Vision-Language Models are Strong Noisy Label Detectors

A PROJECT REPORT

Submitted by

Anvit Gupta (22114009)
Sarvasva Gupta (22114086)

for the fulfillment of
CSN-300: Lab Based Project

Under the guidance of

Prof. Pravendra Singh

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
Department of Computer Science and Engineering
Indian Institute of Technology Roorkee
Roorkee - 247667
2024–2025

Acknowledgements

We, the team, would like to extend our heartfelt gratitude to Prof. Pravendra Singh for providing us with the opportunity to explore and present on the research paper "Vision-Language Models are Strong Noisy Label Detectors" through both a presentation and this report. Their invaluable guidance and encouragement have been instrumental in helping us develop effective skills in presenting technical content and structuring a comprehensive report. We are particularly grateful for their support in refining our approach to conveying complex ideas clearly and cohesively.

We also appreciate the collaborative environment they fostered, which enabled us to work effectively as a group, share ideas, and support one another throughout the project. This experience has been deeply enriching, and we are grateful for the knowledge and skills we gained under their mentorship, not only in understanding the subject but also in communicating it proficiently.

Date: May 15, 2025

Abstract

Fine-tuning vision-language models for downstream tasks is often hindered by the presence of noisy labels in real-world datasets. The paper *"Vision-Language Models are Strong Noisy Label Detectors"* introduces DEFT, a Denoising Fine-Tuning framework that utilizes the robust alignment capabilities of pre-trained models like CLIP to detect and mitigate noisy labels. DEFT operates in two phases: detecting noisy samples via learned textual prompts (Phase 1) and fine-tuning on the cleaned dataset for classification (Phase 2).

In this lab-based project, we studied and reproduced the results of the DEFT framework. Minor modifications were made to the official implementation to ensure successful execution. Our reproduction closely matched the original results in terms of noisy label detection (precision and recall) and downstream classification accuracy. We further explored avenues for improvement. First, we introduced Gaussian noise into the already noisy dataset to evaluate robustness; however, the degradation was minimal, indicating that this may not be an effective stress test. Second, we proposed an alternative to discarding noisy samples: a pseudo-labelling approach that reassigns labels to the detected noisy data and includes them in the fine-tuning phase. This strategy led to an improvement in image classification accuracy.

Our results validate the effectiveness of DEFT and suggest that retaining and refining noisy data through pseudo-labelling can be a promising enhancement to the original framework.

1 Introduction

Fine-tuning vision-language models, such as CLIP, has demonstrated impressive performance in various downstream tasks, including few-shot learning, multi-label learning, and long-tail recognition. However, real-world datasets often contain noisy labels, which can hinder the effectiveness of such fine-tuning processes. While CLIP's zero-shot performance leverages the strong alignment between its visual and textual features, adapting it to domain-specific datasets usually requires fine-tuning on perfectly labeled data, a luxury not always available. Two common approaches for fine-tuning include full fine-tuning (FFT), which updates all model parameters, and parameter-efficient fine-tuning (PEFT), where only a few parameters are adjusted to adapt the model to the new data. However, noisy labels present a significant challenge during the fine-tuning process, as directly adapting to such data often degrades model performance.

The paper *"Vision-Language Models are Strong Noisy Label Detectors"* proposes a novel framework called DEFT (Denoising Fine-Tuning), designed to address this challenge by leveraging the pre-trained alignment of vision-language models to detect and mitigate noisy labels. DEFT introduces a two-phase process: the first phase involves identifying noisy labels by learning positive and negative textual prompts for each class, while the second phase fine-tunes the model on the curated clean data to enhance classification performance. The authors demonstrate that CLIP, and similar models, can be utilized as effective noisy label detectors, capitalizing on their strong visual-textual alignment.

In this lab-based project, we aimed to reproduce the results presented in the DEFT framework. Through modifications to the original GitHub code provided by the authors, we successfully replicated the results, achieving similar precision-recall scores for noisy label detection and image classification accuracy on the downstream task. Building upon this, we explored potential improvements to the method. Initially, we introduced Gaussian noise to the dataset to assess the robustness of the framework, but the results remained relatively stable, indicating the need for further exploration. In a subsequent modification, we implemented pseudo-labelling of the noisy dataset rather than discarding noisy samples. This improvement resulted in a notable increase in classification accuracy, showing that retaining and refining noisy labels can enhance model performance.

Our work not only reinforces the effectiveness of the DEFT framework for noisy label detection but also demonstrates that incorporating pseudo-labelling can provide significant improvements in downstream tasks. This study highlights the potential of refining noisy datasets in fine-tuning vision-language models and paves the way for future advancements in learning with imperfect labels.

- **Reproduction of Results:** Successfully reproduced the results from the paper using the provided codebase, achieving consistent precision-recall metrics for noisy label detection and image classification accuracy on the downstream task.

- **Exploration of Improvements:** Investigated the impact of adding Gaussian noise to the dataset, which did not lead to considerable degradation, suggesting that other improvements should be explored.

- **Implementation of Pseudo-labelling:** Replaced the noisy label removal strategy with pseudo-labelling of the detected noisy samples, which resulted in a noticeable improvement in classification accuracy.
- **Enhancement of DEFT Framework:** Demonstrated that incorporating pseudo-labelling can improve the DEFT framework's performance in practical applications, especially in scenarios with noisy labels.

2 Methodology

2.1 Dataset Description

We evaluate DEFT on both synthetic and real-world noisy label datasets. For synthetic datasets, we use CIFAR-100, Tiny-ImageNet, Stanford-Cars, and CUB-200-2011. Label noise is introduced using a noise transition matrix T, with two types of noise: symmetric noise (where labels are randomly flipped) and instance-dependent noise (where the corruption probability depends on the image). We experiment with different noise ratios r for each type, with symmetric noise at $r \in \{0.2, 0.4, 0.6\}$ and instance-dependent noise at $r \in \{0.2, 0.3, 0.4\}$.

For real-world datasets, we test DEFT on CIFAR-100N, Clothing1M, and WebVision. CIFAR-100N contains noisy labels from human annotations, Clothing1M has 1 million images from online shopping sites, and WebVision consists of 2.4 million images crawled from Google and Flickr, focusing on the first 50 classes.

2.2 Model Architecture

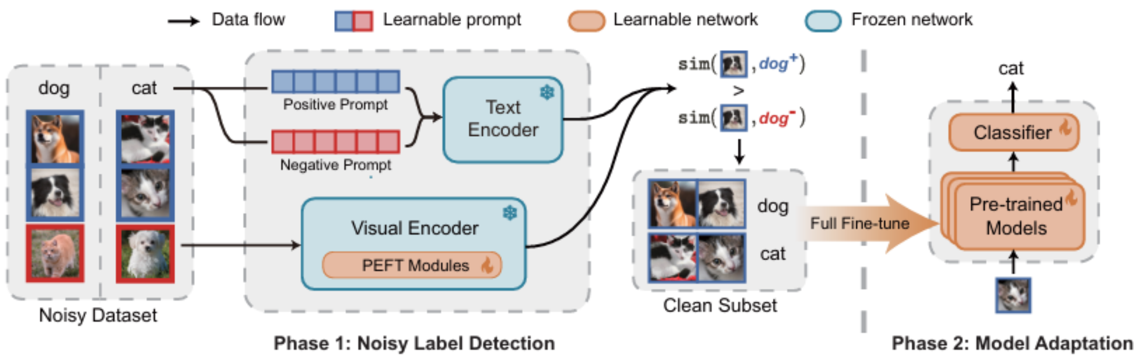

Figure 1: Overview of the DEFT framework. Phase 1 identifies noisy labels using dual textual prompts and PEFT modules. Phase 2 adapts the model using FFT on clean samples selected from Phase 1.

2.2.1 Phase 1: Identifying Noisy Labels with Dual Prompts (PEFT)

Given the empirically observed robustness of textual prompts to label noise, we leverage this property to identify corrupted labels in noisy datasets. Initially, samples with strong agreement between their visual and textual features are marked as clean. However, this naive strategy requires noise ratio estimates or manual thresholds.

To overcome this limitation, we adopt a dual-prompt design. For each class k, we define a pair of learnable prompts:

$$\text{prompt}_k^+ = [V]_1^+[V]_2^+ \dots [V]_M^+[\text{CLS}]_k \tag{1}$$
$$\text{prompt}_k^- = [V]_1^-[V]_2^- \dots [V]_M^-[\text{CLS}]_k \tag{2}$$

where $[V]_m^\pm \in \mathbb{R}^{512}$ are context vectors (in CLIP), M is the prompt length, and $[\text{CLS}]_k$ is the class name.

The positive prompt encourages high similarity with true image features, while the negative prompt learns a sample-dependent similarity threshold:

$$\phi_i = \text{sim}(I_i, T_k^-) \tag{3}$$

A sample (x_i, y_i) is added to the clean subset if:

$$\mathcal{D}_{\text{clean}} = \{(x_i, y_i) \mid \text{sim}(I_i, T_k^+) > \phi_i \quad \text{and} \quad y_i = k\} \tag{4}$$

This strategy is advantageous over loss-based filtering by: 1) Using data-driven thresholds that require no prior noise knowledge, and 2) Enhancing robustness to label noise via cross-modal alignment.

2.2.2 Phase 2: Model Adaptation on Clean Data (FFT)

Although positive prompts can be used directly for classification, their performance degrades on clean datasets. Thus, we introduce a second phase that performs full fine-tuning (FFT) on the pre-trained model using only the clean subset $\mathcal{D}_{\text{clean}}$.

A linear classifier is trained atop the fine-tuned image encoder using the classic cross-entropy loss:

$$\mathcal{L}_{\text{CE}}(x, y) = -\log\left(\frac{\exp(z_y)}{\sum_{k=1}^K \exp(z_k)}\right) \tag{5}$$

where $z = \{z_k\}_{k=1}^K$ are the logits for the K classes.

This phase removes the PEFT modules and is applicable to any pre-trained backbone, making the framework widely adaptable. FFT proves more effective than PEFT when trained on high-quality data, as validated in our experiments.

2.3 Implementation Details

The proposed approach utilizes a pre-trained CLIP model, with a Transformer-based text encoder and ViT-B/16 as the image encoder. Training is performed using the SGD optimizer, configured with a momentum of 0.9, a weight decay of 5×10^{-4}, and a batch size of 64. Both the noisy label detection phase and the model adaptation phase are trained for 10 epochs, with learning rates set to 3×10^{-2} and 5×10^{-4}, respectively.

To adapt the visual and textual encoders during noisy label detection, the method incorporates Visual Prompt Tuning (VPT) for the image encoder and Context Optimization (CoOp) for the text encoder. A warm-up phase of one epoch is applied across all datasets to stabilize training. For real-world noisy datasets, the condition used to identify clean samples is enhanced by enforcing consistency between the model's predicted label and the original training label, accounting for the more complex noise patterns commonly observed in such data.

All experiments are conducted on a single NVIDIA GeForce RTX 3090 GPU. /subsection Evaluation Metrics To assess the effectiveness of clean sample selection, the paper uses precision and recall. Precision measures the proportion of correctly identified clean samples within the selected set, while recall reflects how many of the actual clean samples were successfully retrieved from the noisy dataset. For the image classification task, the performance is evaluated using two metrics: the best test accuracy achieved during training and the final test accuracy obtained at the conclusion of training.

3 Reproduction of Results

3.1 Challenges Faced during Reproduction

- Removed the libraries `torch` and `torchvision` from `requirements.txt` and manually installed them with a command to match CUDA 11.8.
- Resolved an import error in `./model/load_clip.py` by changing the import statement from `from clip import clip` to `from .clip import clip`.
- Manually downloaded the CIFAR-100 dataset and placed it in the correct directory.
- In `main_phase2.py`, explicitly set the `weights_only` argument to `False` to align with the default behavior of PyTorch 2.0.
- Downloaded the pretrained weights, `CIFAR-100_human.pt`, from the UCSC-REAL repository and placed them in the `data/cifar-100/` directory.
- Added the argument `weights_only=False` in both `dataloader/dataloader_cifar.py` and `main_real_phase2.py` to ensure compatibility with the model.

3.2 Comparison with the Original Results

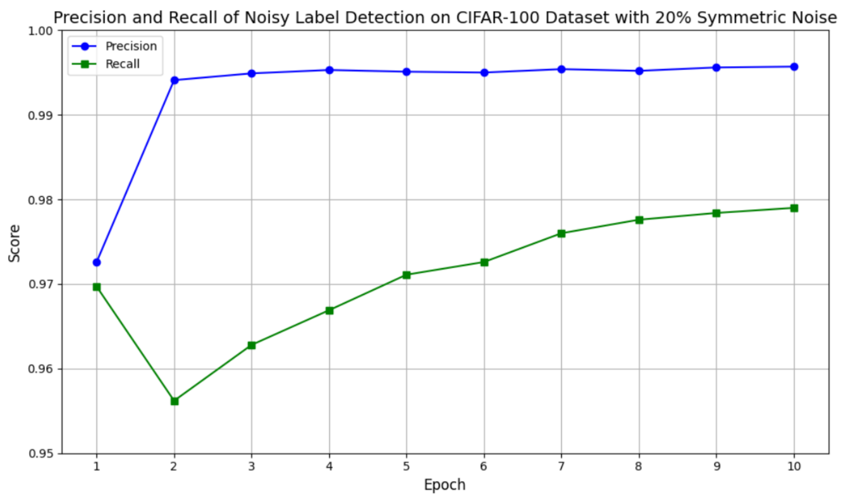

Figure 2: Precision and Recall Graph 1

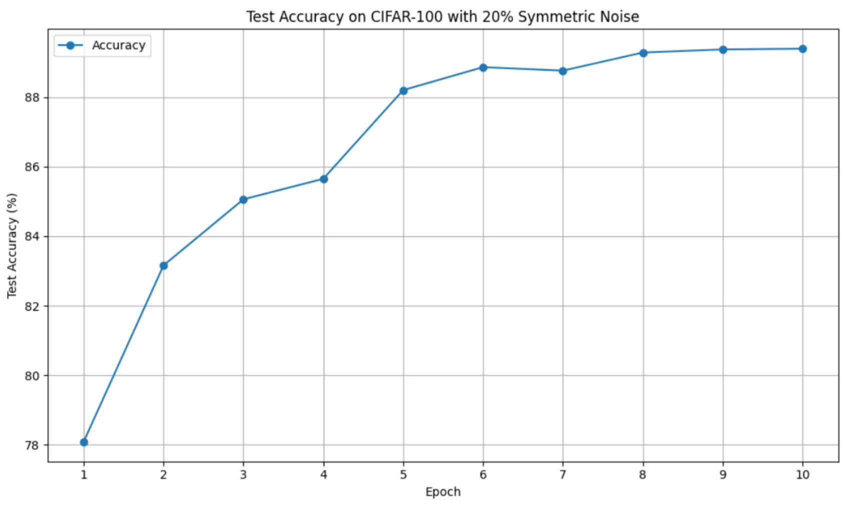

Figure 3: Image Classification Accuracy Graph 1

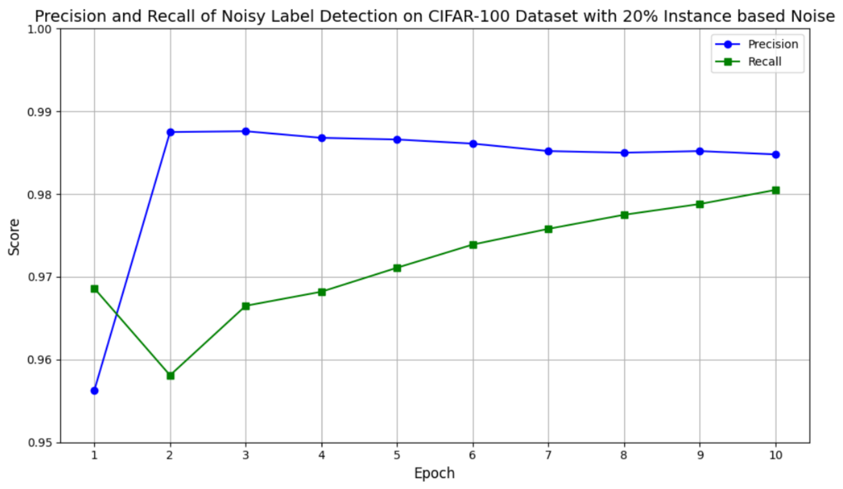

Figure 4: Precision and Recall Graph 2

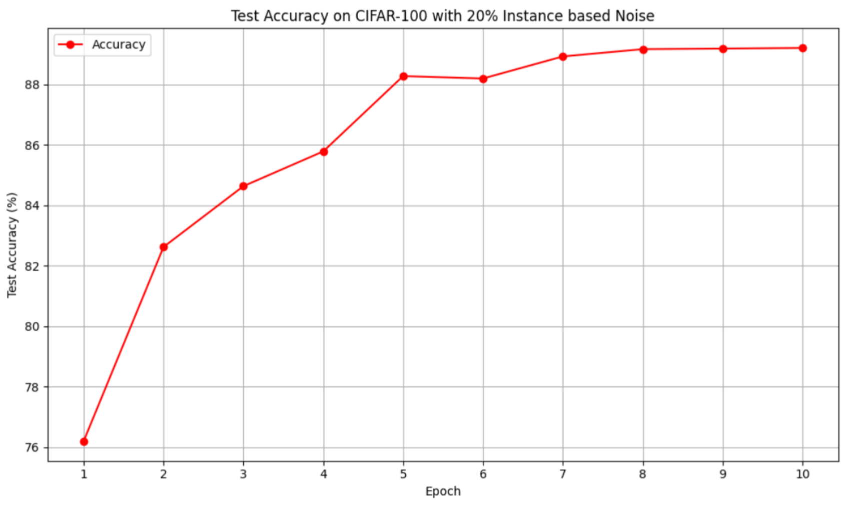

Figure 5: Image Classification Accuracy Graph 2

	Sym. 0.2		Sym. 0.4		Sym. 0.6		Ins. 0.2		Ins. 0.3		Ins. 0.4	
Method	Best Acc.	Last Acc.	Best Acc.	Last Acc.	Best Acc.	Last Acc.	Best Acc.	Last Acc.	Best Acc.	Last Acc.	Best Acc.	Last Acc.
DEFT	87.50	87.50	86.69	86.69	84.00	83.88	87.58	87.58	85.68	85.64	83.73	83.71
Reproduced Results	86.63	86.63	85.94	85.94	84.85	84.85	86.32	86.28	85.22	85.22	82.59	82.26

Table 1: Reproduced Results: Comparison of Best and Last classification accuracies under different symmetric and instance-dependent noise ratios on CIFAR-100.

	Sym. 0.2		Sym. 0.4		Sym. 0.6		Ins. 0.2		Ins. 0.3		Ins. 0.4	
Method	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
DEFT	99.51	97.77	98.75	97.91	97.04	97.27	98.47	97.88	96.32	97.63	94.08	95.28
Reproduced Results	99.57	97.90	98.77	97.65	96.91	97.49	98.48	98.05	96.45	97.82	89.07	97.61

Table 2: Reproduced Results: Comparison of Precision and Recall of Noisy-Label Detection under different symmetric and instance-dependent noise ratios on CIFAR-100.

We successfully reproduced the results for the CIFAR-100 dataset under both symmetric noise conditions with noise ratios $r = \{0.2, 0.4, 0.6\}$ and instance-based noise with noise ratios $r = \{0.2, 0.3, 0.4\}$. As observed, the precision, recall, and image classification accuracies closely match those of the original results, confirming the reproducibility of the model's performance under these noisy conditions.

4 Identified Improvements

4.1 Limitations in the Original Work

While the paper *"Vision-Language Models are Strong Noisy Label Detectors"* presents a robust framework for detecting noisy labels using vision-language models like CLIP, it has a few notable limitations that leave room for further exploration.

	Sym. 0.2	
Method	Best Acc.	Last Acc.
DeFT	89.38	89.35
Reproduced Results (without Gaussian Noise)	89.39	89.39
Results with 1% Gaussian Noise	88.98	88.98

Table 3: Results of Noisy Dataset with Noisy Labels

Firstly, the study focuses exclusively on detecting label noise, assuming the input images themselves are clean. In real-world scenarios, however, data often suffers from multiple sources of corruption, including input-level noise such as Gaussian blur, compression artifacts, or sensor errors. The effectiveness of vision-language models in detecting noisy labels when the input images themselves are noisy remains unexplored.

Secondly, the original work treats noisy data points as samples to be discarded after detection. While this helps in training on a cleaner subset, it may lead to underutilization of potentially informative data. Alternative strategies, such as pseudo-labelling or re-weighting noisy samples, could enable the model to leverage these points more effectively, thereby improving generalization and robustness.

4.2 Proposed Improvements

Based on the limitations identified in the original work, we propose the following directions to enhance the utility and robustness of the noisy label detection framework:

Firstly, we introduce controlled Gaussian noise into the training and/or testing datasets to investigate the effect of input-level corruption on the model's performance. The results, as shown in the Table, reveal that the model remains robust even under mild to moderate input noise. Notably, the performance degradation remains minimal with as little as 1% Gaussian noise added to only the training set, indicating the model's resilience. This challenges the implicit assumption in the original paper that input-level noise may severely affect classification accuracy.

Secondly, instead of discarding the noisy samples identified during the first phase, we propose re-integrating them using a pseudo-labelling strategy. Specifically, a vision-language model such as CLIP can be employed to assign new labels to the filtered-out samples. These pseudo-labelled instances can then be used during the second phase of training (Full Fine-Tuning), potentially recovering lost information and further improving performance. This approach allows for a more data-efficient training process and opens up possibilities for semi-supervised learning in noisy environments.

5 Implementation of the selected Improvement

5.1 Description of the Improvement Undertaken

To explore the usability of noisy data rather than discarding it, we implemented a pseudo-labelling strategy during Phase 2 (FFT) of the DeFT pipeline. The clean subset identified in Phase 1 was retained, and the noisy subset was augmented using pseudo-labels generated from a CLIP model. Specifically, we used the CLIP ViT-B/16 architecture to assign semantic class labels to the data points identified as noisy.

The process works by creating a prompt-based class description for each CIFAR-100 class (e.g., "cat", "truck", "maple tree"), encoding the image and prompts using CLIP, and then selecting the label whose prompt has the highest cosine similarity with the image representation. These pseudo-labels were used in place of discarding noisy points, thereby leveraging all data during Phase 2 training.

Algorithm 1 Pseudo-Labelling Noisy Data using CLIP

```
1:  Input: Noisy dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, clean index list $\mathcal{I}_{clean}$ from Phase 1, class list $\mathcal{C}$
2:  Load CLIP ViT-B/16 model
3:  Generate prompts $\mathcal{P} = \{\texttt{"class\_name"} \mid \texttt{class\_name} \in \mathcal{C}\}$
4:  for $i = 1$ to $N$ do
5:      if $\mathcal{I}_{clean}[i] == 1$ then
6:          continue
7:      else
8:          Encode image: $v_i = \texttt{CLIP.encode\_image}(x_i)$
9:          Encode all prompts: $\{p_j\} = \texttt{CLIP.encode\_text}(\mathcal{P})$
10:         Compute similarity: $s_j = \cos(v_i, p_j), \forall j$
11:         Assign pseudo-label: $y_i = \arg\max_j s_j$
12:     end if
13: end for
14: Use $\mathcal{D}_{clean} \cup \mathcal{D}_{pseudo}$ for Phase 2 training
```

5.2 Model Architecture

The overall DeFT framework remains unchanged, with the addition of a pseudo-labelling block introduced between Phase 1 (Parameter-Efficient Fine-Tuning) and Phase 2 (Full Fine-Tuning). This new module takes the noisy subset identified in Phase 1 and assigns pseudo-labels using CLIP ViT-B/16, as described previously.

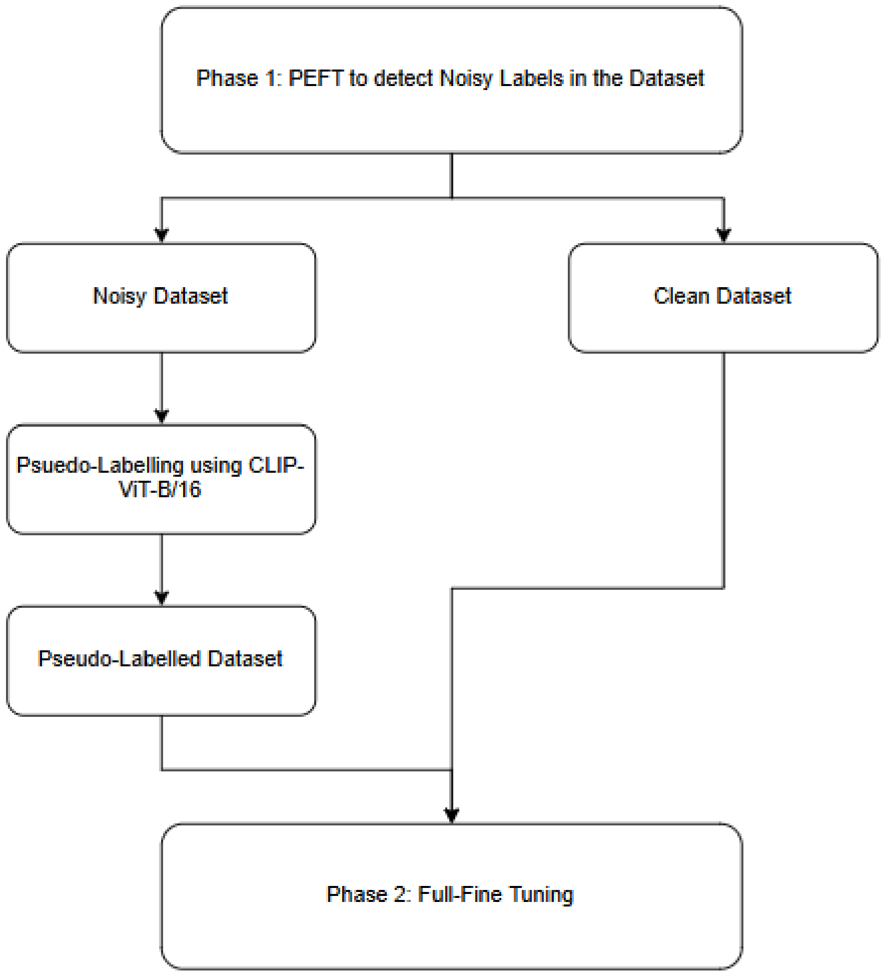
Phase 1: PEFT to detect Noisy Labels in the Dataset
Noisy Dataset
Clean Dataset
Psuedo-Labelling using CLIP-ViT-B/16
Pseudo-Labelled Dataset
Phase 2: Full-Fine Tuning

Figure 6: Modified DeFT architecture with CLIP-based Pseudo-Labelling block inserted between Phase 1 and Phase 2.

This modified pipeline allows the model to utilize all available data—clean and pseudo-labelled—during Phase 2 training.

6 Implementation Details

The implementation of the proposed improvement was carried out using the Google Colab free GPU.

The process was divided into two distinct phases: Phase 1 (Parameter-Efficient Fine Tuning) and Phase 2 (Full-Fine Tuning, FFT) with pseudo-labelling.

In Phase 1, the model architecture remained untouched, as this phase primarily focused on the noisy label detection task. The original setup from the paper was preserved in this phase to maintain consistency.

In Phase 2, the pseudo-labelling procedure was applied, followed by FFT to refine the feature representations and enhance model performance.

The main objective of the experiment was to evaluate the impact of pseudo-labelling on the image classification accuracy. This evaluation metric was selected to assess how effectively the model handled noisy labels after augmenting the dataset with pseudo-generated labels, thereby improving overall classification accuracy.

7 Results and Analysis

7.1 Pseudo-Labelling Results

The following images show the results of pseudo-labelling applied to the dataset. These images illustrate the effect of pseudo-labelling on the dataset and its impact on the model's performance.

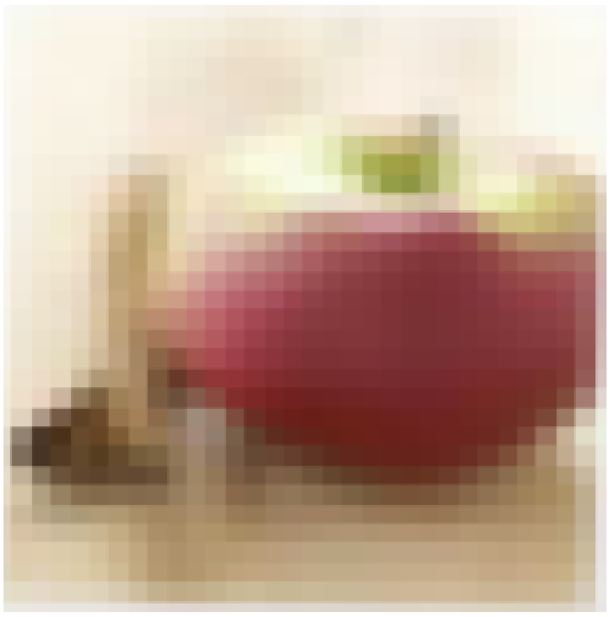
Figure 7: Pseudo-labeling Example 1: Pseudo-labeled as "apple" by CLIP-ViT-B/16

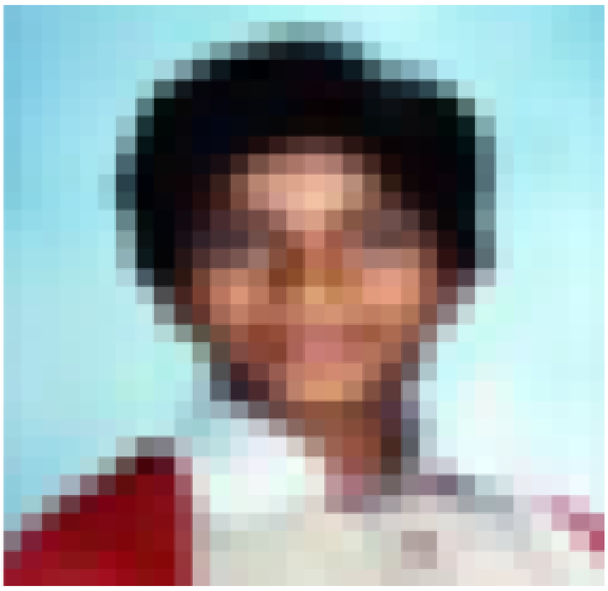

Figure 8: Pseudo-labeling Example 2: Pseudo-labeled as "boy" by CLIP-ViT-B/16

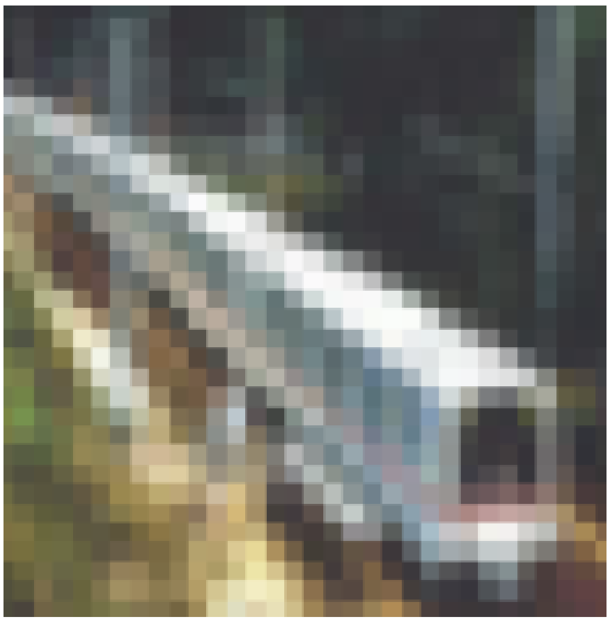

Figure 9: Pseudo-labeling Example 3: Pseudo-labeled as "train" by CLIP-ViT-B/16

7.2 Image-Classification Accuracy over Epochs

The following graph depicts the image classification accuracy over the course of the training epochs. This graph allows us to observe how the model's performance improves as it processes more data during training.

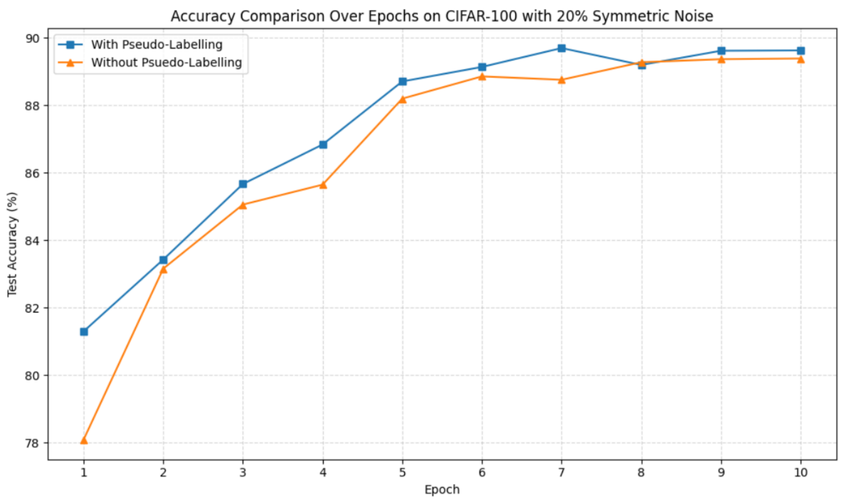

Figure 10: Image Classification Accuracy over Epochs

7.3 Final Accuracies

The table below summarizes the final image classification accuracies obtained after applying pseudo-labelling. These values represent the model's performance at the conclusion of the training process.

	Sym. 0.2	
Method	Best Acc.	Last Acc.
DEFT	89.38	89.35
Reproduced Results (without Gaussian noise)	89.39	89.39
Pseudo Labelling with 100 classes	89.70	89.63

Table 4: Final Accuracy Comparison under symmetric noise ratio 0.2 on CIFAR-100.

8 Future Work

In future work, one can plan to explore the following areas:

8.1 Noisy Label Detection in Datasets with Gaussian Noise

One can aim to develop a framework for detecting noisy labels in datasets that already contain Gaussian noise in the input images. This will help improve model performance in noisy real-world scenarios by addressing both label noise and input noise.

8.2 Exploration of Various Pseudo-labelling Approaches

We also plan to investigate different pseudo-labelling methods, such as confidence-based, entropy-based, and generative approaches. This exploration will help identify more effective techniques for improving model accuracy, particularly in semi-supervised learning tasks with noisy labels.

References

[1] Wei, H., Wang, H., & Lipton, Z. C. (2024). *Vision-Language Models are Strong Noisy Label Detectors*. Available at: https://arxiv.org/abs/2409.19696

[2] DeFT Official Implementation — GitHub Repository. Available at: https://github.com/HotanLee/DeFT

[3] OpenReview — Open Peer Review Platform for Machine Learning Conferences. Available at: https://openreview.net/

[4] NeurIPS — Conference on Neural Information Processing Systems. Available at: https://neurips.cc/

[5] ICLR — International Conference on Learning Representations. Available at: https://iclr.cc/

[6] CVPR 2025 — IEEE/CVF Conference on Computer Vision and Pattern Recognition. Available at: https://cvpr.thecvf.com/Conferences/2025