Towards Robust and Generalized Parameter-Efficient Fine-Tuning for Noisy Label Learning

Yeachan Kim[1*], **Junho Kim**[1*], **SangKeun Lee**[1,2]
[1]Department of Artificial Intelligence, Korea University, Seoul, South Korea
[2]Department of Computer Science and Engineering, Korea University, Seoul, South Korea
{yeachan,monocrat,yalphy}@korea.ac.kr

Abstract

Parameter-efficient fine-tuning (PEFT) has enabled the efficient optimization of cumbersome language models in real-world settings. However, as datasets in such environments often contain noisy labels that adversely affect performance, PEFT methods are inevitably exposed to noisy labels. Despite this challenge, the adaptability of PEFT to noisy environments remains underexplored. To bridge this gap, we investigate various PEFT methods under noisy labels. Interestingly, our findings reveal that PEFT has difficulty in memorizing noisy labels due to its inherently limited capacity, resulting in robustness. However, we also find that such limited capacity simultaneously makes PEFT more vulnerable to interference of noisy labels, impeding the learning of clean samples. To address this issue, we propose **Clea**n **R**outing (CleaR), a novel routing-based PEFT approach that adaptively activates PEFT modules. In CleaR, PEFT modules are preferentially exposed to clean data while bypassing the noisy ones, thereby minimizing the noisy influence. To verify the efficacy of CleaR, we perform extensive experiments on diverse configurations of noisy labels. The results convincingly demonstrate that CleaR leads to substantially improved performance in noisy environments[1].

1 Introduction

The ever-growing size of pre-trained language models (PLMs) has presented significant challenges in adapting these models to desired tasks. In response to this practical limitation, parameter-efficient fine-tuning (PEFT) has emerged as a promising strategy for real-world environments. Instead of fine-tuning all weights, PEFT optimizes only a minimal set of parameters (e.g., biases (Zaken et al., 2022), adapters (Houlsby et al., 2019), prompts (Liu et al., 2022b), or low-rank matrices (Hu et al., 2022)), thereby drastically cutting down the computation and storage costs. Such efficiency has led PEFT methods to become the preferred standard approaches for applying PLMs in real-world contexts, such as federated learning (Kim et al., 2023; Liao et al., 2023) and continual learning (Ermis et al., 2022; Razdaibiedina et al., 2022).

While PEFT enables the efficient optimization of PLMs in real-world settings, datasets in such environments often contain noisy labels (i.e., incorrectly-labeled samples) (Jia et al., 2019; Alt et al., 2020), which adversely affects the generalization capabilities of PLMs (Wu et al., 2022). Given such distinct characteristics of the practical environments, PEFT methods are inevitably exposed to noisy labels during the optimization phase. Despite this significant challenge, there is a lack of prior research on the general adaptability of PEFT methods to noisy label learning (NLL) scenarios.

In this work, we bridge this research gap by exploring PEFT under noisy environments. Our results reveal that PEFT struggles in memorizing noisy labels due to its inherently limited capacity, which interestingly provides robustness[2] to noisy labels. However, we also find that such limited capacity simultaneously makes PEFT more susceptible to interference of noisy labels, which impedes learning ability for clean samples, potentially leading to sub-optimal performance. This characteristic markedly contrasts with the behaviors in full fine-tuning, presenting the necessity of PEFT that steers its limited learning capacity towards clean samples.

In response, we propose **Clea**n **R**outing (CleaR), a novel routing-based PEFT approach that adaptively activates PEFT modules. Our main strategy is to preferentially expose PEFT modules to correctly-labeled samples, while bypassing PEFT

[*] These authors contributed equally to this work.

[1] Our code is available at https://github.com/yeachan-kr/clear

[2] Following (Wang et al., 2021), we define robustness as the preservation of the performance under noisy labels.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5922–5936
August 11-16, 2024 ©2024 Association for Computational Linguistics

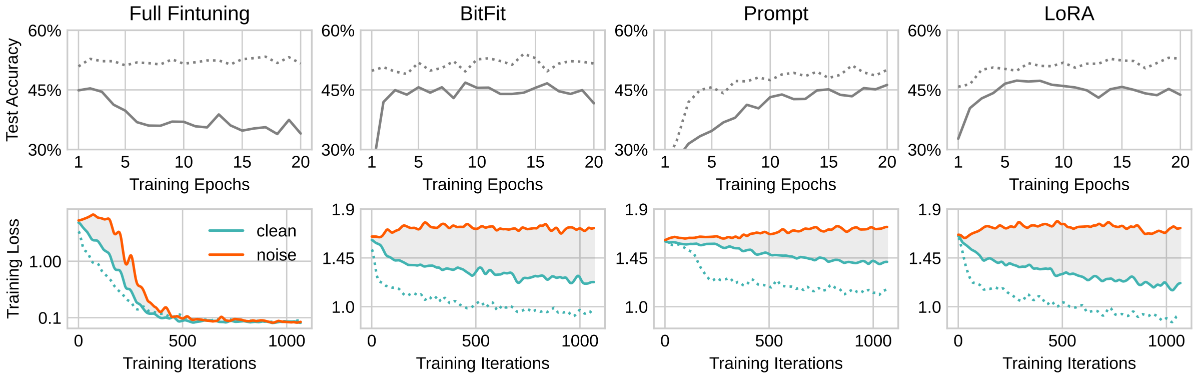


Figure 1: Comparison between PEFT methods and full fine-tuning on SST-5 with symmetric noise (60%). Dashed lines represent the training accuracy and loss of clean samples on uncorrupted datasets (i.e. only clean samples).

weights for noisy samples, thereby minimizing the detrimental impact of noisy ones. To this end, CleaR estimates the probability that a given sample is correctly labeled through the lens of training dynamics. These probabilities are then conditioned to draw the routing decision such that potentially clean samples are more encouraged to route through the PEFT modules. The independent sampling across layers enables the fine-grained optimization for the PEFT modules. Consequently, engaging PEFT primarily with clean samples minimizes the influence of noisy samples on PEFT.

The CleaR approach is designed to be model-agnostic, allowing us to integrate the concept of CleaR with various types of PEFT methods. To evaluate the effectiveness of CleaR-based PEFT methods, we conduct comprehensive experiments across diverse configurations, such as noise rate and noise type. Furthermore, we explore whether existing robust methods can be enhanced by seamlessly incorporating the CleaR approach. In summary, the contributions of this paper include the following:

- We first explore the effectiveness of PEFT methods in the context of noisy environments, providing a comprehensive analysis of their robustness and limitations.
- We propose CleaR, a novel PEFT approach that adaptively activates the PEFT modules to improve generalization capability while minimizing the influence of noisy samples.
- We demonstrate that CleaR-based PEFT methods achieve superior performance across various NLP tasks even under heavy noise conditions, thereby pushing the boundaries of robustness and generalization.

2 Investigation of PEFT on Noisy Labels

In this section, we systematically investigate PEFT methods in the presence of noisy labels.

Noisy environment Following the previous work (Wu et al., 2022), we simulate the noisy environment by randomly flipping the given labels. Specifically, we employ the SST-5 dataset with a symmetric noise rate of 60% (i.e., 60% of the training set contains incorrect labels). Note that the test set is not corrupted to confirm the generalization ability of the trained model. The detailed process is described in Appendix E.

PEFT methods We analyze the three representative types of PEFT methods (Chen et al., 2022) with the full fine-tuning: **LoRA** (Hu et al., 2022) that adds trainable decomposition matrices; **BitFit** (Zaken et al., 2022) that trains only biases; **Prompt Tuning** (Liu et al., 2022b) that appends learnable embeddings to the input of each layer[3].

Observations Figure 1 presents the evaluation results of the PEFT methods alongside the full fine-tuning. The accuracy results (first row in the figure) show that PEFT methods reveal superior robustness to the full fine-tuning, even though all methods suffer from performance degradation. To gain further insights into the behavior of each method, we include the training loss for both clean[4] and noisy samples (second low in the figure), which enables to analyze the learning capacity on clean and noisy samples (Arazo et al., 2019). These results show that PEFT methods have difficulty in memorizing the noisy samples, which interestingly contributes

[3] The setups for each PEFT methods are described in §D

[4] In this paper, we use the terms *clean* and *correctly-labeled* interchangeably to refer samples with ground truth annotation.

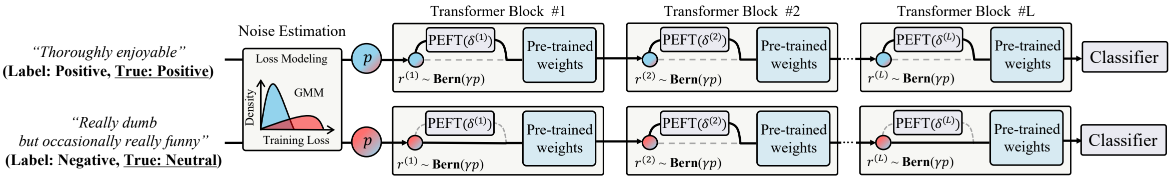


Figure 2: Overview of the **Clea**n **R**outing. CleaR first estimate the probability of each sample being clean based on the training losses. Based on the estimated probability, CleaR adaptively activates PEFT modules by favoring the potentially clean samples.

to the robustness. However, it is evident that PEFT methods also face challenges in learning from clean samples (i.e., a large gap in losses of clean samples when exposed to the noisy dataset). The result implies that PEFT, which inherently has limited capacity (Ding et al., 2023), is more vulnerable to the interference of noisy labels, impairing its learning ability on clean data. This limitation can lead to sub-optimal performance, underscoring the need for PEFT methods that can steer its limited capacity toward clean samples.

3 CleaR: PEFT with Clean Routing

In this section, we elaborate **Clea**n **R**outing (CleaR) in detail. The core idea is to adaptively activate the PEFT modules to circumvent the detrimental effect from noisy labels. To achieve this, CleaR estimates the probability whether a given sample is correctly labeled, leveraging the distinct training dynamics between clean and noisy samples. With these probabilities, CleaR steers the potentially clean samples to route through PEFT modules, whereas the noisy ones are directed to bypass PEFT modules, thereby minimizing their influence on the PEFT. To improve the routing stability of CleaR, we also introduce consistency regularization for PEFT modules. Figure 2 illustrates the overall procedures of CleaR.

3.1 Parameter-Efficient Fine-Tuning Modules

We start by defining PEFT modules in CleaR. Since CleaR is designed to be module-agnostic, diverse PEFT methods can be seamlessly integrated with CleaR method. To showcase such applicability, we consider four representative PEFT modules (i.e., Adapter, BitFit, Prompt-tuning, and LoRA), which are commonly employed within NLP community[5]. While these PEFT modules have distinct characteristics, they can be succinctly represented as additional parameters $\boldsymbol{\delta}$ added to the pre-trained weights $\boldsymbol{\theta}$ of PLMs (Ding et al., 2023). More specifically, since PEFT modules are uniformly distributed across all layers, we represent these modules as a set of additional parameters $\boldsymbol{\delta} = \{\delta^{(1)}, \delta^{(2)}, \dots, \delta^{(L)}\}$, where L is the number of layers. Let the prediction involving these PEFT modules be denoted as $f(x, \boldsymbol{\delta} + \boldsymbol{\theta})$, the objective with an arbitrary loss function $\mathcal{L}$ can be formulated as follows:

$$\min_{\delta} \mathcal{L}(x) = \mathcal{L}(f(x, \boldsymbol{\delta} + \boldsymbol{\theta}), y), \tag{1}$$

where x and y denote the training sample and the given label, respectively. Note that the PEFT modules and the task-specific classifier are only updated during training.

3.2 Routing PEFT Modules

Building upon these PEFT modules, we introduce a clean routing scheme that adaptively activates modules according to the estimated probability that a given sample is correctly labeled.

Estimating clean probability for routing To derive the clean probability for each sample, we leverage the distinct learning patterns when learning with clean and noisy samples: *deep networks prefer to learn clean samples first before fitting noisy ones* (Arazo et al., 2019). Namely, noisy samples tend to have a higher loss than clean samples in the early training stage. This enables to distinguish potentially clean samples from the datasets based on loss deviation (Jiang et al., 2018; Han et al., 2018). Taking advantage of such phenomena, we adopt the widely-used Gaussian Mixture Model (GMM) in noise label learning (Li et al., 2020; Qiao et al., 2022), in which the probability of samples being clean is estimated by the per-sample loss.

Based on the estimated mixture models, we compute the clean probability p using the posterior probability, i.e., $p(g|\ell)$ where ℓ is the loss for the training sample, and g is the Gaussian component

[5] The detailed illustrations of these modules on CleaR are shown in the Appendix §D (Figure 6)

with a smaller mean (i.e., smaller loss). Specifically, we first train our model for the k epochs warm-up to measure the loss of samples, and then we estimate the clean probability for each training sample on every subsequent epoch. It is noteworthy that we leverage training losses obtained from the previous epoch to estimate clean probabilities, thereby reducing the additional computational cost from redundant forward passes.

Sampling routing decision Once the clean probability is estimated, the PEFT modules are stochastically routed across the transformer layers. To derive the routing decision (i.e., routing through PEFT or bypassing PEFT), we sample the decision from a Bernoulli distribution with the estimated clean probability p:

$$r \sim \textsc{Bernoulli}(\gamma p), \tag{2}$$

where r is an independent Bernoulli random variable with a probability γp of being 1 and a probability $1 - \gamma p$ of being 0. The coefficient $\gamma \in [0, 1]$ limits the range of clean probability, setting its upper bound at γ. This coefficient plays a role in preventing over-reliance on the estimated probability, considering that small-loss samples might still contain noisy samples (i.e., high clean probability despite being noisy samples) (Li et al., 2020). Crucially, this routing decision is independently made at each layer, allowing for the fine-grained differentiation of each sample's influence based on its probability of being clean. For example, if the clean probability is 70% and the number of layers is 10, seven PEFT modules are activated in average[6].

Activating PEFT based on the decision The routing decisions across different layers are then applied to all PEFT modules. Formally, let the hidden states in the l-th layer be denoted as $h^{(l)}$, and the hidden state in the next layer is derived as follows:

$$h^{(l+1)} = \begin{cases} \text{Trans}^{(l)}(h^{(l)}, \delta^{(l)} + \theta^{(l)}), & \text{if } r^{(l)} = 1 \\ \text{Trans}^{(l)}(h^{(l)}, \theta^{(l)}), & \text{if } r^{(l)} = 0 \end{cases} \tag{3}$$

where $\delta^{(l)}$ and $\theta^{(l)}$ represent PEFT module and pre-trained parameters in the l-th layer, respectively, and $r^{(l)}$ indicates the routing decision on the layer. $\text{Trans}^{(l)}(\cdot)$ denotes the function of the transformer block. Through the above routing decision, CleaR activates a subset of PEFT modules, i.e., $\boldsymbol{\delta_r} = \{\delta^{(l)} | \delta^{(l)} \in \boldsymbol{\delta}, r^{(l)} = 1\}$, on each forward pass. This routing scheme ensures that PEFT modules are favorably activated for potentially clean samples and deactivated for noisy ones, thereby reducing the influence of noisy samples.

CleaR in inference phase As clean routing only performs in training, we need to decide the routing strategy during inference. To make the most of well-trained PEFT modules and ensure consistency with training, we empirically set the routing probability to the upper bound, i.e., $p = 1.0$ in Eq. (2), and observe that it works well in practice.

3.3 Consistency Regularization for CleaR

While the routing scheme effectively mitigates the influence of noisy labels, model predictions may end up being overly diverse due to varying activations with each forward pass, potentially resulting in training instability. To address this issue, we introduce a consistency regularization to minimize the model variability. Considering that guiding the model to adhere to past predictions can enhance the stability and consistency of training (Shen et al., 2022; Xu et al., 2023), we regulate the model by minimizing the distance between its current and previous predictions. Specifically, we make ensemble predictions from multiple forwards to reduce predictive variance and increase stability:

$$f_{\text{ens}}(x, \bar{\boldsymbol{\delta}}_r + \theta) = \frac{1}{N} \sum_{k=1}^{N} f(x, \bar{\boldsymbol{\delta}}_{r,k} + \boldsymbol{\theta}), \tag{4}$$

where N is the number of forwards, and $\bar{\boldsymbol{\delta}}_{r,k}$ represents activated PEFT modules in the k-th forward of the previously trained model. It is noteworthy that, for computational efficiency, we reuse the predictions, which were previously used for fitting GMM. With the derived predictions, the model with CleaR is optimized with the following loss:

$$\begin{aligned} \min_{\delta_r} \mathcal{L}(x) &= \mathcal{L}_{\text{CE}}(f(x, \boldsymbol{\delta}_r + \boldsymbol{\theta}), y) \\ &+ \lambda \mathcal{L}_{\text{CE}}(f(x, \boldsymbol{\delta}_r + \boldsymbol{\theta}), f_{\text{ens}}(x, \bar{\boldsymbol{\delta}}_r + \boldsymbol{\theta})). \end{aligned} \tag{5}$$

where $\mathcal{L}_{\text{CE}}(\cdot)$ indicates the cross-entropy loss, and λ is a coefficient to control the strength of the regularization.

[6] While different positions of the PEFT could have varying effects on the prediction (Chen et al., 2023), in this work, we focus solely on the clean probability as a trigger to activate the PEFT. We reserve further exploration of this for future work.

Table 1: Evaluation results of Peak accuracy and Average accuracy on SST-5 test set under different levels of label noise. The best and second best results are highlighted in **boldface** and underlined, respectively.

Methods	Clean	Symmetric						Asymmetric					
		20%		40%		60%		10%		20%		40%	
		Peak.	Avg.	Peak.	Avg.	Peak.	Avg.	Peak.	Avg.	Peak.	Avg.	Peak.	Avg.
Full Fine-tuning	53.4	51.3	47.0	50.6	42.9	47.9	35.5	**52.5**	49.1	50.8	46.5	46.1	37.4
PEFT methods													
Adapter (2019)	53.3	51.9	48.1	50.5	45.8	47.2	38.1	52.2	51.0	50.9	47.0	48.1	38.0
BitFit (2022)	53.0	51.7	51.0	50.8	48.1	48.1	43.5	52.1	50.5	52.1	49.2	48.9	42.1
Prompt (2022b)	52.7	51.1	48.6	50.7	49.1	47.7	45.7	51.7	50.8	49.4	48.2	46.1	41.7
LoRA (2022)	**53.6**	52.0	49.5	50.2	47.5	48.2	46.1	51.9	51.1	50.5	47.4	47.2	41.8
PEFT methods with CleaR (ours)													
CleaR$_{\text{Adapter}}$	53.4	**52.4**	**51.8**	51.5	50.4	50.4	49.7	**52.5**	50.8	51.4	47.4	48.1	44.6
CleaR$_{\text{BitFit}}$	53.1	51.9	51.1	**51.6**	**51.2**	**51.4**	**51.1**	52.0	**51.4**	**52.3**	50.4	**49.2**	**48.3**
CleaR$_{\text{Prompt}}$	52.6	51.0	50.5	51.4	49.5	49.4	47.2	52.1	51.2	52.0	**51.4**	47.8	46.5
CleaR$_{\text{LoRA}}$	53.3	51.4	50.1	51.2	49.0	50.0	48.9	52.0	51.1	51.0	50.4	47.6	43.2

4 Experiments

We demonstrate the efficacy of CleaR across diverse configurations of noisy environments.

4.1 Configurations of Noisy Labels

To comprehensively assess our method diverse scenarios characterized by different noisy labels, we evaluate each baseline against three distinct types of noisy labels: symmetric, asymmetric, and instance-dependent. We provide detailed descriptions of each noisy label type and the methodology for constructing noisy label datasets in §E.

4.2 Baselines and Implementations

Following the previous works, we use the BERT-base and BERT-large model (Devlin et al., 2019). Building on this PLM, we mainly compare CleaR with the full fine-tuning and widely-used PEFT methods including Adapter (Houlsby et al., 2019), LoRA (Hu et al., 2022), Prompt tuning (Liu et al., 2022b), and BitFit (Zaken et al., 2022). For a fair comparison, we utilize the same settings for PEFT modules (e.g., bottleneck dimension, prompt length) for baselines and our CleaR. The detailed implementations are represented in §F. Additionally, we compare CleaR with the existing NLL methods to confirm the competitiveness of the proposed method in §5.3.

4.3 Evaluation Metric

Building upon previous work (Wu et al., 2022), we assess each baseline model using two metrics: the instantaneous peak accuracy and the average accuracy across the last few epochs. The former metric evaluates the model's generalization performance, while the latter reflects its stability. Consequently, a smaller gap between these two metrics indicates a more effective model to noisy labels. For all CleaR models, we report the average performance on 10 different seeds considering their stochasticity.

4.4 Sentiment Analysis

We first evaluate baselines in sentiment analysis due to the inherent subjectivity of this task, which often results in noisy labels. Following the previous work, we use the Standard Sentiment Treebank (SST-5) dataset (Socher et al., 2013). For the levels of noisy labels, we scale the symmetric noise from 20% to 60% and asymmetric noise from 10% to 40%, respectively.

Table 1 presents the evaluation results for the task. As observed in the previous analysis, PEFT exhibits better robustness than full fine-tuning across different types and levels of noisy labels. Notably, CleaR-based methods show substantial improvement compared to PEFT methods on both metrics. This demonstrates that CleaR enhances the generalization capability of PEFT (i.e., improved peak accuracy) while maintaining or even strengthening its robustness (i.e., reduced gaps between peak and average accuracy).

4.5 Intent Detection

We further evaluate CleaR on intent detection. Given that the task is typically employed in conversational systems, the query usually consists of only a few words (e.g., 6 to 12 words) (Casanueva

Table 2: Evaluation results of Peak accuracy and Average accuracy on BANKING77 test set under different levels of label noise. The best and second best results are highlighted in **boldface** and underlined, respectively.

Methods	Clean	Symmetric						Asymmetric					
		20%		40%		60%		10%		20%		40%	
		Peak.	Avg.	Peak.	Avg.	Peak.	Avg.	Peak.	Avg.	Peak.	Avg.	Peak.	Avg.
Full Fine-tuning	92.9	88.8	83.4	84.3	72.6	78.2	58.5	90.8	87.6	87.3	79.4	66.9	54.6
PEFT methods													
Adapter (2019)	92.7	88.5	85.4	86.6	78.4	80.9	67.1	90.3	88.6	86.7	78.5	65.3	56.2
BitFit (2022)	92.5	88.9	88.7	86.7	85.9	80.1	76.5	90.2	89.8	86.3	83.1	66.7	62.4
Prompt (2022b)	91.9	87.8	87.4	85.6	84.5	83.2	77.2	89.7	88.4	85.4	84.9	61.6	58.9
LoRA (2022)	93.0	89.2	88.3	86.8	85.8	81.9	77.5	90.1	88.6	86.9	83.1	64.5	61.8
PEFT methods with CleaR (ours)													
CleaR$_{\text{Adapter}}$	**93.1**	**90.1**	89.7	**88.2**	**87.3**	82.3	80.2	**91.4**	90.3	**87.6**	**86.1**	67.3	66.1
CleaR$_{\text{BitFit}}$	92.4	89.8	89.2	87.3	86.9	82.9	82.2	90.7	**90.4**	87.5	86.1	67.1	63.4
CleaR$_{\text{Prompt}}$	92.1	88.1	87.6	85.8	84.9	83.7	81.0	89.9	89.2	85.7	84.8	64.5	62.3
CleaR$_{\text{LoRA}}$	92.8	90.0	**89.8**	**87.4**	86.9	**84.2**	**83.5**	91.3	90.3	87.2	85.9	**68.9**	**68.1**

Table 3: Ablation study of CleaR on SST-5 (60% of symmetric noise). For the ablation of routing strategies, we remove the consistency regularization to solely evaluate each routing strategy.

Methods	Peak.	Avg.
CleaR$_{\text{Adapter}}$(ours)	50.4	49.7
Components in CleaR		
CleaR w/o Clean Routing	48.4	41.1
CleaR w/o Regularization	49.9	48.6
CleaR w/o Clean Routing & Regularization	47.2	40.0
Routing Strategy in CleaR		
CleaR w/ Clean Routing	49.9	48.6
CleaR w/ Deterministic Routing	48.1	44.2
CleaR w/ Random Routing	47.5	40.5
CleaR w/ Noisy Routing	46.9	34.3

et al., 2020). Such brevity can amplify ambiguity, potentially leading to noisy annotations. We utilize the BANKING77 (Casanueva et al., 2020) that encompasses 77 fine-grained intent categories. For the levels of noisy labels, we scale the symmetric noise from 20% to 60% and asymmetric noise from 10% to 40%, respectively.

Table 2 presents the evaluation results for intent detection. Similar to the previous task, PEFT achieves superior robustness to noisy labels compared to full fine-tuning by showing higher average accuracy even on the highest noisy levels. Meanwhile, CleaR consistently outperforms both the PEFT and full fine-tuning methods, demonstrating its efficacy to improve the robustness. It is noteworthy that in certain setups, while PEFT shows better average accuracy, it yields slightly lower peak accuracy due to its limited capacity to memorize clean samples (e.g., asymmetric noise levels between 20% and 40%). Interestingly, CleaR significantly boosts the peak accuracy of each PEFT variant, achieving similar or even better peak accuracy than full fine-tuning. These results again verify that CleaR successfully mitigates the limited memorization of PEFT methods, thereby leading to better generalization and robustness.

5 Analysis

To make a more comprehensive analysis of our CleaR, we designed a series of fine-grained experiments aimed at addressing the following research questions (RQs):

- **RQ1.** How does each component within CleaR contribute to its overall performance? (§5.1)
- **RQ2.** Can CleaR be integrated with other methods for learning with noisy labels? (§5.2)
- **RQ3.** Can CleaR be combined with other noisy label learning methods? (§5.3)
- **RQ4.** Does CleaR offer improvements under more realistic noisy label scenarios? (§5.4)
- **RQ5.** Can CleaR be generalized to the large-sized model? (§C)

5.1 Ablation Studies on CleaR (RQ1)

We perform ablation studies to investigate the contributions of each component and routing strategy in CleaR. Table 3 presents the ablation results.

Routing and Regularization As shown in the upper part of Table 3, omitting routing and consistency regularization largely affects both peak

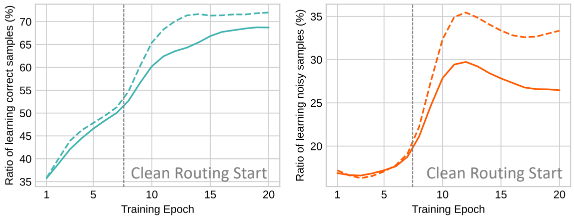
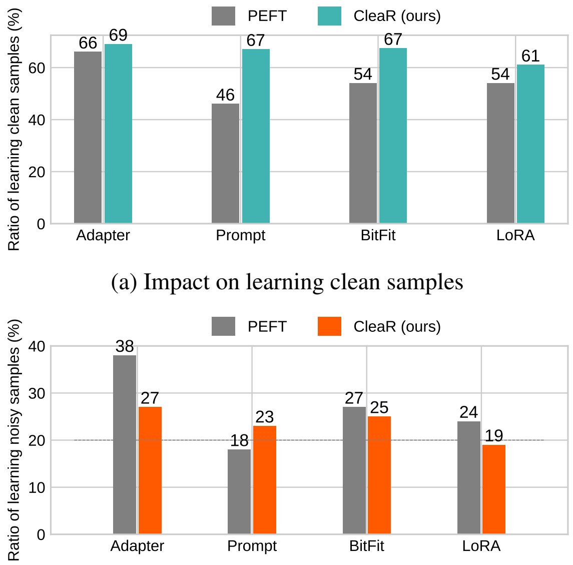


Figure 3: Ratios of memorizing clean (**Left**, larger is better) and noisy samples (**Right**, smaller is better) on different routing methods. Dashed lines and solid lines indicate Deterministic Routing and Clean Routing (ours), respectively.



(a) Impact on learning clean samples

(b) Impact on learning noisy samples

Figure 4: Impact on two memorizations when applying CleaR to PEFT methods. Best viewed in color..

and average accuracy. The results demonstrate that each component is essential for improving generalization and robustness. In particular, we observe that the routing mechanism plays a significant role.

Routing strategy To further investigate the proposed routing scheme, we compared it with three different schemes: (i) Random Routing, where PEFT modules are activated randomly; (ii) Noisy Routing, where PEFT modules are activated in proportion to the noisy probability, representing the opposite approach of CleaR; and (iii) Deterministic Routing, where the noisy samples are filtered out by leveraging the estimated probability based on GMM, and PEFT modules are only deterministically activated on remaining samples.

As shown in the lower part of Table 3, adopting Random and Noisy Routing substantially decreases both metrics, indicating that favoring clean samples in the routing is indeed beneficial to achieve both generalization and robustness ability. We then compare our Clean Routing with the Deterministic Routing. We observe that Deterministic Routing exhibits inferior performance than our Clean Routing. Moreover, Figure 3 illustrates the differences in accuracy on clean and noisy samples between Clean Routing and Deterministic Routing. It reveals that Deterministic Routing leads to increased memorization of noisy samples compared to our stochastic Clean Routing, which is attributed to lower performance. These results emphasize the significance of stochastic routing in Clean Routing, which can differentiate samples in a more fine-grained manner.

Table 4: Complementary effect of incorporating CleaR to NLL methods on SST-5 (60% symmetric noise).

Methods	NLL Methods	Peak.	Avg.
Full Fine-tuning	None	47.9	35.5
	Co-teaching (2018)	50.1	46.1
	SELC (2022)	48.5	39.7
	STGN (2022)	47.7	38.6
Adapter	None	47.2	38.1
	Co-teaching (2018)	50.3	45.9
	SELC (2022)	47.5	39.7
	STGN (2022)	48.7	39.8
CleaR$_{\text{Adapter}}$ (ours)	None	50.4	49.7
	Co-teaching (2018)	50.6	50.1
	SELC (2022)	50.5	**50.2**
	STGN (2022)	**50.8**	49.4

5.2 Memorization Effects (RQ2)

To confirm whether CleaR mitigates the underfitting problems of PEFT methods on clean samples, we compare the ratio of memorizing clean and noisy samples after fine-tuning. Figure 4 shows the comparison. Notably, we observe that adopting CleaR into PEFT methods largely increases the memorization of clean samples, while preserving or even reducing the memorization of noisy samples. These results verify that CleaR successfully mitigates the underfitting of PEFT methods on the clean dataset by favorably activating the PEFT modules on potentially clean samples.

5.3 CleaR on Different NLL Methods (RQ3)

We compare CleaR with existing NLL methods to demonstrate its applicability. We consider three approaches: Co-teaching (Han et al., 2018), SELC

Table 5: Peak and Average accuracy (%) on SST-5 under different levels of instance-dependent noise.

Method	Clean	40%		60%	
		Peak.	Avg.	Peak.	Avg.
Full Fine-tuning	53.4	49.0	43.9	44.8	38.9
PEFT methods					
Adapter	53.3	48.8	44.1	44.2	39.7
BitFit	53.0	50.0	45.4	44.8	41.6
Prompt	52.7	49.5	46.2	42.8	38.8
LoRA	**53.6**	49.1	46.9	44.2	39.8
PEFT methods with CleaR (ours)					
CleaR$_{\text{Adapter}}$	53.4	50.5	46.4	45.7	43.4
CleaR$_{\text{BitFit}}$	53.1	**51.0**	46.5	45.2	42.6
CleaR$_{\text{Prompt}}$	52.6	50.2	44.1	44.8	43.2
CleaR$_{\text{LoRA}}$	53.3	49.7	**47.1**	**46.5**	**44.8**

(Lu and He, 2022), and STGN (Wu et al., 2022)[7]. Table 4 showcases the comparison and integration results with CleaR and Adapter. We observe that NLL methods can be enhanced with CleaR, as it can be seamlessly integrated by adding PEFT modules. The results show that while combining NLL methods with Adapter brings marginal improvement, adopting CleaR leads to a marked enhancement across both metrics. This suggests that CleaR can place the current state-of-the-art NLL methods on a more solid footing by enjoying the robustness and improved generalization ability of PEFT.

5.4 Instance-dependent Label Noise (RQ4)

To investigate CleaR in more realistic settings, we evaluate CleaR in scenarios where noisy labels arise from input features. Table 5 presents the evaluation results on SST-5 with the instance-dependent noise. Similar to the other noise settings, CleaR consistently yields the best performance on both peak and average accuracy across different noise ratios, highlighting the validity of CleaR in addressing feature-dependent noise.

6 Related Work

6.1 Robustness of PEFT Methods

The well-known beneficial properties of PEFT are its generalization capabilities in low-data environments (Liu et al., 2022a; Zaken et al., 2022) and stability (Houlsby et al., 2019; Sung et al., 2021). These studies have demonstrated that full fine-tuning suffers from overfitting on small datasets, whereas PEFT methods achieve superior performance in few-shot settings due to the regularization effect on the pre-trained models (Fu et al., 2023). Regarding model calibration, recent work has shown that preserving the pre-trained features through the PEFT methods improves the model calibration by preventing overfitting (He et al., 2023; Ding et al., 2023). Following the robustness on the overfitting, several studies have demonstrated that PEFT methods also being less suffered from catastrophic forgetting (He et al., 2021), and their robustness against forgetting is further evidenced in continual learning scenarios (Ermis et al., 2022).

Previous studies have broadened the understanding of PEFT in diverse aspects. Building upon this research direction, we explore its robustness to noisy labels, which remains a challenging problem within the deep learning community. Therefore, exploring PEFT methods in this context could provide valuable insights into their practical applicability.

6.2 Noisy Label Learning in NLP

Noisy labels are inevitably introduced on large-scale datasets (Jia et al., 2019; Wu et al., 2023). To mitigate the influence of noisy labels in text classification, Jindal et al. (2019) have proposed a noise transition matrix on top of the classifier, learning the transition distribution of noisy labels, and Wang et al. (2023) have tackled noisy labels in classification tasks with supplemental guidance from large language models (e.g., ChatGPT). In addition, Zhuang et al. (2023) have utilized the dynamics of features with generative models during training to learn the transition matrix for noisy labels. For named-entity recognition, Meng et al. (2021) have introduced a self-training method based on the contextualized augmentations, and Zhou and Chen (2021) have proposed a co-regularization in which multiple models teach each other to avoid negative memorization. In entity linking, Le and Titov (2019) have explored an interpretable method to identify clean samples on the premise that interpretable samples tend to be clean. Recently, Wu et al. (2022) have proposed a noise-robust optimization by introducing stochastic tailor-made gradient noise.

While these works have achieved robustness to noisy labels, these works typically consider the scenarios that the entire parameters in PLMs are optimized during fine-tuning, which is challenging due to the huge number of parameters. To the best of our knowledge, we are the first to explore

[7] The detailed settings are included in the Appendix §G

the PEFT methods for NLL. Given that CleaR is built upon PEFT methods, it can enjoy the benefits of training efficiency and robustness, which is favorable in the real-world environment.

7 Conclusion

In this work, we have explored whether the PEFT methods can be generalized to noisy environments. Interestingly, we have found that, while the limited capacity of PEFT allows the robustness to noisy labels, it can also act as a double-edged sword that hinders learning even with clean samples. In response, we have proposed CleaR, a novel routing-based PEFT approach that adaptively activates the PEFT modules by considering the probability of samples being clean. Extensive experiments have convincingly demonstrated the efficacy of CleaR across diverse configurations of noisy labels. Moreover, our in-depth analysis has demonstrated that CleaR effectively mitigates the underfitting on clean samples of PEFT methods.

8 Limitation

While we have shown that CleaR successfully improves the effectiveness of PEFT on various NLL scenarios, there exists a few limitations.

Exploration on Different Architectures Our efforts have been focused on improving the efficacy of PEFT for encoder models, aligning with previous studies (Wu et al., 2022). Therefore, the applicability of CleaR methods to different architectures (e.g., decoder, encoder-decoder models) remains under-explored in this work. Nevertheless, based on recent evidence suggesting that routing-based PEFT methods can be effectively generalized to various architectures beyond encoder models (Wang et al., 2022; Choi et al., 2023), we believe that CleaR is expected to work well within other architectures. We leave the exploration of this direction as promising future work.

Computational Overheads The adaptive routing mechanism in CleaR could potentially introduce computational overhead in two main areas: (i) determining the probability of each sample, and (ii) executing the PEFT routing. However, the overhead for (i) can be mitigated by caching the samples' losses during the training phase, eliminating the need for separate procedures to compute the training loss. As for (ii), in comparison to other routing-based methods cited as (Choi et al., 2023) that employ parameterized routers, the additional computational costs in CleaR are negligible, as the router in CleaR is non-parametric, and decisions are made through sampling, which streamlines the process.

Acknowledgment

This work was supported by the Basic Research Program through the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2021R1A2C3010430) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2019-II190079, Artificial Intelligence Graduate School Program (Korea University)).

References

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. 2020. TACRED revisited: A thorough evaluation of the TACRED relation extraction task. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1558–1569. Association for Computational Linguistics.

Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. 2019. Unsupervised label noise modeling and loss correction. In *Proceedings of the International Conference on Machine Learning*, volume 97, pages 312–321. PMLR.

Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulic. 2020. Efficient intent detection with dual sentence encoders. *CoRR*, abs/2003.04807.

Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. 2022. Revisiting parameter-efficient tuning: Are we really there yet? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2612–2626. Association for Computational Linguistics.

Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. 2023. Parameter-efficient fine-tuning design spaces. In *Proceedings of the Eleventh International Conference on Learning Representations*. OpenReview.net.

Joon-Young Choi, Junho Kim, Jun-Hyung Park, Wing-Lam Mok, and SangKeun Lee. 2023. Smop: Towards efficient and effective prompt tuning with sparse mixture-of-prompts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14306–14316. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186. Association for Computational Linguistics.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nat. Mac. Intell.*, 5(3):220–235.

Beyza Ermis, Giovanni Zappella, Martin Wistuba, Aditya Rawal, and Cedric Archambeau. 2022. Memory efficient continual learning with transformers. In *Advances in Neural Information Processing Systems*, volume 35, pages 10629–10642. Curran Associates, Inc.

Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2023. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12799–12807.

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*, volume 31, pages 8536–8546. Curran Associates, Inc.

Guande He, Jianfei Chen, and Jun Zhu. 2023. Preserving pre-trained features helps calibrate fine-tuned language models. In *Proceedings of the Eleventh International Conference on Learning Representations*. OpenReview.net.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 2208–2222. 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 International Conference on Machine Learning*, volume 97, 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 *Proceedings of the Tenth International Conference on Learning Representations*. OpenReview.net.

Wei Jia, Dai Dai, Xinyan Xiao, and Hua Wu. 2019. ARNOR: attention regularization based noise reduction for distant supervision relation classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1399–1408. Association for Computational Linguistics.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Proceedings of the International Conference on Machine Learning*, volume 80, pages 2309–2318. PMLR.

Ishan Jindal, Daniel Pressel, Brian Lester, and Matthew S. Nokleby. 2019. An effective label noise model for DNN text classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3246–3256. Association for Computational Linguistics.

Yeachan Kim, Junho Kim, Wing-Lam Mok, Jun-Hyung Park, and SangKeun Lee. 2023. Client-customized adaptation for parameter-efficient federated learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1159–1172. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the Third International Conference on Learning Representations*. OpenReview.net.

Phong Le and Ivan Titov. 2019. Distant learning for entity linking with automatic noise detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4081–4090. Association for Computational Linguistics.

Junnan Li, Richard Socher, and Steven C. H. Hoi. 2020. Dividemix: Learning with noisy labels as semi-supervised learning. In *Proceedings of the Eighth International Conference on Learning Representations*. OpenReview.net.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4582–4597. Association for Computational Linguistics.

Baohao Liao, Yan Meng, and Christof Monz. 2023. Parameter-efficient fine-tuning without introducing new latency. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 4242–4260. Association for Computational Linguistics.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 1950–1965. Curran Associates, Inc.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022b. 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*, pages 61–68. Association for Computational Linguistics.

Yangdi Lu and Wenbo He. 2022. SELC: self-ensemble label correction improves learning with noisy labels. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 3278–3284. ijcai.org.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Xuan Wang, Yu Zhang, Heng Ji, and Jiawei Han. 2021. Distantly-supervised named entity recognition with noise-robust learning and language model augmented self-training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10367–10378. Association for Computational Linguistics.

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 *Proceedings of the Sixth International Conference on Learning Representations*. 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*, pages 960–970. International Committee on Computational Linguistics.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2022. Progressive prompts: Continual learning for language models. In *Proceedings of the Eleventh International Conference on Learning Representations*. OpenReview.net.

Yiqing Shen, Liwu Xu, Yuzhe Yang, Yaqian Li, and Yandong Guo. 2022. Self-distillation from the last mini-batch for consistency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11933–11942. IEEE.

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*, pages 1631–1642. Association for Computational Linguistics.

Yi-Lin Sung, Varun Nair, and Colin Raffel. 2021. Training neural networks with fixed sparse masks. In *Advances in Neural Information Processing Systems*, volume 34, pages 24193–24205. Curran Associates, Inc.

Deng-Bao Wang, Yong Wen, Lujia Pan, and Min-Ling Zhang. 2021. Learning from noisy labels with complementary loss functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10111–10119. AAAI Press.

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*, pages 12528–12540. Association for Computational Linguistics.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5744–5760. Association for Computational Linguistics.

Tingting Wu, Xiao Ding, Minji Tang, Hao Zhang, Bing Qin, and Ting Liu. 2022. STGN: an implicit regularization method for learning with noisy labels in natural language processing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7587–7598. Association for Computational Linguistics.

Tingting Wu, Xiao Ding, Minji Tang, Hao Zhang, Bing Qin, and Ting Liu. 2023. Noisywikihow: A benchmark for learning with real-world noisy labels in natural language processing. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4856–4873. Association for Computational Linguistics.

Yige Xu, Xipeng Qiu, Ligao Zhou, and Xuan-Jing Huang. 2023. Improving BERT fine-tuning via self-ensemble and self-distillation. *J. Comput. Sci. Technol.*, 38(4):853–866.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 1–9. Association for Computational Linguistics.

Wenxuan Zhou and Muhao Chen. 2021. Learning from noisy labels for entity-centric information extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5381–5392. Association for Computational Linguistics.

Yuchen Zhuang, Yue Yu, Lingkai Kong, Xiang Chen, and Chao Zhang. 2023. Dygen: Learning from noisy labels via dynamics-enhanced generative modeling.

In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3674–3686. ACM.

Appendix

A PEFT Analysis on Different Dataset

We further conducted the same analysis on a different dataset to validate the generality of our observations. Specifically, we examined PEFT methods on the BANKING77 (Casanueva et al., 2020) dataset, which focuses on an intent detection task. The results are presented in Figure 5. Similar to our analysis of the SST-5 dataset, we observed a similar trend: (i) PEFT methods exhibit greater robustness than full fine-tuning, and (ii) its limited memorization on noisy label attributes to robustness, although they also inhibit memorization even on clean samples. These findings further support our observations and analysis regarding the robustness of PEFT methods to noisy labels.

B Experiments on Additional Datasets

To further demonstrate the broad applicability of our proposed method, we have evaluated the proposed methods on the TREC and 20Newsgroups datasets. Table 6 shows the evaluation results on the two datasets with 60% symmetric noises. We can observe the similar performance trend with the existing benchmarks, further underscoring the general applicability of the proposed method.

Table 6: Peak and Average accuracy (%) on TREC and 20NewsGroups with 60% symmetric noisy labels. Best results are highlighted in boldface.

Method	TREC		20NewsGroups	
	Peak.	Avg.	Peak.	Avg.
Full Fine-Tuning	90.2	55.9	62.3	37.6
PEFT methods				
Adapter	91.4	70.6	61.8	49.7
BitFit	92.1	91.9	61.8	61.2
LoRA	89.4	77.2	61.2	51.5
PEFT methods with CleaR (ours)				
CleaR$_{\text{Adapter}}$	**93.9**	**92.7**	62.9	**61.6**
CleaR$_{\text{BitFit}}$	93.3	92.6	63.0	58.2
CleaR$_{\text{LoRA}}$	92.0	91.3	**63.2**	57.4

C CleaR on Larger Model

To evaluate how CleaR performs as the model evolves, we compare our CleaR with baselines on BERT-Large, which is 3× larger than the base-sized model. Table 7 represents the evaluation results on SST-5 with different levels of symmetric noise. We observe CleaR still outperforms baselines on both peak and average accuracy by a large margin. These results demonstrate that our CleaR allows us to improve the model performance on noise label settings regardless of the model sizes.

Table 7: Peak and Average accuracy (%) on SST-5 under the BERT-large.

Method	Clean	40%		60%	
		Peak.	Avg.	Peak.	Avg.
Full Fine-tuning	55.1	51.3	43.7	48.6	37.0
PEFT methods					
Adapter	**55.5**	50.9	46.2	49.5	40.8
BitFit	55.4	51.7	49.8	50.5	48.2
Prompt	53.6	50.0	46.1	49.7	45.1
LoRA	54.5	52.4	48.8	49.8	46.1
PEFT methods with CleaR (ours)					
CleaR$_{\text{Adapter}}$	55.2	**53.1**	**52.2**	**53.5**	49.2
CleaR$_{\text{BitFit}}$	54.7	51.9	50.6	51.5	49.2
CleaR$_{\text{Prompt}}$	53.7	52.0	50.5	52.4	**50.5**
CleaR$_{\text{LoRA}}$	54.5	52.2	51.2	50.6	48.8

Table 8: The ratio of training parameters (%) for each PEFT method. Note that CleaR methods have the same number of trainable parameters.

Methods	Trainable Parameters (%)
Adapter (2019)	0.455%
BitFit (2022)	0.078%
Prompt (2022b)	0.024%
LoRA (2022)	0.111%

D CleaR on PEFT methods

In this paper, we apply CleaR to the existing PEFT methods. Specifically, we select the widely-used and different types of PEFT methods, which include Adapter (Houlsby et al., 2019), Prompt Tuning (Li and Liang, 2021), BitFit (Zaken et al., 2022), and LoRA (Hu et al., 2022). For each PEFT methods, we follow the commonly-used setup, and the ratio of trainable parameters are listed in Table 8. We name the adopted version of each method as CleaR$_{\text{Adapter}}$, CleaR$_{\text{Prompt}}$, CleaR$_{\text{BitFit}}$,

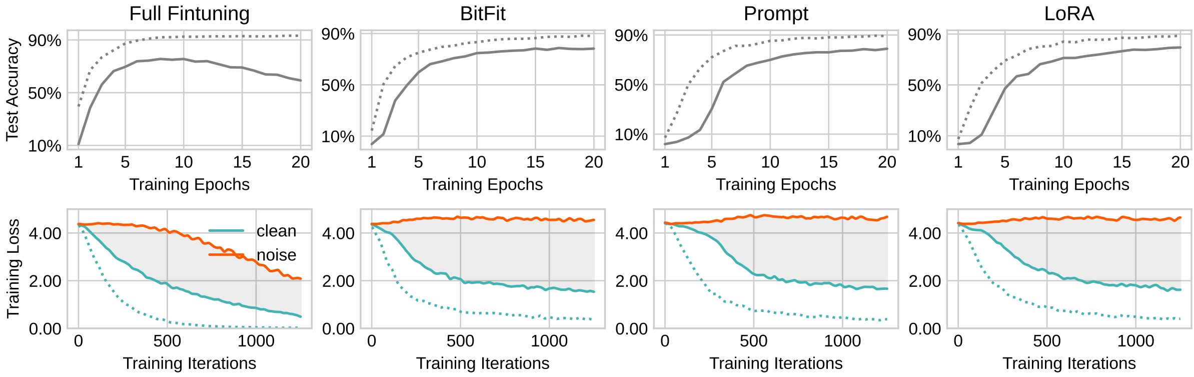
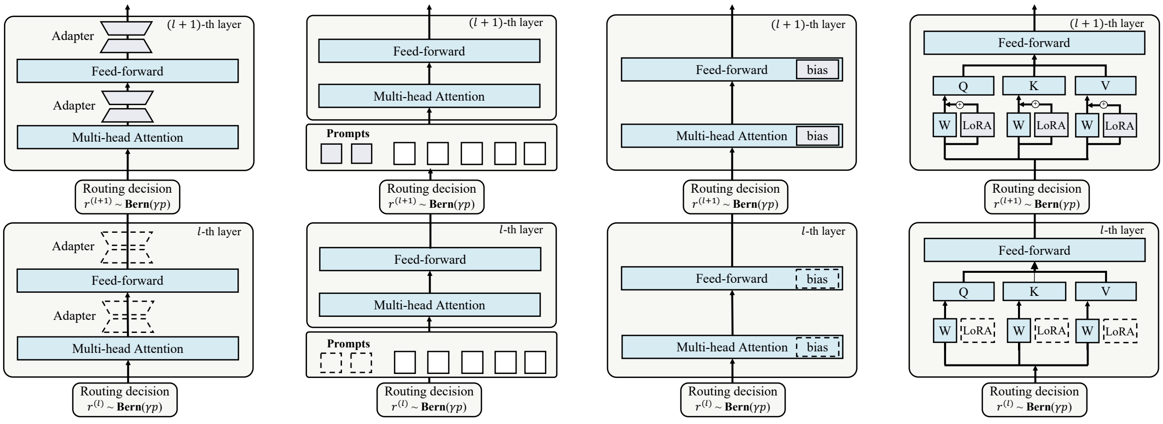


Figure 5: Comparison between PEFT methods and full fine-tuning on BANKING77 with symmetric noise (60%). Dashed lines represent the training accuracy and loss of clean samples on uncorrupted datasets (i.e. only clean samples).



(a) CleaR-Adapter

(b) Clear-Prompt

(c) CleaR-BitFit

(d) CleaR-LoRA

Figure 6: Detailed illustration of the CleaR adaptation to PEFT methods (e.g., Adapter, Prompt Tuning, BitFit, LoRA). Dashed lines indicate the unused modules, except for the $\text{CleaR}_{\text{BitFit}}$ that uses fixed pre-trained biases.

and $\text{CleaR}_{\text{LoRA}}$. For each method, we provide the graphical procedure in Figure 6, and its details are as follows:

$\text{CleaR}_{\text{Adapter}}$ Adapter (Houlsby et al., 2019) employs a bottleneck architecture, comprising down- and up-projection matrices with non-linearity applied to the bottleneck feature. These adapters are positioned above the multi-head self-attention and position-wise feed-forward layers. In the CleaR method, we group these two adapters and determine their usage through a sampling process.

$\text{CleaR}_{\text{Prompt}}$ We adopt P-Tuning v2 (Liu et al., 2022b) as a representative method for prompt tuning. This approach simply appends trainable prompts to the original input embeddings and fine-tunes these appended prompt embeddings during the fine-tuning phase. In the CleaR method, we group these prompt embeddings and decide whether or not the prompts are appended to the input. As a result, in $\text{CleaR}_{\text{Prompt}}$, the input length can vary based on the routing decision.

$\text{CleaR}_{\text{BitFit}}$ Different to other PEFT methods that introduce trainable weights (e.g., adapters, prompts), BitFit (Zaken et al., 2022) fine-tunes the existing biases in the PLMs. To apply routing to BitFit, we store the pre-trained biases, and the routing mechanism determines whether the training model utilizes the pre-trained (fixed) biases or the trainable biases.

$\text{CleaR}_{\text{LoRA}}$ LoRA (Hu et al., 2022) employs a bottleneck architecture similar to the adapter. However, the bottleneck in LoRA is situated within the weights for the attention matrices. In the CleaR method, we decide whether the linear weights for the attention mechanism (e.g., query, key, and value weights) incorporate trainable weights or not.

E Detailed Process for Generating Noisy Labels

We detail the process for generating noisy labels when the noise rate p is given.

- **Symmetric noise:** To generate this noise, we create the noise transition matrix $T \in R^{C \times C}$ where C is the number of classes. We then set a value of p to its diagonal elements and distribute the remaining probability $(1-p)$ to other non-diagonal elements. Based on the probability in the matrix, we flip the labels in training samples.
- **Asymmetric noise:** Similar to the symmetric noise, we create the noise transition matrix $T \in R^{C \times C}$ where C is the number of classes. We then set a value of p to its diagonal elements and assign the remaining probability $(1-p)$ to the next element of the diagonal values to implement single-flip noise (Qiao et al., 2022). Based on the probability in the matrix, we flip the labels in training samples.
- **Instance-dependent noise:** To generate instance-dependent noise, we first pre-train the classifier on the original dataset. We then select the two classes, which are the most confident u and the second most confident classes s, and calculate the distance of decision boundaries between the classes, i.e., $[f_u(x) - f_s(x)]^2$. Based on the distance, we define the noise function as $\tau = -\frac{1}{2}[f_u(x) - f_s(x)]^2 + \frac{1}{2}$. A smaller distance between the classes results in a larger flipping probability to the second most confident class s. Lastly, to control the degree of noisy labels, we multiply τ by a certain constant factor such that the final proportion of noise matches the pre-defined noise probability.

F Implementation Details and Setups

In this section, we detail to implement the baselines and our CleaR on various tasks.

PEFT implementation. We compare our CleaR with four strong baselines, which include Adapter (Houlsby et al., 2019), LoRA (Hu et al., 2022), BitFit (Zaken et al., 2022), and Prompt-tuning (Liu et al., 2022b). Specifically, we set the bottleneck dimension r for the Adapter and LoRA as 16 and 4, respectively. For LoRA, we only apply LoRA weights on query and value attention weights. Moreover, we fine-tune all bias parameters in transformer blocks for BitFit. For Prompt-tuning, we set the fixed length as 20 for prompts in each transformer layer, following P-tuning v2 (Liu et al., 2022b).

Hyper-parameters. For the hyper-parameters to fine-tune in CleaR, we select the best warmup epoch in $[3, 10]$ and clean probability weights γ in $[0.5, 1]$ corresponding to each PEFT method and task. We also use the number of forward $N = 5$ for constructing ensemble predictions on consistency regularization. we use the consistency regularization coefficient $\lambda = 1$ for all experiments. For other settings, we use Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We also train all models using a batch size of 32 and sweep the learning rates in {1e-4, 2e-4, 3e-4, 4e-4, 5e-4} for PEFT methods. For full fine-tuning, we select the best learning rates in {1e-5, 2e-5, 3e-5, 4e-5, 5e-5}. All models are fine-tuned for 20 epochs.

Hardware Details. We train all our models using four RTX 3090 GPUs. We utilize mixed-precision training (Micikevicius et al., 2018) to expedite the training procedure. All the implementations are performed using the PyTorch framework.

G Details for other NLL Methods

We compare our CleaR with the following NLL methods:

Co-teaching. Co-teaching (Han et al., 2018) trains two models simultaneously and lets each model select clean training samples (i.e., small-loss instances) for training each other model. Co-teaching framework gradually drops the noisy samples to prevent overfitting. To this end, they require an estimation of the noise level (τ), warm-up steps (T_k), and coefficient (c). We set the noise level τ as the ground truth noise ratio. We vary the warm-up steps T_k in {1500, 2000, 2500, 3000, 3500}, and select the best coefficient c from search range of $[1, 2]$ for each fine-tuning method.

SELC. SELC (Lu and He, 2022) trains the model using ensemble prediction based on historical model outputs to correct the noisy labels. Specifically, they first train their models with given labels until the turning point T, which represents the model would start overfitting on noisy levels. And then they combine the given labels with ensemble predictions with momentum α as the target. We estimate the turning point T by leveraging metrics, following (Lu and He, 2022), corresponding to each fine-tuning method. We also select the best momentum α by searching parameters from $[0.5, 1]$.

STGN. STGN (Wu et al., 2022) trains the model by reducing the disturbance on correct samples and increasing the perturbation on corrupted ones. Specifically, they utilize the standard deviation σ_f to perturb the gradient of loss and forgetting events threshold λ_f to separate corrupted data from correct ones. For full fine-tuning, we use the same setup in (Wu et al., 2022). For adapter-based tuning, we set $\sigma_{max} = 2\sigma_f$ and select the $\sigma_f = 7e-4$ and $\lambda_f = 4$.

We implement other NLL methods based on standard BERT with reference to their public code and make comparisons under the same setting. For other hyper-parameters (i.e., learning rate, and warmup epochs), we select the optimal values by searching on the same parameter space with baselines and CleaR.