Report on Core ML

Sheil Maniar

March 2025

1 Outline

This is my report on the Core ML assignment

The task was to add symmetric noise [0.2,0.8] to the labels in the CIFAR-10 dataset. We were then to train the model against different loss functions like Cross-Entropy, Focal Loss etc. According to paper, it has been said that normalizing losses enhances the robustness of our model. Normalized Cross-Entropy Loss was implemented on our model.
Next, we had to implement the APL (Active Passive Loss) framework in our model. In the same paper, it is mentioned how the framework further improves performance in noisy datasets.
All results were plotted to demonstrate and compare the performance of all methods under different noise rates.

2 Loss Functions Used

2.1 Cross Entropy Loss

Cross Entropy has various subparts like Binary Cross Entropy, Multiclass Cross Entropy etc. Here, we will be looking into Multiclass as that is relevant to the CIFAR-10 dataset.

Multiclass Cross-Entropy Loss, also known as categorical cross-entropy or softmax loss, is a widely used loss function for training models in multiclass classification problems. For a dataset with N instances,Multiclass Cross-Entropy Loss is calculated as

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}(y_{i,j}\log(p_{i,j}))$$

where

- C is the number of classes.
- $y_{i,j}$ are the true labels for class j for instance i
- $p_{i,j}$ is the predicted probability for class j for instance i

Advantages of CE

- Faster convergence compared to other loss functions like MSE
- Works well with softmax activation for probability distribution outputs

Disadvantages of CE

- Complexity increases with the number of classes
- Requires careful handling of class imbalance problems
- Can be computationally expensive for very large numbers of classes

2.2 Normalized Cross Entropy (NCE)

Normalized Cross Entropy (NCE) is a modified version of Cross Entropy that measures the relative improvement of a model over a baseline solution. It is calculated as:

$$\text{NCE} = \frac{-\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1-y_i)\log(1-p_i))}{-(p\log(p) + (1-p)\log(1-p))}$$

where:

- N is the number of samples
- y_i is the true label for sample i (typically 0 or 1 for binary classification)
- p_i is the predicted probability for the positive class of sample i
- p = $\sum_i y_i/N$ is the average probability (proportion of positive labels) in the dataset

Advantages over Standard Cross Entropy

- Provides a relative performance measure by comparing to a baseline solution
- Accounts for class imbalance by normalizing with respect to the baseline entropy
- More interpretable for imbalanced datasets where standard cross entropy can be misleading

Disadvantages compared to Standard Cross Entropy

- Less commonly used in standard machine learning libraries
- The denominator approaches zero when p is close to 0 or 1, potentially causing numerical instability
- May not be suitable for multiclass classification without modification

2.3 APL Framework

The Active-Passive Loss (APL) framework balances robustness and performance. Active losses (e.g., Cross-Entropy, Focal Loss, etc.) maximize the probability of the true class, while passive losses (e.g., Mean Absolute Error, Reverse Cross-Entropy) also minimize the probabilities of incorrect classes. Pairing of active and passive losses to address underfitting and enhance noise tolerance, making APL effective in noisy-label scenarios.At a high level, a loss is defined as "Active" if it only optimizes at $q(k = y|x) = 1$, otherwise, a loss is defined as "Passive". We denote the basic function of loss $\mathcal{L}(f(x), y)$ by $\ell(f(x), k)$, that is $\mathcal{L}(f(x), y) = \sum_{k=1}^{K} \ell(f(x), k)$. Then, we can define the active and passive loss functions as:

Active loss function $\mathcal{L}_{\text{Active}}$ is an active loss function if $\forall(x, y) \in \mathcal{D}\forall k \neq y \quad \ell(f(x), k) = 0$.
Passive loss function $\mathcal{L}_{\text{Passive}}$ is a passive loss function if $\forall(x, y) \in \mathcal{D}\exists k \neq y \quad \ell(f(x), k) \neq 0$.
Formally,
$\mathcal{L}_{\text{APL}} = \alpha \cdot \mathcal{L}_{\text{Active}} + \beta \cdot \mathcal{L}_{\text{Passive}}$,
where,α,β are parameters to balance the two terms.

For my active loss, I chose to implement NCE (Normalized Cross Entropy), as it would demonstrate how effective it is against plain NCE.
For my passive loss, I have implemented (Mean Absolute Error). As a passive loss, MAE helps address the underfitting problem that robust active losses often face. Specifically, it minimizes the probabilities at incorrect class positions.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

3 Features of Code

1) Improved Model Architecture:
Added batch normalization and dropout to improve generalization with noisy labels.
Batch normalization works by normalizing the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. It solves the problem of internal covariate shift.

2) Data Augmentation :
Data augmentation is a technique that artificially expands a training dataset by creating modified versions of existing data, aiming to improve model generalization and reduce overfitting.

3) MultiStepLR Scheduler :
While running the code, I observed that accuracy plateaus out after a few epochs. Implementing a learning rate scheduler decreases the learning rate after a certain number of epochs, which increases the accuracy in the coming epochs.

4) Weight Decay
Added L2 regularization (weight_decay=1e-4) to prevent overfitting to noisy labels.

5) Saving of Output
All results are saved in a .txt file for further analysis (if needed).

4 Parameters Used

- Number of epochs = 12
- Batch Size = 10
- Learning rate CE = 0.005
- Learning rate NCE and APL = 0.001
- Noise rates = [0.3, 0.5, 0.7]
- Optimizer - Adam
- Weight Decay = 0.0001
- Milestones for Multistep LR = 4,8
- Gamma for MultiStepLR = 0.25
- APL Parameters

Noise Rate	Alpha	Beta
0.3	1.4	0.6
0.5	1.0	1.0
0.7	0.6	1.4

Table 1: APL parameters for different noise rates

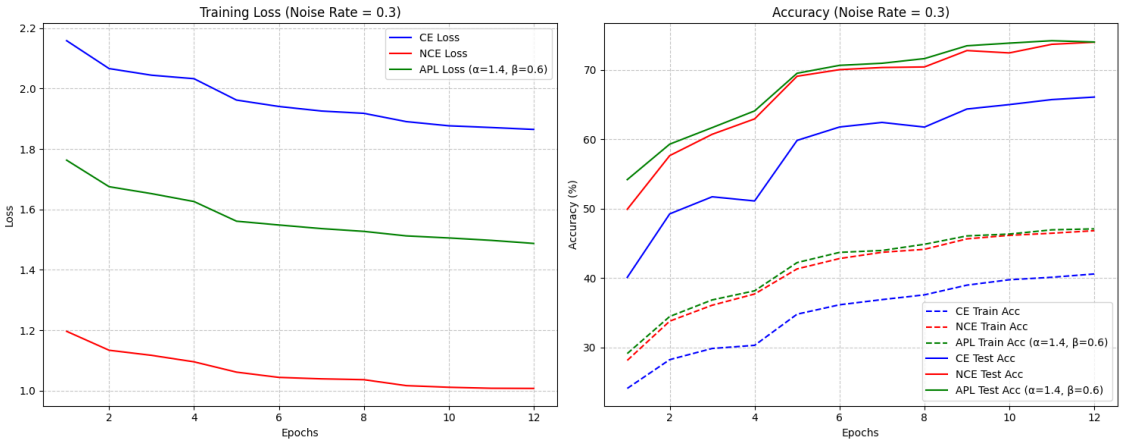
Active loss function is more powerful under low noise rates while passive loss function is superior when noise rate is relatively high. Hence, different weights have been given for different noise levels.

Layer	Configuration
Conv2D	$3 \to 32$ channels, 3×3 kernel, padding=1
BatchNorm2D	32 channels
Conv2D	$32 \to 64$ channels, 3×3 kernel, padding=1
BatchNorm2D	64 channels
MaxPool2D	2×2 pool size
Dropout	dropout rate $= 0.3$
Fully Connected	$64 \times 8 \times 8 \to 512$
BatchNorm1D	512 channels
Fully Connected	$512 \to 10$

Table 2: My CNN Architecture for CIFAR-10 Classification

5 Results

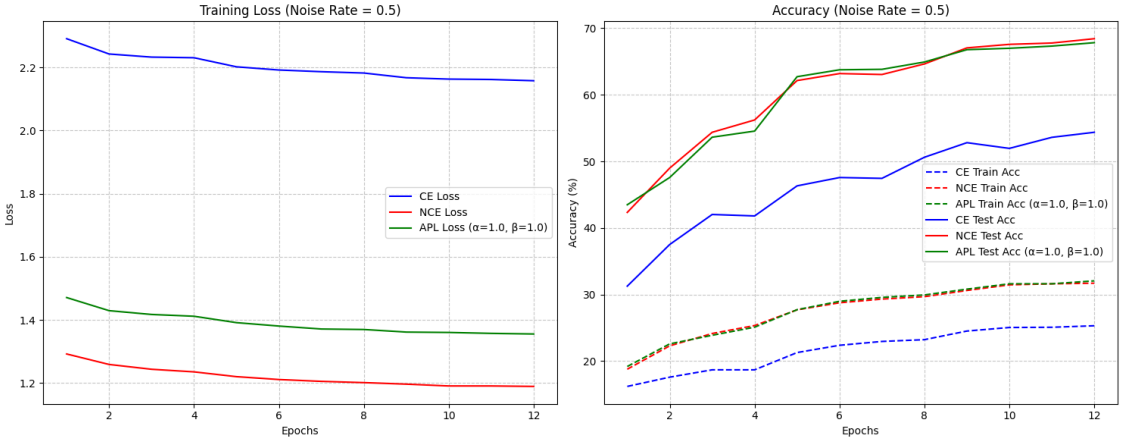
5.1 Noise Rate = 0.3



Loss and Accuracy Plots

As we can see, Normalized Cross Entropy is performing way better than it's vanilla counterpart. APL is also performing and is right on par with NCE in terms of accuracy.

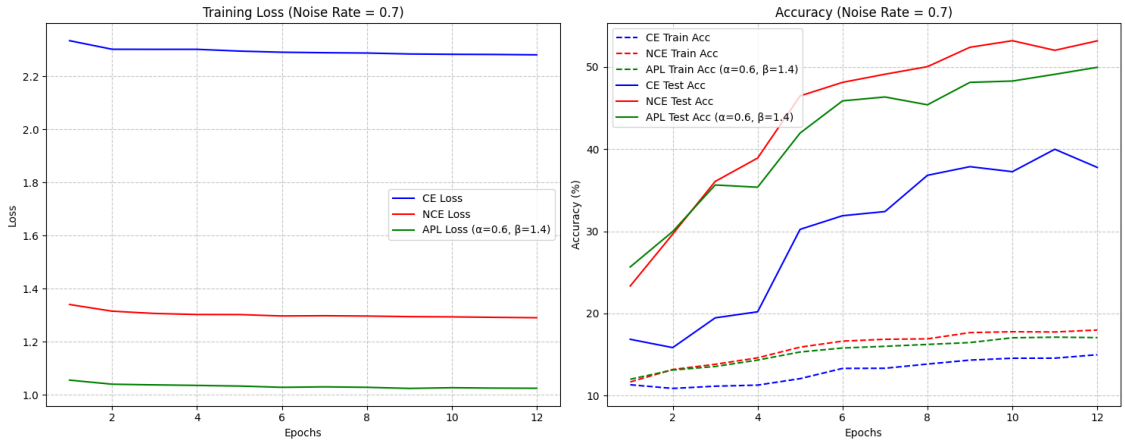
5.2 Noise Rate = 0.5



Loss and Accuracy Plots

Both APL and NCE are superior compared to vanilla CE. We can also notice how APL is closing in on NCE at a decent enough noise level in the loss plot.

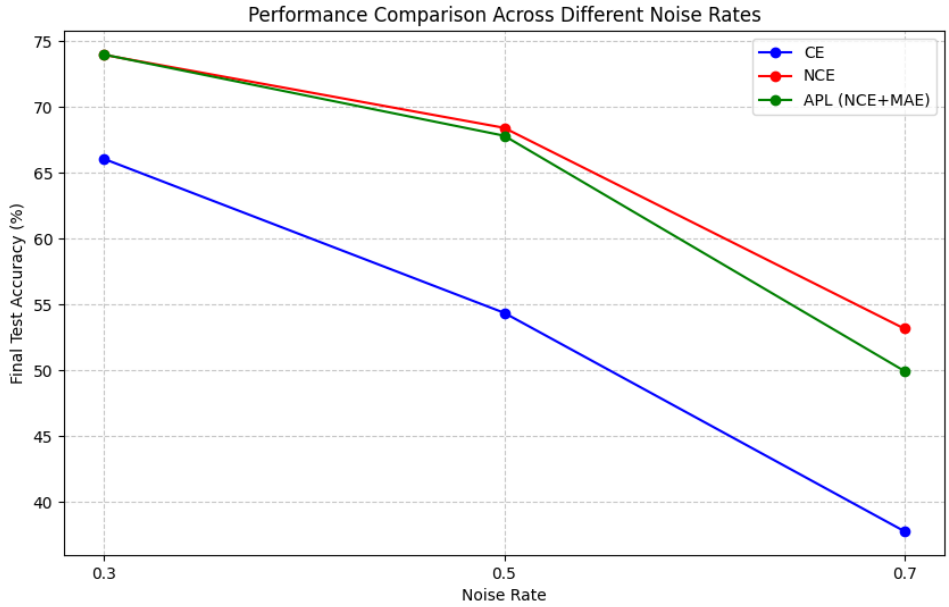
5.3 Noise Rate = 0.7



Loss and Accuracy Plots

As noise level gets higher, APL has overtaken NCE in keeping the noise to the minimum. It shows how passive loss framework is great at identifying incorrect class labels.

5.4 Noise Rate Comparison



Accuracy vs Noise rate

As expected, accuracy of the loss functions decreases as noise increases. Model has difficulty in identifying the correct labels. The rate at which accuracy degrades is different for each loss function, with CE performing the worst.

One can notice how there is a bump in accuracy in every plot after epoch 4 and 8. This means that our MultiStepLR is working as intended.

5.5 Summing all Results Up

Noise Rate	CE Test Acc (%)	NCE Test Acc (%)	APL Test Acc (%)
0.3	66.07	73.98	74.00
0.5	54.37	68.43	67.83
0.7	37.77	53.17	49.95

Table 3: Comparison of test accuracies at the last epoch for different noise rates

Noise Rate	CE Train Loss	NCE Train Loss	APL Train Loss
0.3	1.864	1.007	1.487
0.5	2.157	1.189	1.355
0.7	2.280	1.290	1.025

Table 4: Comparison of training losses at epoch 12 for different noise rates

To conclude, Normalization of loss function helps in improving accuracy and reducing loss compared to just standard loss functions.
APL shows it's robustness at high noise rates. It's loss at 0.7 noise is less than half of what CE's loss is.

6 Points to be Noted

There are certain points to be noted.
1. The balance between active and passive components might need further tuning. The research indicates that the underfitting problem of normalized losses is complex and scaling alone may not solve it.
2. The original APL paper notes that "scaled NCE and NFL only slightly improve learning" in some scenarios, suggesting that parameter tuning has limitations.
3. As noise rate increases (more and more labels are incorrectly labeled), accuracy in general, decreases for all of our loss functions. This follows true with our research paper.

Through this, it has been established that normalizing losses enhances robustness to noisy labels. However, while robust, these losses can suffer from underfitting, limiting performance on clean data. The Active-Passive Loss (APL) framework balances robustness and performance. Pairing the active and passive losses to address underfitting and enhance noise tolerance, making APL effective in noisy-label scenarios. It should be pointed out that these results are not absolute and depend upon the fine tuning of the parameters.