in LNL. However, as we cannot directly obtain clean and noisy samples from datasets, controlling LoRAs to accomplish our notion is a challenge. Note that the memorizing effect (Arpit et al., 2017) demonstrates that deep networks would first memorize clean samples and then noisy samples. Based on this, we introduce dynamic regularization to adjust the parameter updates of the two LoRAs over time. Specifically, we constrain noise parameter updates of noisy LoRA in the early training stage and make ideal parameters of clean LoRA completely memorize clean data. As training progresses, the restrictions on noisy LoRA are gradually lessened while the constraints on clean LoRA are reinforced, resulting in noisy samples being mostly memorized by noisy LoRA.

In the second stage, we train the classifier model using samples carefully selected through the noisy label detector. In this step, we first utilize selected clean samples as contextual references for reliable relabeling of noisy samples, and then merge the clean samples and relabeled noisy samples to fine-tune the classifier model. In this way, we can obtain a denoised fine-tuning dataset while maximizing data utility to improve the generalization ability of our framework in LNL tasks. It is worth noting that the sample selection is independent of the in-training classifier model, which can effectively avoid the issue of vicious cycles. Overall, our main contributions can be summarized as follows:

- We propose a new framework that decouples sample selection from model training to address the LNL tasks, effectively avoiding the vicious cycles common in existing solutions.
- For sample selection, we introduce two LoRAs to construct a noisy label detector: the ideal parameters of clean LoRA to memorize clean samples and the noise parameters of noisy LoRA to memorize noisy samples.
- Based on the memory effect, we dynamically constrain the parameters of LoRAs so that noisy LoRA absorbs the side effects of noisy samples and clean LoRA fits clean samples.
- We conduct extensive experiments across diverse text classification datasets under varying noise conditions, demonstrating the superiority of Delora over existing baselines in both noise label detection and text classification.

2 Related Work

2.1 Sample Selection for LNL.

Model fine-tuning often relies on large-scale, high-quality data. However, the dataset inevitably introduces noisy labels during collection. For LNL, sample selection methods are popular solutions that strive to select clean samples from noisy datasets using specific criteria, such as the widely applied "small-loss" mechanism or model predictions. Among them, (Han et al., 2018; Shu et al., 2019; Qiao et al., 2022) set a fixed threshold for loss value to divide the noisy data, Yuan et al. (2024) further proposes a dynamic-enhanced threshold strategy to improve the previous method based on fixed thresholds. However, they all require manually setting thresholds, which increases the cost of hyper-parameter tuning. For our proposed Delora, the noisy LoRA prediction functions as a learnable "threshold" for identifying noisy labels. Moreover, these approaches remain unstable and susceptible to vicious cycles (self-confirmation bias), particularly in high-noise scenarios. This instability stems from their inherent reliance on the in-training model (Feng et al., 2024) and their exclusive focus on learning with label noise from scratch. Different from previous studies, our Delora decouples sample selection and classifier model training to break this vicious cycle.

2.2 Parameter-Efficient Fine-tuning.

As LLMs get bigger, PEFT is more essential for conserving resources. A lot of strategies, including LoRA, adaptors, and prompt learning, have been put forth by researchers to improve fine-tuning effectiveness. Among them, the PEFT based on LoRA (Hu et al., 2022) is popular and widely used. Recently, Kim et al. (2024) explores PEFT's robustness to noisy labels and finds that PEFT's limited capacity enhances robustness to noisy samples but also hinders learning from clean samples. Then, they propose a routing-based method to adaptively activate PEFT modules. However, they still require manually setting a fixed loss threshold and relying on losses of the in-training model for sample selection. We address their shortcomings and leverage PEFT's limited capacity to separately memorize clean and noisy samples.

3 Preliminaries

Problem Setup. Given a training dataset $\mathcal{D}$=$\{(x_i, y_i)\}_{i=1}^{N}$ with N samples and K classes, where $y \in \{1, \ldots, K\}$ is the observed label of the sample x, and y is possibly the incorrect label.

LoRA fine-tuning. LoRA (Low-Rank Adaptation) is a PEFT (parameter-efficient fine-tuning) technique for effectively modifying LLMs for a new downstream task. The main idea behind LoRA is that while fine-tuning, the weight updates Δw to the base model weights $w_0 \in \mathbb{R}^{m \times n}$ have a low intrinsic rank. As a result, the update Δw may be broken down into two low-rank matrices, $B \in \mathbb{R}^{m \times r}$ and $A \in \mathbb{R}^{r \times n}$, for efficient parameterization, with $\Delta w = BA$. For r, it means the intrinsic rank of Δw with $r \ll min(m, n)$. During training, only A and B are updated to find suitable $\Delta w = BA$ targeting specific tasks while keeping w_0 constant. For inference, the updated weight matrix w can be obtained as $w = w_0 + \Delta w$. Denoting the prediction with LoRA modules as $f(x, w_0 + \Delta w)$, the objective with an arbitrary loss function $\mathcal{L}$ can be formulated as follows:

$$\min_{\Delta w} \mathcal{L}(x) = \mathcal{L}(f(x, w_0 + \Delta w), y), \qquad (1)$$

where x and y are the training sample and its label, respectively. The model f with LoRA modules Δw is only updated during training.

4 Method

In this section, we present our proposed denoising learning framework Delora in detail. The main idea is to decouple sample selection and model training during the fine-tuning of LLMs on downstream tasks, avoiding the issue of vicious cycles. Overall, Delora comprises two pivotal stages. In the first stage, Delora introduces dual LoRAs to construct a noisy label detector, selecting clean samples and noisy samples. In the second stage, Delora leverages the curated clean samples and relabeled noisy samples to train the classifier model, further boosting the performance of our framework on text classification in the context of noisy environments. Figure 2 shows our proposed framework.

4.1 Identifying Noisy Labels with Dual LoRAs

In this stage, our core challenge lies in the construction of a noisy label detector for identifying noisy samples. The previous strategy is to set a fixed threshold for loss value, such that the clean samples are associated with a smaller mean loss value and noisy ones with bigger values. However, these methods (Qiao et al., 2022; Feng et al., 2024) rely on the manual setting of thresholds, which restricts its practical applicability. Inspired by a recent study (Liu et al., 2024), we introduce two distinct LoRAs memorizing the information of clean and noisy samples to overcome these limitations.

Introducing dual LoRAs to construct a noisy label detector. Given a language model with weights w_0^i, we introduce two LoRAs: clean LoRA $L_c = \Delta w_c^i$ and noisy LoRA $L_n = \Delta w_n^i$. Here, i denotes the index of layers of transformers. For simplicity, we drop the superscript i since our method operates over all the LoRA-enabled weight matrices of the base model. The parameters of Δw_c are ideal parameters to memorize clean samples (desired memorization), while the parameters of Δw_n are noisy parameters to memorize noisy samples (undesired memorization). During training, the clean LoRA Δw_c aims to uncover distinguishable features by minimizing the cross-entropy (CE) between the predicted results of samples and their corresponding label, while the noisy LoRA Δw_n serves as a learnable sample-dependent cross-entropy threshold to select clean samples. Specifically, the learnable threshold ϕ_i for the i-th training sample x_i with a label y_i is formulated as:

$$\phi_i = CE(f(x_i, w_0 + \Delta w_n), y_i), \qquad (2)$$

which represents the cross-entropy between the observed label and the undesired prediction generated by noisy LoRA Δw_n. Based on this threshold, the

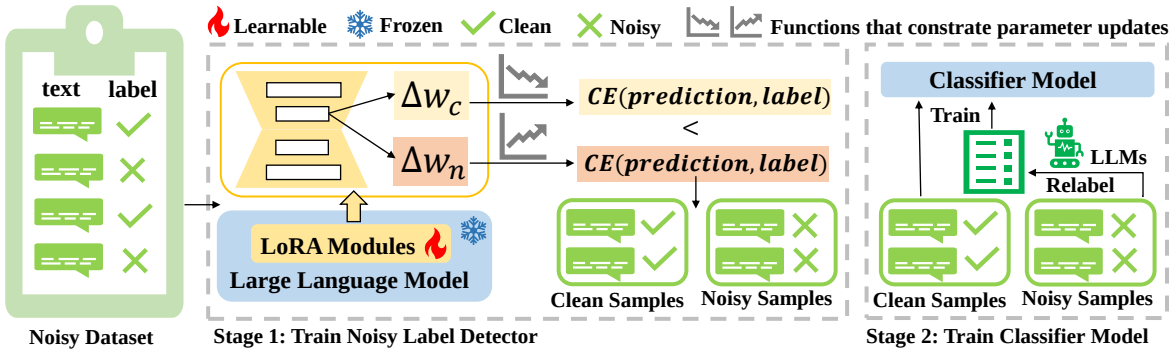


Figure 2: The architecture of our proposed framework Delora. Stage 1: We introduce two separate LoRAs (clean LoRA Δw_c and noisy LoRA Δw_n) to construct the noisy label detector. Stage 2: We leverage the selected clean samples and relabeled noisy samples to train the classifier model.

clean subset $\mathcal{D}_c$ of $\mathcal{D}$ can be constructed as follows:

$$\mathcal{D}_c = \{(x_i, y_i) \mid CE(f(x_i, w_0 + \Delta w_c), y_i) < \phi_i\}. \tag{3}$$

The proposed selection strategy outperforms traditional small loss-based approaches in two ways: (1) it is more practical by using data-driven thresholds, which eliminates the need for manual setting; (2) the introduction of PEFT modules (LoRAs) improves its robustness to label noise, allowing it to identify challenging hard noise. However, the key challenge is optimizing dual LoRAs (*i.e.*, the clean and noisy LoRA), and the noisy label detector.

Optimization for Dual LoRAs. Studies of the memory effect show that deep networks would first memorize clean samples and then noisy samples. From the perspective of network parameter updates, if we can strengthen the updates for parameters of clean LoRA Δw_c during early training, and weaken the updates in later training, the clean samples can be better memorized by Δw_c. On the other hand, for noisy LoRA Δw_n, we should limit their parameter Δw_n updates in early training, and strengthen their updates in later training to memorize noisy samples. This intuition drives us to formulate a new optimization objective for the dual LoRA to achieve our goal. To be specific, we design the corresponding optimization objective:

$$\mathcal{L}_{LoRA} = \tau_1(t)\Delta\sigma_c + \tau_2(t)\Delta\sigma_n, \tag{4}$$

where $\tau_1(t)$ and $\tau_2(t)$ are two mathematical functions that are relevant to the training epoch t, $\Delta\sigma_c = ||\Delta w_c^t - \Delta w_c^{t-1}|| = ||\sigma_c^t(B) - \sigma_c^{t-1}(B)|| + ||\sigma_c^t(A) - \sigma_c^{t-1}(A)||$ is defined to measure the parameter change of clean LoRA Δw_c between two adjacent epochs via the Euclidean distance, Δw_c^t are the parameters of Δw_c obtained in epoch t, $\sigma_c^t(A)$ and $\sigma_c^t(B)$ correspond to the parameters of two low-rank matrices in epoch t, respectively. More specifically, $\Delta\sigma_c$ limits the parameter change of Δw_c between two adjacent epochs. If the weight, *i.e.*, $\tau_1(t)$, is high, $\Delta\sigma_c$ will decrease quickly. Namely, modifications to Δw_c^t are restricted. For noisy LoRA, the update of $\Delta\sigma_n = ||\Delta w_n^t - \Delta w_n^{t-1}|| = ||\sigma_n^t(B) - \sigma_n^{t-1}(B)|| + ||\sigma_n^t(A) - \sigma_n^{t-1}(A)||$ is also constrained in a similar manner.

Define for τ_1 and τ_2. As analyzed, combining the memory effect of deep networks, we should dynamically adjust the parameter update of clean LoRA Δw_c and noisy LoRA Δw_n. That is to say, during the early training, Δw_c should be updated rapidly to fit clean data; while in later stages, its updates should slow down to prevent overfitting mislabeled data. In contrast, the update pattern for Δw_n follows the opposite strategy. Therefore, we define $\tau_1(t)$ as a rising function to constrain the parameter update ($\Delta\sigma_c$) of clean LoRA Δw_c and $\tau_2(t)$ as a decreasing function to constrain the parameter update ($\Delta\sigma_n$) of noisy LoRA Δw_n. In this work, we set $\tau_1(t) = t^{h_1}$ and $\tau_2(t) = t^{-h_2}$, where h_1 and h_2 are two hyperparameters.

Optimization for the Noisy Label Detector. After addressing the optimization challenges of the dual LoRA, we shift our focus to optimizing the noisy label detector. For the noisy label detector, optimizing the threshold ϕ_i in an end-to-end manner might be challenging due to its indirect involvement in forward propagation. We can still optimize the associated parameters by stating the clean probability of the i-th samples as follows:

$$p_i^c = \frac{e^{CE(f(x_i, w_0+\Delta w_c), y_i)}}{e^{CE(f(x_i, w_0+\Delta w_c), y_i)} + e^{CE(f(x_i, w_0+\Delta w_n), y_i)}} \tag{5}$$

Obviously, selecting samples with clean probability $p_i^c > 0.5$ corresponds to the criterion provided in Eq (3). In this way, the original threshold-based selection method can be converted into binary classification problems for determining if a sample is clean or not. Specifically, given a sample (x_i, y_i) from $\mathcal{D}$, we can use the dual LoRAs as a binary classifier to detect label noise. If the classifier makes a positive prediction of x_i, the sample is classified as clean.

To optimize the noisy label detector, we need to create positive and negative training samples for binary classification, *i.e.*, correctly and incorrectly labeled samples. Inspired by the work (Kim et al., 2019) on negative learning, for each text $x_i \in \mathcal{D}$, we randomly flip its class label y_i to one of the other classes, *i.e.*, $y_i^n \in \{1, \ldots, K\} \backslash \{y_i\}$, to construct negative datasets D_n. For positive datasets, we first use LLMs (*i.e.*, GPT-4o) to generate pseudo-labels for each text $x_i \in \mathcal{D}$, then select samples where the generated pseudo-labels match the original labels y_i to construct the positive datasets D_p. Here, we treat the samples in D_p as positive samples and the samples in D_n as negative samples (see Appendix A for details). After that, the optimization objective can be calculated as follows:

$$\mathcal{L}_{Detector} = \frac{1}{N_p} \sum_{i=1}^{N_p} \ell_{nll}(p_i^c, y_i) + \ell_{nll}(1 - p_i^c, y_i^n), \tag{6}$$

where N_p denotes the size of D_p, $D_p \subseteq D$, $\ell_{nll}(\cdot, \cdot)$ is the negative log-likelihood loss, defined as $\ell_{nll}(p_i, y) = -\log p_{iy}$.

Although the design of $\mathcal{L}_{LoRA}$ and $\mathcal{L}_{Detector}$ aim to help the noisy label detector perform sample selection, they focus primarily on the comparison between different LoRAs (*i.e.*, Δw_c and Δw_n). To enhance the noisy label detector's ability to learn task-specific representations, we introduce the cross-entropy loss $\mathcal{L}_{ce} = -\frac{1}{N} \sum_{i=1}^{N} \log f(x_i, w_0 + \Delta w_c + \Delta w_n)$. It is worth noting that the optimization objective $\mathcal{L}_{warm} = \mathcal{L}_{ce} + \mathcal{L}_{LoRA}$ is computed in the warm-up stage, the optimization objective $\mathcal{L} = \mathcal{L}_{ce} + \mathcal{L}_{LoRA} + \mathcal{L}_{Detector}$ is computed after warm-up. With these objectives, Delora effectively identifies noisy samples using learned thresholds.

4.2 Training Classifier Model after Selection

After obtaining the selected clean and noisy samples, we introduce the classifier model training stage to learn a classifier model using selected samples. For selected clean samples, we directly utilize the cross-entropy loss to learn from them. For selected noisy samples, we refine and leverage them following the recent work (Yuan et al., 2024), rather than discarding them as done in most previous works (Qiao et al., 2022; Kim et al., 2024). Specifically, we leverage clean samples to construct demonstrations prompting GPT-4o to relabel noisy samples. Then, we utilize the robust loss function to learn from the relabeled noisy samples. See Appendix B for more details. Notably, the clean and noisy samples obtained through Eq. (3) in the last stage, which enables the second stage to be broadly applicable to various classifier models (pre-trained language models or open-source LLMs) and fine-tuning paradigms (full fine-tuning or PEFT), regardless of their backbone architectures. We validate the versatility of Delora in Section 5.5.

5 Experiments

5.1 Experimental Settings

Synthetic Datasets. We first fully evaluate our approach on five text classification benchmarks by synthesizing noisy labels with a variety of noise types and ratios. Specifically, the experiments are evaluated on the following benchmark datasets: Trec (Li and Roth, 2002), SST-2 (Socher et al., 2013), SST-5 (Socher et al., 2013), 20ng (Lang, 1995), AGNews (Gulli, 2005). Following the previous experimental setup (Qiao et al., 2022; Yuan et al., 2024), we then artificially introduce the noise by using three different strategies: (1) **Symmetric Noise (S)** chooses one of the other classes at random to replace the label; (2) **Asymmetric Noise (A)** carries out pairwise label flipping, in which a class i can only change to the following class (i mode K) + 1; (3) **Instance-dependent Noise (I)** alters labels according to the transition probability determined by the related attributes of the instance. For all kinds of noise, the noise rate is set to 20% and 40%. See Appendix C for more details.

Real-World Datasets. We further evaluate our approach on three real-world datasets with noisy labels: Yorùbá(Hedderich et al., 2020), Hausa (Hedderich et al., 2020), AlleNoise (Raczkowska et al., 2024). See Appendix D for more details.

Baselines for Noisy Label Detection. To justify the effectiveness of Delora in detecting noisy labels, we first compare our approach with two other sample selection strategies: (1) **LLMs-detection** strategy, which directly determines whether a given sample is clean using the LLM (*i.e.*, GPT-4o). (2) **Small-loss** strategy, which selects some samples

Dataset	Method	20%S		40%S		20%A		40%A		20%I		40%I	
		Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
Trec	LLMs-detection	70.37	70.31	70.16	70.20	69.96	69.97	69.04	68.82	-	-	-	-
	Small-loss	81.15	88.55	60.16	87.82	81.53	74.02	59.41	96.20	-	-	-	-
	Delora (Ours)	**99.47**	**95.30**	**99.28**	**96.44**	**99.19**	**98.06**	**99.12**	**97.27**	-	-	-	-
SST-5	LLMs-detection	60.91	60.71	59.93	59.92	59.80	59.85	59.46	59.38	67.11	65.47	65.56	67.58
	Small-loss	80.83	79.35	59.83	78.45	79.83	77.85	59.26	76.63	80.01	74.03	60.11	73.96
	Delora (Ours)	**98.11**	**86.75**	**96.37**	**93.59**	**97.18**	**94.90**	**94.04**	**92.61**	**95.31**	**85.02**	**91.99**	**89.29**
SST-2	LLMs-detection	78.94	79.62	79.41	78.78	79.53	79.31	77.88	77.53	85.97	85.08	84.58	85.64
	Small-loss	80.87	85.78	59.75	80.80	80.07	85.48	61.20	68.34	80.59	74.95	61.27	81.78
	Delora (Ours)	**99.97**	**89.07**	**99.95**	**86.34**	**99.96**	**88.60**	**99.76**	**86.75**	**98.35**	**87.27**	**96.43**	**88.86**

Table 1: We compare the Precision (%) and Recall (%) of Delora with LLMs-detection and small-loss to evaluate the performance of noisy label detection (*i.e.*, clean sample selection performance). **Bold** means the best score.

Model	Trec				SST-5						SST-2					
	20%**S**	40%**S**	20%**A**	40%**A**	20%**S**	40%**S**	20%**A**	40%**A**	20%**I**	40%**I**	20%**S**	40%**S**	20%**A**	40%**A**	20%**I**	40%**I**
Base (Clean)	98.60				58.05						97.03					
Base	95.20	90.20	94.20	87.40	54.08	49.59	54.81	47.70	53.07	46.76	86.43	64.62	86.70	65.88	83.85	63.15
LLM-base	71.35				70.51						91.51					
Co-Teaching	95.51	90.98	95.32	89.24	53.99	49.72	55.07	47.24	52.63	46.45	87.29	67.21	89.69	69.60	85.59	67.16
SENT	95.49	91.25	95.43	90.53	54.05	49.61	55.17	47.68	53.70	46.94	87.46	67.17	89.12	69.10	85.38	66.23
LAFT	95.42	91.28	94.43	90.32	55.00	49.13	54.50	47.69	52.57	47.02	88.07	67.39	89.72	68.80	85.45	66.34
SelfMix	96.21	90.52	95.24	90.80	53.63	49.61	55.80	47.59	53.00	46.66	87.58	66.78	89.97	68.92	85.61	66.05
CleaR	96.01	90.45	95.35	90.69	53.69	49.97	54.95	47.63	53.64	46.62	87.21	66.81	89.36	69.43	85.13	66.54
NoiseAL	97.30	96.54	96.96	95.95	55.00	50.48	54.94	48.12	54.00	47.32	91.90	86.25	91.97	86.72	91.55	85.01
Delora (Ours)	**98.46**	**97.60**	**98.30**	**97.40**	**57.39**	**55.62**	**57.57**	**55.39**	**57.02**	**55.02**	**96.50**	**95.75**	**96.27**	**95.18**	**96.08**	**95.00**

Table 2: Performance (test accuracy %) comparison of Delora with other LNL baselines on synthetic noise datasets. Base (Clean) refers to the base model trained on ground truth data without noisy labels. LLM-base refers to directly using LLMs (GPT-4o) on the test dataset. **Bold** means the best score for each dataset.

with small training losses as clean via the Gaussian Mixture Model, appearing in most LNL works.

Baselines for LNL. Then, we further compare Delora with previous sample selection-based LNL baselines as follows: (1) Base model (Llama3.1-8B-Instruct (Dubey et al., 2024)) without noise-handling; (2) Some methods that use small-loss strategy: **Co-Teaching** (Han et al., 2018), **SelfMix** (Qiao et al., 2022), **NoiseAL** (Yuan et al., 2024), **CleaR** (Kim et al., 2024); (3) Other technologies : **SENT**(Wang et al., 2022b), **LAFT** (Wang et al., 2023). See Appendix E for more details.

Evaluation Metrics. For the first stage (noisy label detection stage), we evaluate the selection of clean samples using precision and recall metrics. A higher recall suggests that more clean samples are found in the noisy dataset, whereas a higher precision means that there are more real clean examples in D_c. For the last stage (classifier model training stage), we use the test accuracy to evaluate the generalization performance of our framework on text classification in the context of noisy environments.

The implementation details are in Appendix F.

5.2 Performance for Noisy Label Detection

Table 1 presents the performance for noisy label detection on three synthetic datasets. The results presented in Table 1 show that our proposed noisy label detector outperforms the comparison strategies across all dataset settings, demonstrating significant improvements in precision and recall. The LLMs-detection strategy directly leverages the

zero-shot ability of LLMs to perform binary classification (clean or noisy). The lack of targeted demonstrations in LLM prompts has resulted in lower selection performance for this strategy. In contrast, the small-loss strategy and Delora make better use of training samples to perform better in precision and recall. Moreover, leveraging the different LoRA modules to memorize the information of clean and noisy samples separately, Delora surpasses the small-loss strategy in detecting noisy labels, particularly under severe noise settings and fine-grained classification datasets. For instance, in datasets with a noise rate of 40%, the small-loss strategy performs poorly in precision, while our method performs well in various situations. Additionally, Delora reduces the need to manually set loss thresholds, making it a practical and effective approach to detect label noise in real-world tasks.

Method	Hausa	Yorùbá	AlleNoise
Noise Ratio	50.37%	33.28%	15.00%
Base	49.80±0.26	67.02±0.32	65.75±0.25
Co-Teaching	46.47±0.38	66.23±0.69	65.32±0.36
SENT	46.15±0.57	66.21±0.24	66.66±0.38
LAFT	50.56±0.31	69.13±0.54	66.57±0.81
SelfMix	50.81±2.62	69.27±1.07	67.67±3.67
CleaR	51.66±1.14	70.11±1.10	67.73±2.40
NoiseAL	52.34±0.69	72.13±0.24	69.80±0.62
Ours	**60.12**±0.22	**78.56**±0.30	**76.28**±0.23

Table 3: Main results on real-world noise datasets

5.3 Performance for LNL

Table 2 and Table 3 show the main results for three synthetic and real-world noisy datasets (more results in Appendix G). From these results, we found that (1) Delora significantly outperforms all baselines on synthetic datasets with varying noise types and ratios, which validates the effectiveness of our proposed two-stage decoupling framework in addressing the LNL task. (2) For fine-grained classification datasets, while most compared methods show little improvement on SST-5 due to its fine-grained nature, Delora shows substantial improvement. (3) For real-world datasets with a high noise rate, previous methods show limited improvement on Hausa. In contrast, Delora achieves a significant performance boost, showcasing its ability to combat label noise in practical situations.

5.4 Ablation Studies

We perform ablation studies to investigate the contributions of each component and routing strategy in Delora. Table 4 presents the ablation results.

The effect of noisy label detector. Noisy Label Detector (NLD) effectively identifies noisy samples and partitions the noisy dataset, which can alleviate the overfitting of noisy labels and the issue of vicious cycles in the subsequent classifier model training stage. The performance of Delora will decrease greatly when we remove the NLD, which indicates that utilizing the NLD is crucial.

The effect of classifier model training. The classifier model training stage (CT) leverages the selected samples to fine-tune the classifier model, obtaining the best classification performance. If we remove CT and only rely on the trained clean LoRA to perform classification tasks, the performance of Delora will decrease by a large margin, which emphasizes the necessity of CT in Delora.

The effect of different optimization objectives. In our experiments, we combine different optimization objectives ($\mathcal{L}_{LoRA}, \mathcal{L}_{Detector}, \mathcal{L}_{ce}$) to train the noisy label detector. The results in Table 4 demonstrate that each component is essential for improving generalization and robustness, failing to employ them leads to a decline in the results.

The effect of selected noisy samples. To maximize the use of training data without discarding remaining noisy samples (NS), we relabel and repurpose them to further train the classifier model. This process has also been proven to be crucial, as its removal leads to substantial performance drops.

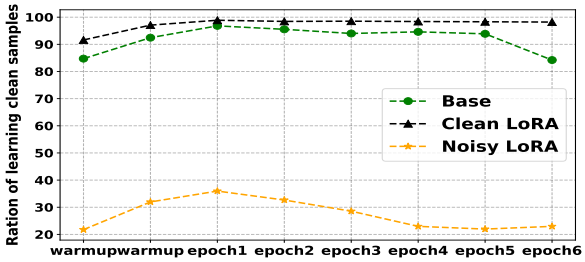
Variant	Sym		Asym	
	20%	40%	20%	40%
Delora (Ours)	**98.46**	**97.60**	**98.30**	**97.40**
w/o NLD	95.20	90.20	94.20	89.40
w/o CT	96.02	91.01	94.77	90.08
w/o $\mathcal{L}_{LoRA}$	96.46	92.00	95.68	90.43
w/o $\mathcal{L}_{Detector}$	96.91	91.98	95.07	90.75
w/o $\mathcal{L}_{ce}$	96.54	91.33	95.44	90.35
w/o NS	97.41	94.29	96.97	91.53

Table 4: Ablation study on the Trec dataset.

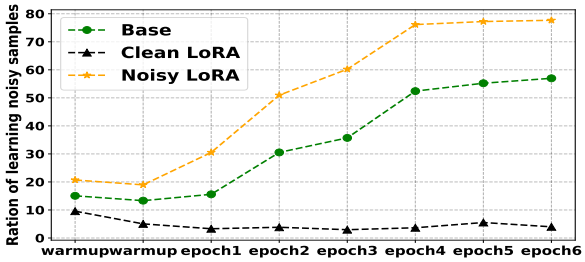
5.5 Analysis

Memorization performance for different LoRAs. In our proposed Delora, we adopt the clean and noisy LoRA to memorize the information of clean and noisy samples separately. To confirm whether different LoRA modules fit clean and noisy samples, we compare the ratio of memorizing clean and noisy samples for different LoRAs during fine-

tuning. As shown in Figure 3, we observe that the base model first memorizes clean samples and then gradually fits noisy samples, which is consistent with the memory effect. For clean LoRA, it enhances the memorization of clean samples while reducing the memorization of noisy samples. In contrast, noisy LoRA exhibits the opposite effect, the parameters in noisy LoRA are restricted to absorb the side effects of mislabeled data. These results indicate that Delora effectively constrains different LoRA modules to memorize clean and noisy samples separately, thereby facilitating the successful construction of the noisy label detector.

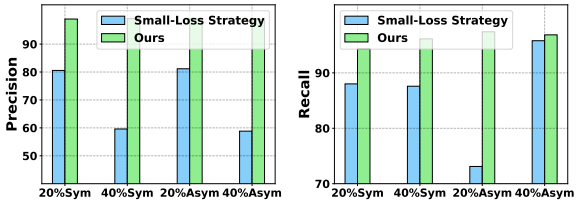


(a) Impact on learning clean samples



(b) Impact on learning noisy samples

Figure 3: Memorization performance of different LoRAs during fine-tuning on Trec under 20%A. The green line refers to the base model without noise handling.



(a) Precision On Trec (b) Recall On Trec

Figure 4: Precision and recall of noisy label detection with BERT as the backbone on the Trec dataset.

Impact of Noisy Label Detector Backbones. Table 1 has demonstrated the superiority of Delora in detecting noisy labels based on Llama 3.1-8B. To evaluate the impact of different language model backbones on noise detection, we use the small language model BERT as the backbone and report the results in Figure 4. As shown in Figure 4, our method maintains consistently superior performance, underscoring its resilience across different backbones. We analyze how our method consistently outperforms others across different language models. Generally, for LNL, incorrect labels can propagate errors during backpropagation, causing adjustment bias across all trainable parameters. Delora effectively mitigates this issue by ensuring that the influence of incorrect labels is restricted to the noise-specific parameters within the noise LoRA, thereby preserving the integrity of the clean LoRA, which is dedicated to learning from clean data. This solution achieves functional segregation at the parameter level, which is architecture-independent.

Delora for Various Classifier Models. The results in Table 4 highlight the crucial role of the classifier model training stage in our method. Notably, Delora seamlessly integrates with various classifier models at this stage. To showcase the versatility of our proposed LNL framework, we conduct experiments on Trec using a diverse set of models, including BERT (full fine-tuning), Llama 3.2-3B (Dubey et al., 2024), Gemma 2-9B (Rivière et al., 2024), Llama 3.1-8B. As shown in Table 5, Delora exhibits the best performance on average.

Architecture	BERT	Llama3.2	Gemma2	Llama3.1
Base	93.60	93.88	94.49	94.20
Co-Teaching	94.88	94.81	95.74	95.32
SelfMix	95.16	96.05	95.12	95.24
NoiseAL	96.80	96.23	97.14	96.96
Delora	**97.40**	**97.50**	**98.12**	**98.30**

Table 5: Test accuracy (%) using various classifier models on Trec under 20%A. **Bold** means the best score.

Analysis of Efficiency. In this part, we explore the efficiency of our proposed methods. Although the proposed methods introduce additional trainable parameters, we perform the systematic multi-dimensional efficiency comparisons (see the Table 6, from the perspective of final classification performance) to demonstrate that our method achieves a superior Pareto frontier in the accuracy-parameters-memory trade-off space. Specifically, from the Table 6, there are some key findings as follows:

(1) Compared to the base model with the standard single LoRA, we require only +13.6 MB parameters and +3.2 GB memory but achieve +3.26%, +7.40%, +4.10%, and +10% accuracy on the Trec dataset with different noisy settings, respectively.

(2) Compared to other baselines (SelfMix and NoiseAL), our proposed method outperforms them

Methods	Parameter (M)	Memory (GB)	20%S	40%S	20%A	40%A
standard single LoRA	13.7	10.4	95.20	90.20	94.20	87.40
SelfMix	28.3	19.3	96.21 (+1.01)	90.52 (+0.32)	95.24 (+1.04)	90.80 (+3.40)
NoiseAL	28.4	21.7	97.30 (+2.10)	96.54 (+6.34)	96.96 (+2.76)	95.95 (+8.55)
Ours	27.3	13.6	98.46 (+3.26)	97.60 (+7.40)	98.30 (+4.10)	97.40 (+10.00)

Table 6: Detailed results for Trec datasets with the LLaMA 3.1 8B as backbone.

Methods	20%S	40%S	20%A	40%A
Single LoRA → SelfMix	+0.07% / +1M params	+0.02% / +1M params	+0.07% / +1M params	+0.23% / +1M params
Single LoRA → NoiseAL	+0.19% / +1M params	+0.56% / +1M params	+0.24% / +1M params	+0.76% / +1M params
Single LoRA → Ours	+0.24% / +1M params	+0.54% / +1M params	+0.30% / +1M params	+0.74% / +1M params

Table 7: Parameter efficiency on Trec datasets with different noisy settings.

Methods	20%S	40%S	20%A	40%A
Single LoRA → SelfMix	+0.11% / +1G memory	+0.04% / +1G memory	+0.12% / +1G memory	+0.38% / +1G memory
Single LoRA → NoiseAL	+0.14% / +1G memory	+0.43% / +1G memory	+0.19% / 1G memory	+0.58% / +1G memory
Single LoRA → Ours	+1.02% / +1G memory	+2.31% / +1G memory	+1.28% / +1G memory	+3.13% / +1G memory

Table 8: Memory efficiency on Trec datasets with different noisy settings.

with relatively fewer resources. Moreover, we further compare our method with other baselines by quantifying parameter efficiency via accuracy gain per parameter (ΔAcc/ΔParams) (see the Table 7) and memory efficiency via accuracy gain per parameter (ΔAcc/ΔMemory) (see the Table 8). In the above experiments, we chose the LLaMA 3.1 8B as the backbone.

In general, while the dual LoRA modules introduce additional parameters, our systematic analysis of parameter efficiency (ΔAcc/ΔParams) and memory efficiency (ΔAcc/ΔMemory) demonstrates that the increased parameterization is justified. Moreover, through comparative analysis across different methods, we quantify the accuracy gain per additional 1M parameters or 1GB memory, ultimately proving that our approach achieves a superior Pareto frontier in the three-dimensional trade-off space of accuracy, parameter count, and memory usage.

6 Conclusion

In this work, we mitigate the issues of the vicious cycle in current mainstream sample selection methods and further explore the PEFT methods in the era of LLMs to solve the LNL tasks. Specifically, we decouple this task into sample selection and classifier model training. For sample selection, we introduce the dual LoRA to construct a new noisy label detection approach, which restricts the clean LoRA to fit clean data and the noisy LoRA to absorb the side effects of mislabeled data. Then we can train the robust classifier model by leveraging the carefully selected samples. Extensive experiments have convincingly demonstrated the superior performance of Delora in both noisy label detection and text classification. Moreover, our in-depth analysis has demonstrated that Delora can generalize to other language models.

Limitations

While Delora has demonstrated significant improvements in text classification tasks, there are some limitations to consider in the following aspects: (1) Due to resource constraints, we have not evaluated our framework on larger language models, such as Llama-3.2 70B. (2) The experiments in this paper are limited to the text classification task and do not explore other tasks (Wang et al., 2022a), such as text generation tasks. Interestingly, we discovered a recent study (Luo et al., 2024) that extends the NoiseAL method, originally designed for text classification, to text generation tasks. This opens up new perspectives and potential improvements for our work. We leave the exploration of this direction as promising future work.

References

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron C. Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 233–242. PMLR.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Chen Feng, Georgios Tzimiropoulos, and Ioannis Patras. 2024. Clipcleaner: Cleaning noisy labels with CLIP. In *Proceedings of the 32nd ACM International Conference on Multimedia, MM 2024, Melbourne, VIC, Australia, 28 October 2024 - 1 November 2024*, pages 876–885. ACM.

Aritra Ghosh, Himanshu Kumar, and P. S. Sastry. 2017. Robust loss functions under label noise for deep neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 1919–1925. AAAI Press.

Antonio Gulli. 2005. The anatomy of a news search engine. In *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005 - Special interest tracks and posters*, pages 880–881. ACM.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W. Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8536–8546.

Michael A. Hedderich, David Ifeoluwa Adelani, Dawei Zhu, Jesujoba O. Alabi, Udia Markus, and Dietrich Klakow. 2020. Transfer learning and distant supervision for multilingual transformer models: A study on african languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2580–2591. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yeachan Kim, Junho Kim, and SangKeun Lee. 2024. Towards robust and generalized parameter-efficient fine-tuning for noisy label learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 5922–5936. Association for Computational Linguistics.

Youngdong Kim, Junho Yim, Juseung Yun, and Junmo Kim. 2019. NLNL: negative learning for noisy labels. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 101–110. IEEE.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pages 331–339. Morgan Kaufmann.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*.

Chang Liu, Viraj Shah, Aiyu Cui, and Svetlana Lazebnik. 2024. Unziplora: Separating content and style from a single image. *CoRR*, abs/2412.04465.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 61–68. Association for Computational Linguistics.

Junyu Luo, Xiao Luo, Kaize Ding, Jingyang Yuan, Zhiping Xiao, and Ming Zhang. 2024. Robustft: Robust supervised fine-tuning for large language models under noisy response.

Xiangwei Lv, Guifeng Wang, Jingyuan Chen, Hejian Su, Zhiang Dong, Yumeng Zhu, Beishui Liao, and Fei Wu. 2025. Debiased cognition representation learning for knowledge tracing. *ACM Transactions on Information Systems*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Dan Qiao, Chenchen Dai, Yuyang Ding, Juntao Li, Qiang Chen, Wenliang Chen, and Min Zhang. 2022. Selfmix: Robust learning against textual label noise with self-mixup training. In *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, pages 960–970. International Committee on Computational Linguistics.

Alicja Raczkowska, Aleksandra Osowska-Kurczab, Jacek Szczerbinski, Kalina Jasinska-Kobus, and Klaudia Nazarko. 2024. Allenoise - large-scale text classification benchmark dataset with real-world label noise. *CoRR*, abs/2407.10992.

Morgane Rivière, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozinska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucinska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju-yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjösund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, and Lilly McNealus. 2024. Gemma 2: Improving open language models at a practical size. *CoRR*, abs/2408.00118.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 1917–1928.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL.

Lidong Wang, Yin Zhang, and Keyong Hu. 2022a. FEUI: fusion embedding for user identification across social networks. *Appl. Intell.*, 52(7):8209–8225.

Song Wang, Zhen Tan, Ruocheng Guo, and Jundong Li. 2023. Noise-robust fine-tuning of pretrained language models via external guidance. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 12528–12540. Association for Computational Linguistics.

Zhihao Wang, Zongyu Lin, Junjie Wen, Xianxin Chen, Peiqi Liu, Guidong Zheng, Yujun Chen, and Zhilin Yang. 2022b. Learning to detect noisy labels using model-based features. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5796–5808. Association for Computational Linguistics.

Bo Yuan, Yulin Chen, Yin Zhang, and Wei Jiang. 2024. Hide and seek in noise labels: Noise-robust collaborative active learning with llms-powered assistance. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 10977–11011. Association for Computational Linguistics.

Zhilu Zhang and Mert R. Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8792–8802.

A Construction of Positive and Negative Sample

Given the original dataset $\mathcal{D}$, for each text $x_i \in \mathcal{D}$, we randomly flip its class label y_i to one of the other classes, *i.e.*, $y_i^n \in \{1, \ldots, K\} \setminus \{y_i\}$, to construct negative datasets D_n. For texts and their corresponding labels in D_n, we regard them as negative samples, which train the noisy label detector in such a logical form: "input text does not belong to this complementary label."

For positive datasets, we first use LLMs (*i.e.*, GPT-4o) to generate pseudo-labels for each text $x_i \in \mathcal{D}$, subsequently select samples where the pseudo-labels match the original labels y_i to construct the positive datasets D_p. Here, we treat the samples in D_p as positive samples, which train the noisy label detector in such a logical form: "input text belongs to this label." The details of LLM prompts are in Appendix H.

B Learning From Clean samples and Noisy samples

In this part, we present the details of the classifier model training to learn from clean samples and noisy samples. Denote $f(x) \in \mathbb{R}^K$ as the output of the classifier model f, where K is the number of classes. The confidence of x for each class $k \in \{1, \ldots, K\}$ can be represented as follows: $p(k; x) = \frac{e^{f(k;x)}}{\sum_{k=1}^{K} e^{f(k;x)}}$.

Learning from Clean Samples. For the selected clean samples $x_i \in \mathcal{D}_c$, we directly utilize the cross-entropy loss for the classifier model:

$$\mathcal{L}_{\mathcal{D}_c} = -\frac{1}{N} \sum_{i=1}^{N_{\mathcal{D}_c}} \log p(y_i; x_i) \tag{7}$$

where $\mathcal{D}_c$ is the selected clean subsets, $N_{\mathcal{D}_c}$ denotes the size of $\mathcal{D}_c$, and N denotes the size of the entire dataset.

Learning from Noisy Samples. After selecting clean samples from $\mathcal{D}$ by the noisy label detector, the remaining samples are considered noisy samples. To maximize the utilization of training data, we construct the demonstrations via clean samples and leverage the strong in-context learning ability of LLMs (GPT-4o) to generate new labels for these noisy samples. Then, we put these corrected samples in the correction subsets $\mathcal{D}_o$. With the help of LLMs (GPT-4o), the number of noisy samples has greatly decreased. However, even the most powerful LLM cannot generate the right labels for each noisy sample. To learn from the corrected noisy samples in $\mathcal{D}_o$, we resort to the reversed cross-entropy loss function. This loss function has a noise-robust property, which can let us optimize the classifier model given a dataset with a lower noise ratio (Ghosh et al., 2017; Zhang and Sabuncu, 2018; Yuan et al., 2024). Specifically, we utilize the reversed cross-entropy loss for sample (x_i, y_i) in $\mathcal{D}_o$:

$$\mathcal{L}_{\mathcal{D}_o} = -\frac{1}{N} \sum_{i=1}^{N_{\mathcal{D}o}} \sum_{k=1}^{\mathcal{K}} p(k; x_i) \log q(k|x_i), \tag{8}$$

where $q(k|\boldsymbol{x})$ is the ground-truth distribution over labels, $N_{\mathcal{D}_o}$ denotes the size of $\mathcal{D}_o$.

Overall Learning Objectives. Finally, we train the classifier model on selected clean samples and relabeled noisy samples by: $\mathcal{L} = \mathcal{L}_{\mathcal{D}_c} + \mathcal{L}_{\mathcal{D}_o}$.

C Detailed Process for Generating Noisy Labels

High-quality data is typically crucial. However, in real-world scenarios, collected data often contains biases (Lv et al., 2025) and noise. We have simulated such non-ideal conditions in our experiments. Specifically, in our evaluative experiments, we first select the following datasets: **Trec**, **SST-2**, **SST-5**, **20ng**, and **AGNews**. Then we synthesize noisy labels with a variety of noise types and ratios for these datasets. When the noise ratio $\varepsilon \in [0, 1)$ is given, we explain the details of synthetic noise generation processes in the following:

Symmetric noise. Symmetric noise chooses one of the other classes at random to replace the label. Each class has the same probability of incorrectly flipping to any other class. To generate this noise, we establish the noise transition matrix $T \in R^{K \times K}$, where K represents the number of classes. We then modify the elements in the noise transition matrix according to the following formula:

$$T_{ij} = \begin{cases} \varepsilon, & i = j \\ \frac{1-\varepsilon}{k-1}, & i \neq j \end{cases},$$

where i and j respectively represent the horizontal and vertical axes in the noise transition matrix. Lastly, we flip the labels in training samples based on the probability in the matrix.

Asymmetric noise. Asymmetric noise carries out pairwise label flipping, in which a class i can

only change to the following class (i mode K) + 1. That is to say, similar classes are mistakenly flipped between each other. To generate this noise, we establish the noise transition matrix $T \in R^{K \times K}$, where K represents the number of classes. We then modify the elements in the noise transition matrix according to the following formula:

$$T_{ij} = \begin{cases} \varepsilon, & i = j \\ 1 - \varepsilon, & i = j + 1 \text{ (mod } K) \\ 0, & \text{otherwise} \end{cases}$$

where i and j respectively represent the horizontal and vertical axes in the noise transition matrix. Lastly, we flip the labels in training samples based on the probability in the matrix.

Instance-dependent noise. Instance-dependent noise alters labels according to the transition probability determined by the related attributes of the instance. The generation of such noise is affected by text features, which is more consistent with the noise generation process in the real world, and more challenging. We follow previous works (Yuan et al., 2024) for instance-dependent noise generation. The detailed algorithm of noisy label generation is summarized in Algorithm 1.

D Details of Real-World Datasets with Noisy Labels

In our evaluative experiments, we also select the following real-world datasets with noisy labels: **Hausa**, **Yorùbá**, and **AlleNoise**. **Yorùbá** and **Hausa** are low-resource African languages with five and seven categories, respectively, in their text categorization datasets. With a level of 33.28% for the latter and 50.37% for the former, they both incorporate real-world noise. **AlleNoise** comprises 502310 brief texts categorized by 5692 types. Due to incorrectly categorized data points, there is a 15% noise level in it.

Table 9 introduces detailed statistics about all datasets used in our experiments.

E Details of Baselines

In our evaluative experiments, we compare our Delora with the following LNL methods:

Basic models. We train the LLaMA-3.1-8B-Instruct only with standard cross-entropy loss without noise handling.

Co-Teaching (Han et al., 2018). Co-Teaching trains two models simultaneously and lets them instruct one another using each mini-batch.

#Dataset	#Class	#Training	#Validation	#Test
Trec	6	4952	500	500
20ng	20	9051	7527	2263
AGNews	4	112400	7600	7600
SST-2	2	5099	1820	1820
SST-5	5	8544	1101	2210
Hausa	5	2045	290	582
Yorùbá	7	1340	189	379
AlleNoise	5692	400k	50k	50k

Table 9: The detailed statistics of all datasets used in our experiments.

SelfMix (Qiao et al., 2022). SelfMix uses the Gaussian Mixture Model to split samples and semi-supervised learning to manage label noise.

SENT(Wang et al., 2022b). SENT transfers the noise distribution to a clean set and trains a model to distinguish noisy labels from clean ones using model-based features.

LAFT (Wang et al., 2023). LAFT examines the possibility of using supervision data—such as confidence scores—produced by ChatGPT to address the noisy label issue in pre-trained language model fine-tuning.

NoiseAL (Yuan et al., 2024). NoiseAL is a novel framework that introduces active learning to combine the non-pretrained model (BiLSTM), pre-trained language model (BERT), and LLMs for learning from noisy labels.

CleaR (Kim et al., 2024). CleaR is a new PEFT technique that minimizes the impact of noisy data while adaptively activating the PEFT modules to enhance generalization ability.

F Implementation Details and Setups

In this section, we detail to implement the baselines and our Delora.

For Delora and all baselines, we report the average performance on 5 different seeds considering their stochasticity. In the main experiments, we chose the LLaMA-3.1-8B-Instruct as the backbone model for Delora (both the noisy label detection stage and the model training stage) and other baseline methods, the LLaMA was fine-tuned by using LoRA.

LoRA implementation. For Delora, we set the bottleneck deminsion r for the clean LoRA and noisy LoRA as 32. For these two LoRAs, we only apply LoRA weights on query and value attention weights.

Algorithm 1 Instance Dependent Noise Generation

Input: Clean samples $(x_i, y_i)_{i=1}^n$, $y_i \in [1, \dots, k]$; Noisy ratio ε;

Train an LSTM classifier f;
Get output from an LSTM classifier $f_{x_i} \in \mathbb{R}^k$ for all $i = 1, \dots, n$;
Set $N_{noisy} = 0$;
while $N_{noisy} < n \times \varepsilon$ **do**
    Randomly choose a sample x_i, $argmax(softmax(f_{x_i})) \neq y_i$;
    set its noisy label $\bar{y_i} = argmax(softmax(f_{x_i}))$;
    $N_{noisy} = N_{noisy} + 1$;
end while

Output: Noise samples $(x_i, \bar{y_i})_{i=1}^n$;

Hardware Details. We train our framework on Nvidia RTX 3090 and Nvidia A100 GPU. We utilize mixed precision training (Micikevicius et al., 2018) to expedite the training procedure. All the implementations are performed with Python, PyTorch, and HuggingFace.

Hyper-parameters. In order to strike a balance between effectiveness and efficiency, we set 8 training epochs for the noisy label detection stage, and 6 training epochs for the classifier model training stage. Moreover, in the noisy label detection stage, we perform model warm-up for 2 epochs on all datasets. We select the batch size from $[16, 32]$, and sweep the learning rates in $[1e-4, 2e-4, 3e-4, 4e-4, 5e-4]$ for Delora. The selection of hyper-parameters is selected according to the performance on a clean development set.

In our work, our proposed Delora is a two-stage framework consisting of a noisy label detection stage and a classifier model training phase. The pseudo-code is presented in Algorithm 2.

G More detailed Results

To further demonstrate the broad applicability of our proposed method, we have evaluated the proposed methods on the 20ng and AGNews datasets. Table 10 shows the evaluation results for noisy label detection. Table 11 shows the evaluation results for text classification.

H Construction of Prompt

In this section, we list the prompt used in our experiments. To optimize the noisy label detector, we leverage the LLM in the construction of positive datasets. The prompt is as follows:

Zero-Shot-CoT Prompt

Below is a text classification problem. Note that you can only select the label in {options}. Let's think step by step and give your answer.
SENTENCE: {text}
LABEL:

Then, in the training of classifier models, we use the selected clean samples to construct demonstrations, prompting LLM to generate the new labels via Few-Shot-CoT. The prompt is as follows:

Few-Shot-CoT Prompt

Below is a text classification problem. Note that you can only select the label in {options}. Let's think step by step and give your answer.
SENTENCE: {text1}
LABEL: {label1}
SENTENCE: {text2}
LABEL: {label2}
...
SENTENCE: {text}
LABEL:

I More Detailed Analysis

I.1 Results under Extreme Noise Conditions.

To further validate the robustness of the proposed methods under extreme noise conditions, we evaluate our model on Trec datasets under large noise ratios. Table 12 reports the precision and recall for noise detection, where Delora exhibits a robust capability in identifying severe noise. Table 13 reports the test accuracy for text classification, where Delora exhibits a robust capability in classifying

Dataset	Method	20%S		40%S		20%A		40%A		20%I		40%I	
		Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
20ng	LLMs-detection	97.97	59.80	98.17	58.97	98.40	59.62	98.17	58.88	98.63	59.75	97.76	59.46
	Small-loss	81.71	86.50	63.02	83.78	81.07	88.17	61.00	85.97	81.16	86.50	61.59	80.25
	Delora (Ours)	**79.68**	**71.34**	**60.17**	**58.43**	**80.46**	**72.80**	**60.04**	**59.56**	**96.16**	**85.14**	**72.54**	**86.90**
AGNews	LLMs-detection	99.81	63.69	99.31	63.52	99.25	60.34	98.82	60.32	99.45	60.34	98.74	53.69
	Small-loss	80.82	92.24	81.09	90.19	80.95	91.22	80.61	92.24	81.09	90.19	80.95	91.22
	Delora (Ours)	**99.28**	**93.85**	**97.70**	**93.86**	**99.16**	**93.94**	**96.71**	**94.29**	**99.36**	**95.01**	**97.39**	**94.85**

Table 10: We evaluate the performance of noisy label detection (*i.e.*, clean sample selection performance) on the 20ng and AGNews datasets by comparing the precision (%) and recall (%) of Delora with LLMs-detection and small-loss. The best results results are highlighted in **Bold**.

Model	20ng						AGNews					
	20%**S**	40%**S**	20%**A**	40%**A**	20%**I**	40%**I**	20%**S**	40%**S**	20%**A**	40%**A**	20%**I**	40%**I**
Base (Clean)	88.14						96.16					
Base	83.85	72.78	82.98	63.79	81.80	70.12	91.46	88.39	91.71	88.88	91.62	88.57
LLM-base	72.15						84.52					
Co-Teaching	85.35	72.85	83.98	63.94	82.16	71.97	92.03	88.75	92.55	89.38	92.88	88.88
SENT	84.17	73.97	83.80	64.40	82.44	71.56	92.57	89.31	92.37	89.91	92.6	89.45
LAFT	85.64	74.17	83.95	64.37	82.64	71.58	93.09	91.38	93.19	89.82	92.81	90.65
SelfMix	80.87	78.99	78.19	65.52	77.68	70.54	92.22	89.45	92.88	90.65	91.97	89.24
CleaR	84.58	72.89	83.28	64.87	83.58	70.44	92.98	90.30	92.76	89.39	92.46	89.80
NoiseAL	85.95	77.11	85.89	75.79	84.62	75.69	92.20	90.31	93.08	89.02	92.69	90.31
Delora (Ours)	**88.51**	**82.16**	**88.40**	**79.60**	**86.87**	**80.90**	**95.64**	**95.29**	**95.84**	**95.17**	**95.64**	**95.86**

Table 11: Performance (test accuracy %) comparison of Delora with other LNL baselines on synthetic noise datasets (20ng and AGNews datasets). Base (Clean) refers to the base model trained on ground truth data without noisy labels. LLM-base refers to directly using LLMs (GPT-4o) on the test dataset. **Bold** means the best score for each dataset.

texts. Compared to other baselines, our methods show superior performance under high noise ratios.

I.2 Further Analysis for the modules in Noisy Label Detector.

As outlined in Section 4.1, our proposed noisy label detector consists of three key modules: the dual LoRA (the clean and noisy LoRA), a constraint on the parameter update $\Delta\sigma_c$ of the clean LoRA, and a constraint on the parameter update $\Delta\sigma_n$ of the noisy LoRA. In this part, we conduct further

Dataset	Method	60%S		60%A	
		Prec.	Rec.	Prec.	Rec.
Trec	LLMs-detection	60.26	60.32	60.04	60.05
	Small-loss	59.14	73.45	58.58	69.76
	Delora (Ours)	**97.75**	**92.97**	**97.04**	**92.25**

Table 12: We evaluate the performance of noisy label detection (*i.e.*, the clean sample selection performance) on the Trec by comparing the Precision (%) and Recall (%) of Delora with LLMs-detection and Small-loss.

Dataset	Trec							
Noise(↓) / Method(→)	Base	Co-Teaching	SENT	LAFT	SelfMix	CleaR	NoiseAL	Delora
60%S	78.40	80.06	81.28	81.46	81.52	79.93	81.12	**84.53**
60%A	56.20	61.27	62.25	63.68	65.36	65.52	67.21	**71.96**

Table 13: The detailed results (test accuracy %) on Trec datasets. **Bold** means the best score.

Modules			Trec			
dual LoRAs	**constraint on $\Delta\sigma_n$**	**constraint on $\Delta\sigma_c$**	**20%S**	**40%S**	**20%A**	**40%A**
✗	✗	✗	95.20	90.20	94.20	87.40
✔	✗	✗	82.27	77.82	80.96	75.63
✔	✔	✗	96.37	95.21	95.86	95.01
✔	✔	✔	**98.46**	**97.60**	**98.30**	**97.40**

Table 14: Further ablation study for the noisy label detector on the Trec dataset.

ablation studies to elucidate the factors that contribute to the success of our approach. Experiments are performed on Trec datasets, and experimental results are shown in Table 14.

Firstly, we only introduce two LoRA modules and do not constrain their parameter update, it can be seen that the classification performance on the test set is very poor. Since dual LoRA modules are simultaneously influenced by noisy labels during training, they fail to learn distinct representations for clean and noisy samples. As a result, they struggle to effectively identify noisy samples in the noisy label detection phase, ultimately hindering the training of the downstream classifier model. That is to say, simply introducing more LoRA modules itself will not bring any performance improvement and even have negative impacts.

Secondly, if we add the constraint on the parameter update $\Delta\sigma_n$ of noisy LoRA, the test accuracy is significantly improved. This suggests that imposing a constraint on $\Delta\sigma_n$ allows the noisy LoRA to absorb the adverse effects of mislabeled data, enabling the clean LoRA to focus on learning from clean samples more effectively. That is to say, it can help different LoRAs to learn distinct representations for clean and noisy samples.

Thirdly, we introduce an additional constraint on the parameter update $\Delta\sigma_c$ of the clean LoRA. The results show that this further enhances test accuracy, indicating that restricting $\Delta\sigma_c$ helps the clean LoRA retain less mislabeled data, leading to improved performance.

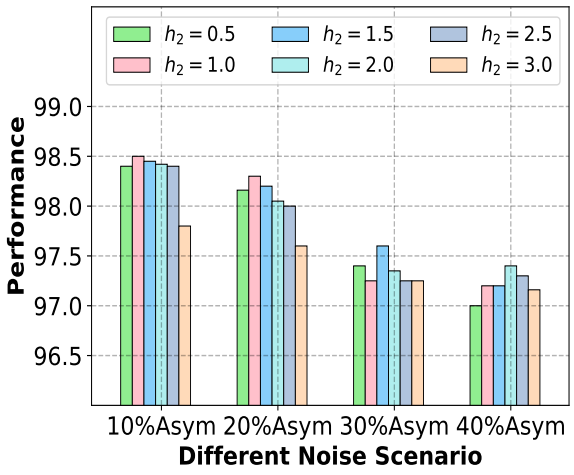
Overall, further studies show that combining the three modules of our proposed noisy label detector can achieve a steady performance improvement, which proves the necessity of each module in the noisy label detector.

I.3 Further Analysis for the Hyper-parameter Setting in Constraint Functions.

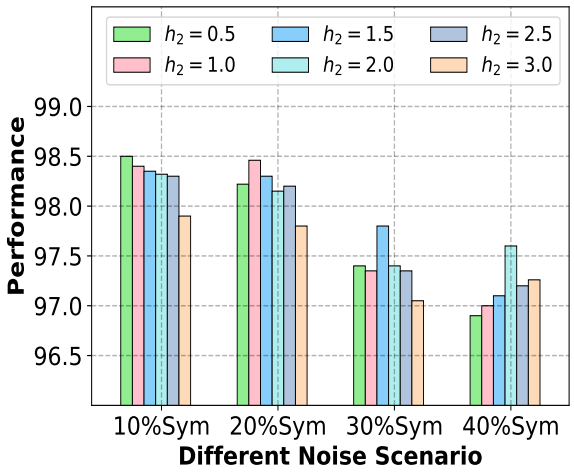
As we mentioned in Section I.2, we add a constraint on the parameter update $\Delta\sigma_c$ of the clean LoRA Δw_c, and a constraint on the parameter update $\Delta\sigma_n$ of the noisy LoRA Δw_n. Specifically, in this paper, we set $\tau_1(t) = t^{h_1}$ and $\tau_2(t) = t^{-h_2}$ to constrain the parameter update of clean LoRA and noisy LoRA, where h_1 and h_2 are two hyper-parameters. Here, we explore the hyper-parameter setting in these two constraint functions.

Hyper-parameter h_2. Specifically, in Figure 5 (a), we compare the performance of different h_2 values under various Asym noise conditions. For each noise setting, we identify the optimal h_2 value and visualize the results in Figure 5 (c). Similarly, in Figure 5 (b), we compare the performance of different h_2 values under various Sym noise conditions. For each noise setting, we identify the optimal h_2 value and visualize the results in Figure 5 (d).

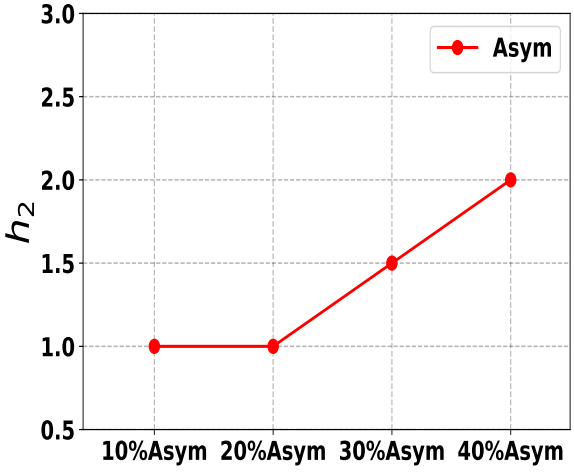
From the results in Figure 5 (c-d), we observe a positive correlation between the optimal h_2 value and the noise ratio. In other words, as the noise ratio increases, a larger h_2 accelerates the updates of Δw_n, allowing the noisy LoRA Δw_n to better



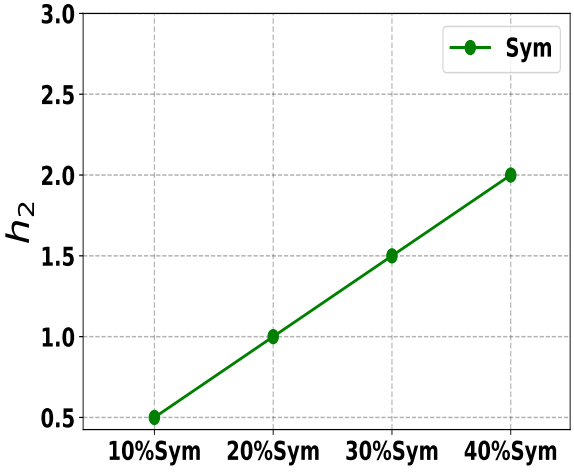
(a) The performance of our methods with different h_2 under different noise scenarios



(b) The performance of our methods with different h_2 under different noise scenarios



(c) Relationship between the optimal h_2 and noise ratio



(d) Relationship between the optimal h_2 and noise ratio

Figure 5: Analysis for the choice of hyper-parameter h_2 under different noise ratios on the Trec datasets.

absorb the impact of mislabeled data.

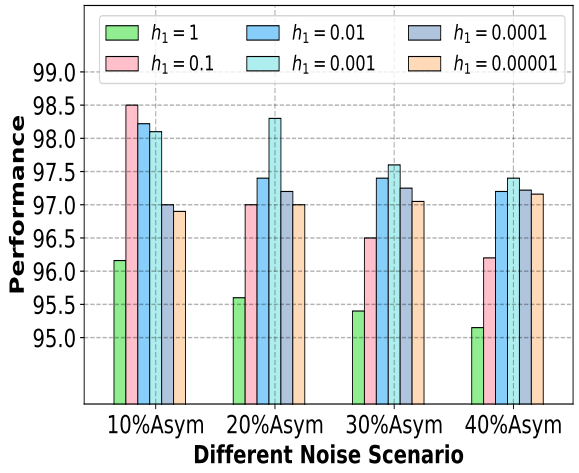
Hyper-parameter h_1. In Figure 6 (a), we compare the performance of different h_1 values under various Asym noise conditions. For each noise setting, we identify the optimal h_1 value and visualize the results in Figure 6 (c). Similarly, in Figure 6 (b), we compare the performance of different h_1 values under various Sym noise conditions. For each noise setting, we identify the optimal h_1 value and visualize the results in Figure 6 (d).

From the results in Figure 6 (c-d), we observe a negative correlation between the optimal h_1 value and the noise ratio. That is to say, as the noise ratio increases, a smaller h_1 accelerates the updates of Δw_c, allowing the clean LoRA Δw_c to better prevent the impact of mislabeled data.

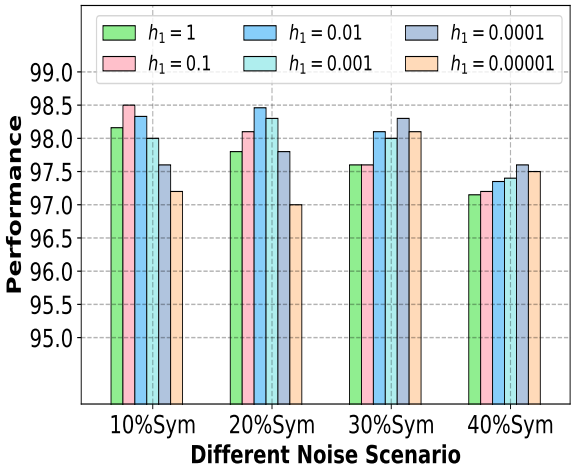
Overall, adding the constraint on the parameter update $\Delta\sigma_c$ of the clean LoRA and on the parameter update $\Delta\sigma_n$ of the noisy LoRA can make different LoRAs focus on learning different information from clean and noisy samples, thereby increasing the robustness of the noisy label detector.

I.4 Effect of Robust Loss Function

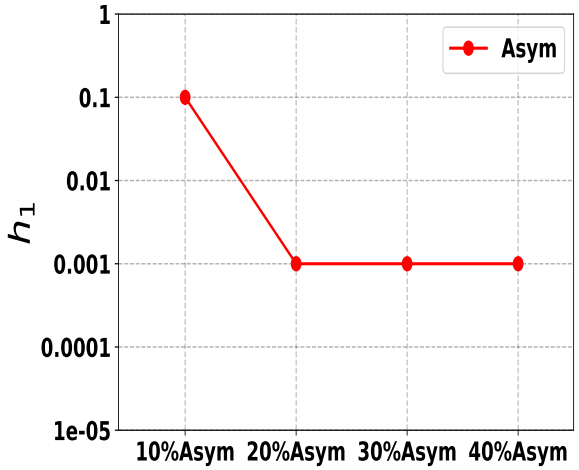
As discussed in Section B, due to GPT-4o being unable to generate correct labels for each noisy sample selected by the noisy label detector, we utilize the reversed cross-entropy loss functions to better learn from $\mathcal{D}_o$ with a certain noise ratio. We conduct an ablation experiment (see Table 15) to verify the effectiveness of this robust loss function by replacing it with cross-entropy loss functions.



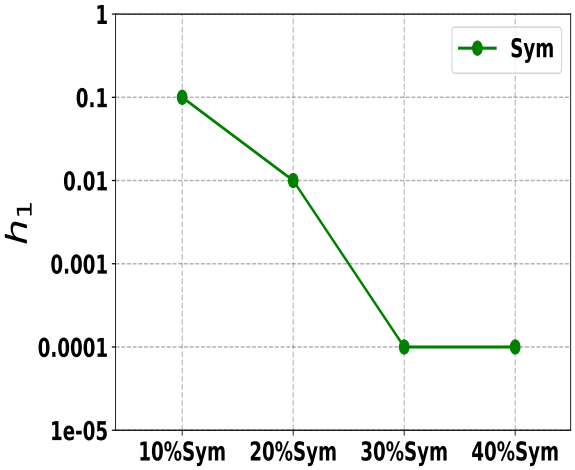
(a) The performance of our methods with different h_1 under different noise scenarios



(b) The performance of our methods with different h_1 under different noise scenarios



(c) Relationship between the optimal h_1 and noise ratio



(d) Relationship between the optimal h_1 and noise ratio

Figure 6: Analysis for the choice of hyper-parameter h_1 under different noise ratios on the Trec datasets.

Dataset	**Trec**				**SST-5**						**SST-2**					
Loss function(↓) / Noise(→)	20%**S**	40%**S**	20%**A**	40%**A**	20%**S**	40%**S**	20%**A**	40%**A**	20%**I**	40%**I**	20%**S**	40%**S**	20%**A**	40%**A**	20%**I**	40%**I**
cross-entropy loss	97.21	96.1	96.98	95.98	56.34	54.04	56.32	53.41	55.39	53.59	94.6	94.07	94.51	93.33	94.73	93.74
reversed cross-entropy loss	**98.46**	**97.60**	**98.30**	**97.40**	**57.39**	**55.62**	**57.57**	**55.39**	**57.02**	**55.02**	**96.50**	**95.75**	**96.27**	**95.18**	**96.08**	**95.00**

Table 15: Ablation study for loss functions on $\mathcal{D}_o$. **Bold** means the best score.

Algorithm 2 The proposed framework Delora

Input: A training dataset $\mathcal{D}$=$\{(x_i, y_i)\}_{i=1}^{N}$, $y \in \{1, \ldots, K\}$, warmup epochs T_w, epochs T_d to train the noisy label detector, epochs T_c to train the classifier model, clean LoRA parameters Δw_c, noisy LoRA parameters Δw_n.

// Stage 1: Training the Noisy Label Detector
while epoch $\leq T_w$ **do**
    Warm-up the LLM containing dual LoRAs on the $\mathcal{D}$ by $\mathcal{L}_{warm} = \mathcal{L}_{ce} + \mathcal{L}_{LoRA}$
    epoch = epoch + 1
end while
while epoch $\leq T_d$ **do**
    Construct the clean subset D_c by Eq. (2) and Eq. (3)
    Construct the positive and negative samples
    Update parameters of Δw_c and Δw_n by $\mathcal{L} = \mathcal{L}_{ce} + \mathcal{L}_{LoRA} + \mathcal{L}_{Detector}$
    epoch = epoch + 1;
end while
// Step 2: Training the Classifier Model
while epoch $\leq T_c$ **do**
    Compute the cross-entropy loss $\mathcal{L}_{\mathcal{D}_c}$ for selected clean samples by Eq.(7)
    Compute the reversed cross-entropy loss $\mathcal{L}_{\mathcal{D}_o}$ for relabeled noisy samples by Eq.(8)
    Update the parameter of the classifier model by $\mathcal{L} = \mathcal{L}_{\mathcal{D}_c} + \mathcal{L}_{\mathcal{D}_o}$
    epoch = epoch + 1
end while