GNR 650 (Autumn 2024): Assignment 1

Label Noise Learning using ViT

Soumen Mondal (23m2157)
Siddhant Gole (23m2154)

September 15, 2024

1 Introduction

This project focuses on addressing the challenging task of label noise learning in computer vision that involves learning to classify images with correct labels even if the image had a noisy label tagged with it during training. The motivation stems from the fact that training deep learning models require huge amount of annotated data, these annotations are prone to human errors. Thus, there is a need to make deep learning models robust to these errors i.e., they assign correct labels even if at training, the image was annotated with a noisy one. The primary objective involves handling the noisy labels before training by creating a clean set of labels using the `Turtle` architecture proposed in the ICML'24 paper 'Let Go of Your Labels with Unsupervised Transfer' [1] then fine tuning a ViT on the cleaned set of labels. This approach was able to handle 40% symmetric noise added to the CIFAR-100 dataset and achieved 88% accuracy on the test data.

2 Methodology

2.1 Dataset

The dataset utilized for training and evaluation was the CIFAR-100 dataset consisting of 100 distinct classes with 600 images belonging to each class (500 train and 100 test images). Noise was introduced in the dataset by randomly flipping the labels of images in each class. 40% noise was introduced in the entire dataset. The dataset was made available by flattening the (32 x 32 x 3) images to a 3072 vector and the corresponding noisy labels as the ground truth in the train set with 50000 images and the test set had 10000 flattened image vectors with no labels.

2.2 Method

We aim to address the problem of training a classification model on a dataset where the training labels are corrupted by random asymmetric noise. The dataset contains 50,000 training images with noisy labels. The number of possible classes is 100, and we do not have access to the true labels for the training set. The challenge is to recover reliable labels from the noisy training set without knowing the true labels.

To address this issue, we utilize a method called `Turtle` [1], which generates pseudo-labels for the training images. `Turtle` is based on sophisticated methodologies combining CLIP and DINO models, and it requires only the input images and the number of classes to generate

pseudo-labels. Importantly, `Turtle` does not need to know the original noisy labels from the dataset. However, the pseudo-labels generated by `Turtle` do not necessarily correspond to the true labels, as there may be a permutation between the pseudo-labels and the true labels (i.e., pseudo-label 0 could correspond to true label 5, etc.). Furthermore, since the original labels are noisy, applying methods such as the Hungarian assignment would not be effective, as they would attempt to match the noisy labels rather than the true labels.

To overcome this challenge, we propose a majority voting strategy that aligns the pseudo-labels generated by `Turtle` with the noisy training labels, assuming that the majority of the noisy labels in a given pseudo-class will correspond to the true class. The steps of our method are as follows:

1. For each pseudo-label $p \in \{0, 1, \ldots, 99\}$ generated by `Turtle`, we collect all training images that have been assigned this pseudo-label. Given that there are 50,000 training images and 100 pseudo-labels, we expect approximately 500 images to be assigned to each pseudo-label.

2. For the set of images corresponding to each pseudo-label p, we consider the noisy labels provided in the original training set. We plot a histogram of these noisy labels. This histogram shows the distribution of noisy labels assigned to the images grouped under pseudo-label p.

3. We then compute the mode of the histogram, which corresponds to the most frequent noisy label for the images associated with pseudo-label p. The idea is that the majority of the images with pseudo-label p should have the same true label, even though the labels are noisy.

4. We assign this majority noisy label as the final label for all the images under pseudo-label p. In other words, if the mode of the noisy labels for pseudo-label p is 7, then we assume that pseudo-label p corresponds to the true label 7, and we relabel all the images in this pseudo-class with label 7.

5. This process is repeated for each pseudo-label, resulting in a final de-noised set of corrected labels for the training images. Most importantly, the final true label comes from the noisy labels after majority voting on noisy labels so we expect there would be some error in the labels in the de-noised set as well.

This majority voting approach is based on the assumption that the noise in the labels is not overwhelming (i.e., less than 50%), and thus the majority of images assigned to each pseudo-label should correspond to the same true label. By leveraging the pseudo-labels from `Turtle` and the noisy training labels, we can align the pseudo-labels with the true labels without requiring direct access to the true labels.

3 Experimental Setup

3.1 Architecture Details

The architecture utilized for the label de-noising and image classification tasks is composed of multiple key components, each playing a crucial role in improving label quality and classification performance. The architecture follows a structured pipeline involving unsupervised clustering, label refinement, and fine-tuning. The detailed process is explained below:

- **Feature Embedding Blocks (FEBs):** For the unsupervised label denoising task, feature extraction from CIFAR-100 images is first performed using pre-trained Vision Transformers (ViT-L/14) and DINOv2 (ViT-g/14). The Feature Embedding Blocks (FEBs) in these models generate robust, high-dimensional representations of the input images, which are used for clustering. These embeddings ensure spatial integrity and capture significant semantic information about the images, making them suitable for unsupervised clustering in the `Turtle` framework. By leveraging both ViT-L/14 and DINOv2 representations, `Turtle` captures a richer set of features, enhancing the clustering process.

- **Unsupervised Label Denoising via `Turtle`:** The `Turtle` framework is applied to cluster the images in the joint representation space of both ViT-L/14 and DINOv2. `Turtle` searches for pseudo-label assignments that maximize the margin of linear classifiers in the combined representation spaces of these models. This allows the framework to group images into meaningful clusters by leveraging the complementary strengths of both representations. After clustering, a majority voting mechanism (on noisy labels) is employed to assign ground truth labels to each cluster. The majority label in each cluster is propagated to all images within that cluster, thereby creating a clean set of labels for the entire dataset.

- **ViT B-16 Fine-Tuning for Image Classification:** After obtaining clean labels through the `Turtle` framework, the de-noised CIFAR-100 dataset is used for fine-tuning a standard pre-trained ViT B-16 model (86 Million parameters). The final classification head which was over 1000 output neurons is replaced by a new classification head over 100 output neurons so that it can give classes between 0 to 99. The newly cleaned labels, obtained from the clustering process was used to replace the noisy labels in the dataset. The fine-tuning process involves training the entire ViT B-16 model on the de-noised dataset, ensuring that all parameters (full fine-tuning) are updated to improve classification accuracy. For experimentation purpose, we will also finetune the model for last 3 layers and last 1 layer.

This architecture pipeline efficiently addresses the issue of label noise by improving the quality of the dataset through `Turtle`'s unsupervised learning capabilities, followed by fine-tuning a standard ViT B-16 model to achieve high classification accuracy on test set.

3.2 Training Configuration

In this section, we describe the training configuration used for fine-tuning the ViT-B/16 model on thede-noised CIFAR-100 dataset. The model is pre-trained on ImageNet21k and fine-tuned with the following hyperparameters and training settings:

3.2.1 Batch Size and Number of Epochs:

The batch size used for training is set to 128. The data is split into batches of 128 samples, and the model is updated after each batch during the training phase. We have selected two number of epochs for fine-tuning the model.

3.2.2 Learning Rate:

The initial learning rate is set to 0.0001. A `CosineAnnealingWarmRestarts` learning rate scheduler is applied during training to dynamically adjust the learning rate over epochs. The scheduler is configured with a minimum learning rate of 1×10^{-5} and $T_{mult} = 1$. The cosine annealing

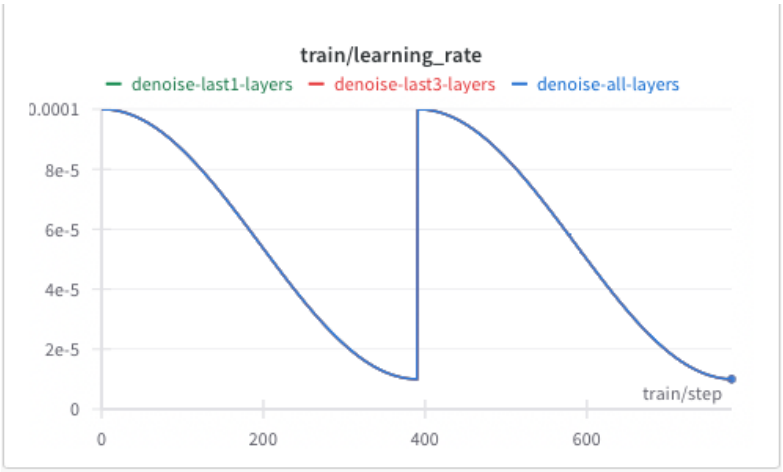

Figure 1: Learning Rate Schedule for CosineAnnealingWarmRestarts

restarts occur after one epoch, allowing the learning rate to warm up and then gradually decrease throughout the training process as shown in Figure 1.

3.2.3 Optimizer:

The optimizer used for this training process is `AdamW` (Adam with weight decay). AdamW is a variant of the Adam optimizer that decouples the weight decay from the gradient-based update. This allows us to apply weight decay for regularization without altering the optimizer's behavior. The weight decay is set to 0.1 to prevent overfitting, while the learning rate follows the scheduler described above.

3.2.4 Gradient Clipping:

Gradient clipping is also employed with a maximum norm of 10.0 to prevent the gradients from becoming too large, ensuring numerical stability during backpropagation.

3.3 Evaluation Metrics

Explanation of the evaluation metrics used to assess the model's performance:

- **Accuracy:** Accuracy is a widely used metric to assess the performance of classification models. It measures the percentage of correctly classified images out of the total number of images. The accuracy metric is calculated as:

$$\text{Accuracy} = \left(\frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \right) \times 100 \tag{1}$$

  Higher accuracy indicates that the model is effectively classifying the images into their respective categories.

- **Loss:** The training process incorporates standard Cross-Entropy Loss for fine-tuning the Vision Transformer (ViT B-16) on the de-noised CIFAR-100 dataset. After the label refinement process via `Turtle`, the de-noised labels are used to fine-tune a standard ViT B-16 model. The Cross-Entropy Loss is applied during this stage to minimize the error in

classification. This loss function measures the discrepancy between the true labels and the predicted label distribution for each image:

$$L_{\text{CE}} = -\sum_{j=1}^{|B|}\sum_{i=1}^{C} y_i^j \log(\hat{y}_i^j)$$

where:

- y_i^j is the true label (one-hot encoded) for the j-th image and i-th class.
- $\hat{y}_i^j$ is the predicted probability of the model for the j-th image and i-th class.
- $|B|$ is the batch size and C is the number of classes.

By minimizing the Cross-Entropy Loss, the fine-tuned ViT B-16 model learns to classify images more accurately, with the de-noised labels further improving its ability to generalize on the CIFAR-100 dataset.

4 Results

4.1 Training Process

The training process for our approach involved monitoring both the training loss and accuracy over the epochs to assess the model's convergence and performance. The de-noised CIFAR-100 dataset, with labels refined using the `Turtle` framework, was used for fine-tuning the Vision Transformer (ViT B-16) model. Training curves were plotted against the number of optimizer steps to track performance. Both the training loss and accuracy demonstrated a clear trend over the epochs, indicating that the model was learning from the training data and improving its classification performance. We trained the model for 2 epochs. We have kept the same training configuration for all the fine-tuning experiments with different layers.

As shown in Figure 2, the training loss consistently decreases as the model minimizes classification errors, while the accuracy gradually improves, suggesting that the model is effectively learning to classify the images more accurately with the cleaned labels. Additionally, the convergence of both the loss and accuracy curves indicates that the model is not overfitting to the training data.

Utilizing the training set as a validation set is problematic. Given that the training set contains noisy labels, using it to monitor validation accuracy would not provide a meaningful evaluation of the model's generalization ability. Consequently, we refrain from using the noisy training set as a substitute for validation to avoid potential misinterpretation of the model's performance. However, for brevity, we still show the training accuracy but it should be noted that this metric tells how well the model is learning the de-noised set. Improving training accuracy should not be the sole objective as it will learn the de-noised set which still contains some noisy labels so during test time it may not generalize well.

4.2 Testing

Since the training dataset contains noisy labels and cannot be used to reliably monitor the model's performance during training, we opt for an alternative approach. In this work, we test

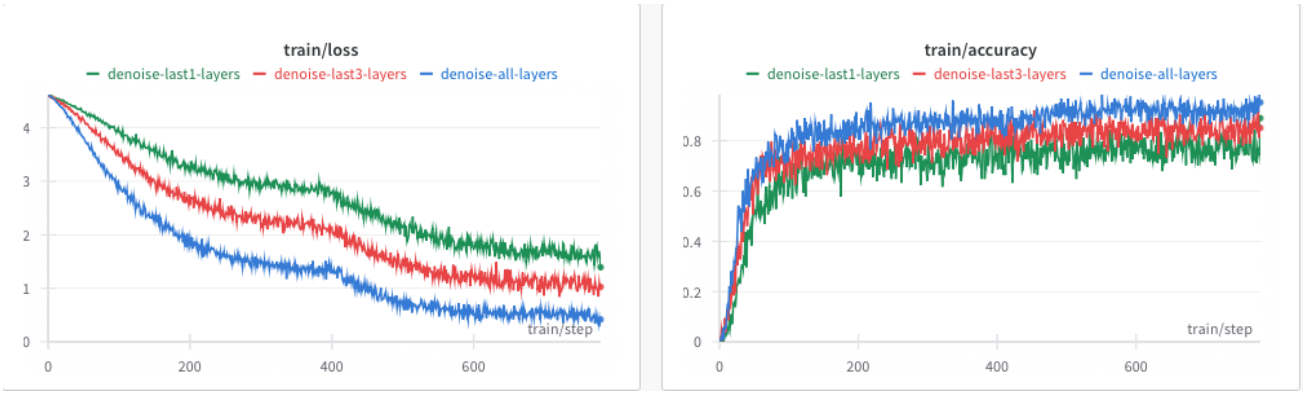

Figure 2: Training loss and accuracy. Note that the loss and accuracy are computed on the de-noised dataset so it might not reflect the true pattern as it still contains some noisy labels.

Figure 3: Test loss and accuracy. Note that the loss and accuracy are computed on the publicly available CIFAR100 test dataset. Hence, it should not be used in practice to tune the hyper-parameters of the model. It is shown for the sake of completeness and for illustrative purposes only.

the model directly on the clean test set after each epoch. The CIFAR-100 test set, which is publicly available and contains 10,000 images with clean labels, provides us with an opportunity to evaluate the model's accuracy in an illustrative manner.

This testing procedure allows us to observe how the model's accuracy evolves across epochs. However, it is important to note that this approach is strictly for illustrative purposes and should not be considered a standard practice in real-world scenarios. In a typical setting, the test set is reserved for final evaluation, and frequent testing would introduce potential biases during model development. In practice, the test set should be used only once at the end of the training process to provide an unbiased estimate of the model's generalization ability. Thus, while we monitor the model's accuracy on the test set after each epoch in this experiment, it is merely to demonstrate the trend in accuracy across epochs, and should not be interpreted as a recommended validation approach. The test set performance is shown in Figure 3.

4.3 Performance Evaluation

After training, the performance of the Vision Transformer (ViT B-16) model was evaluated on the test set of 10,000 CIFAR-100 images to assess its accuracy in classifying unseen examples.

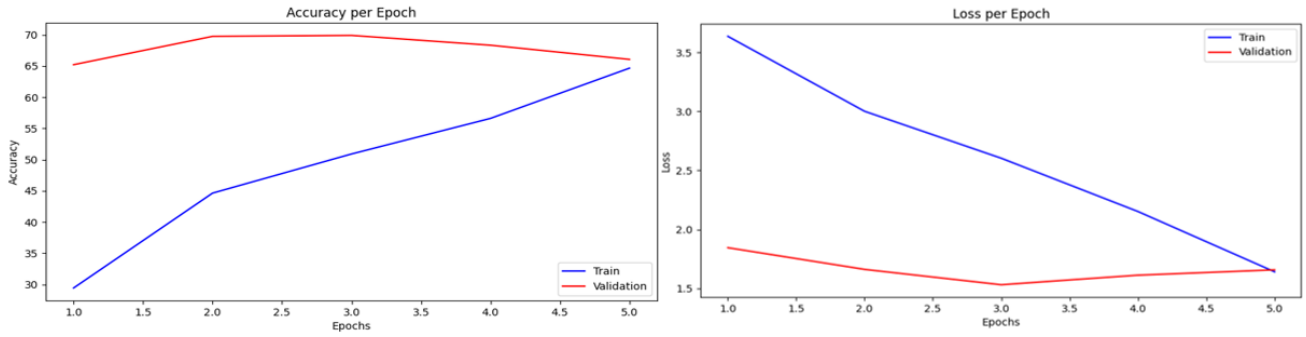

Figure 4: Loss and accuracy plots for Resnet-18. Note that the validation here actually refers to the test set performance at the end of each epoch.

The final test accuracy achieved on the test set was 88%, indicating that the model generalizes well to new examples.

The results of the training process and performance evaluation demonstrate the effectiveness of the label de-noising via the `Turtle` framework. By cleaning the noisy labels and fine-tuning the ViT B-16 model, the approach achieves high accuracy while maintaining parameter efficiency. This makes it a suitable choice for image classification tasks in datasets with noisy labels.

Model Name	**# of Parameters**	**Train Accuracy**	**Number of Epochs**
ViT B/16	86 Million	0.91	2
ResNet 18	11 Million	0.64	5

Table 1: Classification results on train set at the end of last epoch.

Model Name	**# of Parameters**	**Test Accuracy**	**Number of Epochs**
ViT B/16	86 Million	0.88	2
ResNet 18	11 Million	0.66	5

Table 2: Classification results on test set at the end of last epoch.

4.4 Comparison with Resnet-18

To compare the performance of Vision Transformers (ViTs) with CNN-based architectures, the denoised dataset was also fine-tuned on a ResNet-18 architecture, where the classification head was replaced. ResNet-18 consists of 18 layers: 17 convolutional layers and one fully connected layer, making it relatively lightweight compared to deeper ResNet variants like ResNet-50. The key innovation in ResNet-18 is the use of residual blocks that allow gradients to flow more easily during backpropagation. Each residual block contains a skip connection that bypasses one or more layers, facilitating the training of deeper networks. The model was trained for a total of 5 epochs. Both the training and test accuracies gradually increased, reaching 64.66% and 66.06%, respectively. This suggests that the performance was still affected by some noisy labels that remained even after obtaining the clean set. The corresponding plots are shown in Figure 4.

4.5 Relevant Links of Results

The following links provide access to various resources related to the results of our experiments. We have kept all of our codes in GitHub.

- **Code:** The Python files for training, validation, testing and plotting can be found on https://github.com/soumenkm/IITB-GNR650-ADLCV/tree/main/Assignment1
- **Checkpoint:** The final checkpoint for ViT B/16 (ckpt-ep-1.pth, epoch index starts from 0!) and ResNet 18 (resnet18-cifar100-finetuned.pth at the end of epoch 4, epoch index starts from 0!) can be found on Google Drive.

4.6 Explanation of Codes

We have used multiple Python files to simplify the training and testing processes. The functionality of each of the Python files available in the main GitHub repository is explained below:

- dataset.py: Defines the class responsible for loading and preprocessing the CIFAR-100 dataset.
- denoise.py: Contains the methods for loading and applying the `Turtle` framework to denoise the noisy labels.
- train.py: Provides the training loop and optimization process for fine-tuning the Vision Transformer (ViT B-16) model. This file includes code for defining the loss function (Cross Entropy Loss), adjusting learning rates, and tracking the model's accuracy and loss over epochs.
- train.py: Contains functions or scripts for fine-tuning the ViT B/16 model on the training data.
- eval.py: Includes functions for evaluating the model's performance after fine-tuning. This file is responsible for calculating metrics such as accuracy and loss on the test set and generating performance summaries.
- resnet.py: Code for fine-tuning Resnet-18 on the denoised set.

5 Conclusion

In conclusion, the experimentation with the `Turtle` framework for label denoising and fine-tuning the Vision Transformer (ViT B-16) on the CIFAR-100 dataset demonstrated promising results. Through rigorous training and evaluation, the model showcased commendable performance, particularly in improving classification accuracy after refining noisy labels.

Upon evaluation on the test set, the fine-tuned ViT B-16 model achieved an accuracy of 88%, further highlighting the effectiveness of the `Turtle` framework in handling noisy labels. These findings emphasize the importance of unsupervised label refinement and informed model fine-tuning in improving classification performance. Moving forward, the insights gained from this study can inform future research and application of similar methods to other challenging datasets or tasks where label noise is prevalent.

References

[1] Artyom Gadetsky, Yulun Jiang, and Maria Brbic. *Let Go of Your Labels with Unsupervised Transfer*. 2024. arXiv: 2406.07236 [cs.LG]. URL: https://arxiv.org/abs/2406.07236.