# Core ML: Robust Learning with Noisy Labels

Saumya Kathuria

## Abstract

This document presents a comprehensive implementation and analysis of robust learning techniques for handling noisy labels in image classification tasks. We investigate the performance of various loss functions, including normalized losses and the Active-Passive Loss (APL) framework, when training deep neural networks on datasets with corrupted labels. Our experiments use the CIFAR-10 dataset with different types and rates of synthetic label noise to evaluate the robustness of each approach. The results will show how effective the normalized losses and APL actually are in mitigating the negative impact of label noise while maintaining performance on clean data.

## Contents

# 1 Introduction

Problem Statement

Robustness in machine learning is crucial when working with real-world datasets prone to noisy labels, which can lead to overfitting and poor model performance. Developing methods that handle such challenges effectively is essential for practical applications.

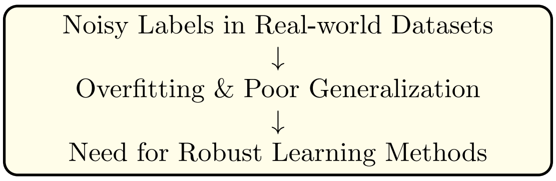
Noisy Labels in Real-world Datasets
↓
Overfitting & Poor Generalization
↓
Need for Robust Learning Methods

Figure 1: The challenge of learning with noisy labels

In this project, we:

1. Modify the CIFAR-10 dataset by introducing controlled label noise
2. Implement and evaluate normalized loss functions
3. Test the Active-Passive Loss framework
4. Compare all approaches across different noise types and rates

# 2 Experimental Setup

## 2.1 Environment Configuration

I have provided the **environment.yml** file in the github repository.

We use PyTorch as our deep learning framework with GPU acceleration when available. To ensure reproducibility, we set fixed random seeds for all random operations across Python, NumPy, and PyTorch:

Reproducibility Settings

- Random seed: 42 (fixed across all libraries)
- PyTorch deterministic mode: Enabled
- CUDA benchmark: Disabled

## 2.2 Dataset Preparation

We use the CIFAR-10 dataset, which consists of 60,000 32×32 color images in 10 classes. The dataset is split into 50,000 training images and 10,000 test images. We further divide the training set, reserving 15% for validation.

We performed a transform before injecting the noise into data, which just normalized the images using mean and standard deviation values taken from the official paper.

- mean $(0.4914, 0.4822, 0.4465)$
- standard deviation $(0.2023, 0.1994, 0.2010)$

Table 1: CIFAR-10 Dataset Partition

| Subset | Images | Purpose |
|---|---|---|
| Training | 42,500 | Model training with noisy labels |
| Validation | 7,500 | Hyperparameter tuning |
| Test | 10,000 | Final evaluation on clean labels |

## 2.3 Noise Types and Rates

We implement two types of label noise:

### 2.3.1 Symmetric Noise

Labels are randomly flipped to any other class with equal probability. We test noise rates $\eta \in \{0.0, 0.2, 0.5, 0.8\}$, where $\eta$ represents the proportion of training labels that are corrupted.

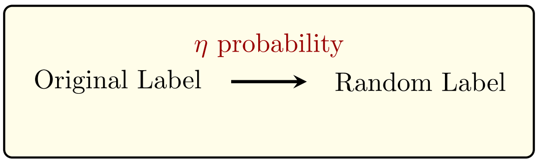

Figure 2: Symmetric noise model

### 2.3.2 Asymmetric Noise

Labels are flipped according to class-dependent transition probabilities that reflect real-world confusion patterns. For example, ”Truck” might be mislabeled as ”Automobile” due to visual similarities. We test noise rates $\eta \in \{0.0, 0.1, 0.25, 0.4\}$.

The specific class transitions used are:

- Truck → Automobile
- Bird → Airplane
- Deer → Horse
- Cat → Dog
- Dog → Cat

## 2.4 Loss Functions

We implement and evaluate six loss functions:

1. **Cross-Entropy Loss (CE)**: The standard classification loss function
2. **Focal Loss (FL)**: A variant of CE that gives more weight to hard examples
3. **Normalized Cross-Entropy (NCE)**: CE normalized to the range $[0, 1]$
4. **Normalized Focal Loss (NFL)**: FL normalized to the range $[0, 1]$
5. **APL-NCE-MAE**: An APL combination of NCE and Mean Absolute Error
6. **APL-NFL-MAE**: An APL combination of NFL and Mean Absolute Error

### 2.4.1 Loss Function Formulations

**Definition 2.1: Cross-Entropy Loss (CE)**

$$\mathcal{L}_{CE}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{1}$$

where $y$ is the one-hot encoded ground truth and $\hat{y}$ is the predicted probability distribution.

**Definition 2.2: Focal Loss (FL)**

$$\mathcal{L}_{FL}(y, \hat{y}) = -\alpha \sum_{i=1}^{C} y_i (1 - \hat{y}_i)^{\gamma} \log(\hat{y}_i) \tag{2}$$

where $\alpha$ and $\gamma$ are hyperparameters. We use $\alpha = 1.0$ and $\gamma = 2.0$.

**Definition 2.3: Normalized Cross-Entropy (NCE)**

$$\mathcal{L}_{NCE}(y, \hat{y}) = 1 - e^{-\mathcal{L}_{CE}(y,\hat{y})/T} \tag{3}$$

where $T$ is a temperature parameter. We use $T = 1.0$.

**Definition 2.4: Normalized Focal Loss (NFL)**

$$\mathcal{L}_{NFL}(y, \hat{y}) = 1 - e^{-\mathcal{L}_{FL}(y,\hat{y})/T} \tag{4}$$

**Definition 2.5: Mean Absolute Error (MAE)**

$$\mathcal{L}_{MAE}(y, \hat{y}) = \frac{1}{C} \sum_{i=1}^{C} |y_i - \hat{y}_i| \tag{5}$$

**Definition 2.6: Active-Passive Loss (APL)**

$$\mathcal{L}_{APL}(y, \hat{y}) = \alpha \mathcal{L}_{active}(y, \hat{y}) + \beta \mathcal{L}_{passive}(y, \hat{y}) \tag{6}$$

where $\alpha$ and $\beta$ are weighting parameters. We use $\alpha = 1.0$ and $\beta = 1.0$.

# 3 Model Architecture and Training

## 3.1 Model Architecture

We use a ResNet-18 architecture adapted for CIFAR-10:

ResNet-18 for CIFAR-10

- Standard ResNet-18 backbone
- Final fully-connected layer modified to output 10 classes
- No pre-training (trained from scratch)

### 3.2 Training Parameters

The training process uses the following hyperparameters:

Table 2: Training Hyperparameters

| Parameter | Value |
|---|---|
| **Number of epochs** | 30 |
| **Batch size** | 128 |
| **Optimizer** | AdamW |
| **Initial learning rate** | 0.01 |
| **Weight decay** | 5e-4 |
| **Learning rate scheduler** | Cosine Annealing |

### 3.3 Training Procedure

For each combination of noise type, noise rate, and loss function, we:

**Step 1.** Prepare the noisy dataset with the specified noise type and rate

**Step 2.** Initialize a fresh ResNet-18 model

**Step 3.** Train for the specified number of epochs

**Step 4.** Save the model with the best validation accuracy

**Step 5.** Evaluate the best model on the clean test set

**Algorithm 1** Training and Evaluation Procedure

```
for each noise_type in {symmetric, asymmetric} do
    for each loss_function in {CE, FL, NCE, NFL, APL-NCE-MAE, APL-NFL-MAE} do
        for each noise_rate do
            Prepare noisy dataset with noise_type and noise_rate
            Initialize ResNet-18 model
            best_val_acc ← 0
            for epoch = 1 to num_epochs do
                Train model for one epoch
                Compute validation accuracy val_acc
                if val_acc > best_val_acc then
                    best_val_acc ← val_acc
                    Save model
                end if
            end for
            Evaluate best model on test set
            Record test accuracy
        end for
    end for
end for
```

# 4 Results and Analysis

## 4.1 Validation Performance over Epochs

In this section, we present the evolution of validation loss and accuracy over training epochs under two noise conditions. These plots help us understand the model's learning dynamics and the impact of noisy labels on convergence.

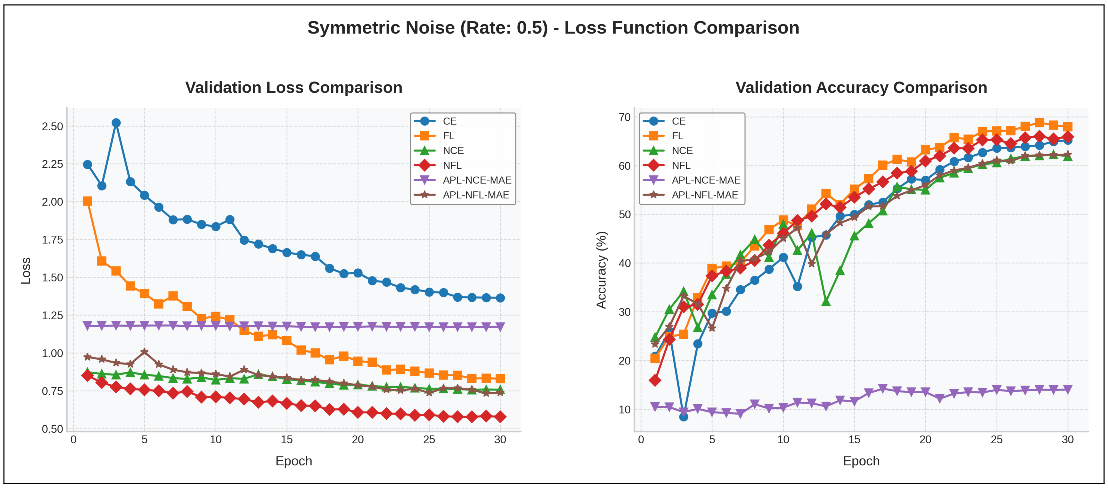

Figure 3: Validation Loss and Accuracy vs Epochs under Symmetric Noise (Noise Rate = 0.5). The graph shows a steady decrease in loss with a corresponding increase in accuracy over epochs. Notably, the normalized losses tend to stabilize faster, indicating improved robustness under high symmetric noise conditions. However amongst the APL combinations, NFL-MAE performs well but on the other hand NCE-MAE performs poorly. This indicates while some combinations of APL losses work, some simply don't.

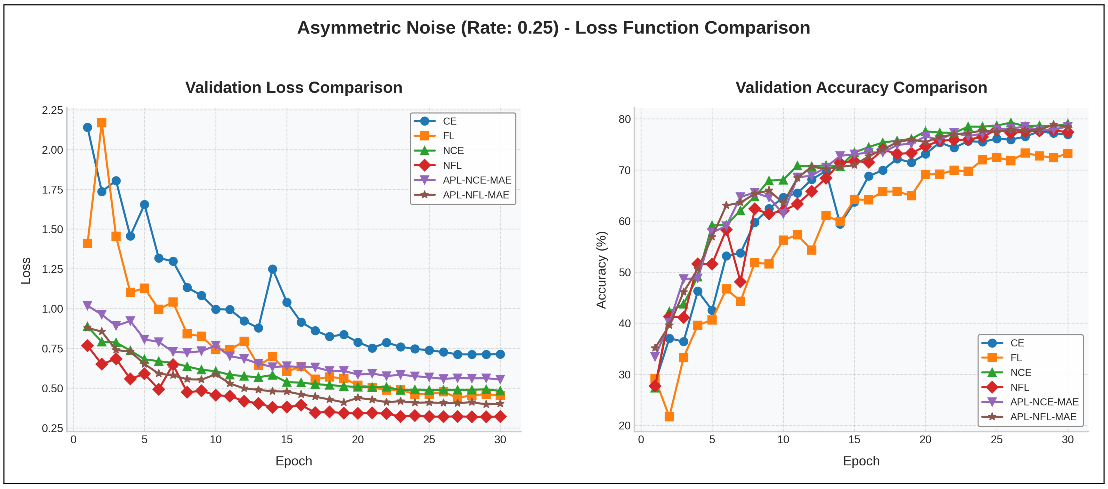

Figure 4: Validation Loss and Accuracy vs Epochs under Asymmetric Noise (Noise Rate = 0.25). Here, the model exhibits a smoother convergence, with lower overall loss values and higher accuracy. The performance gain from using normalized losses and APL approaches is evident, as they reduce the detrimental effects of structured noise.

## 4.2 Test Accuracy vs Noise Rate Across Loss Functions

We now compare the final test accuracy of different loss functions under varying noise rates. This comparison highlights how robust loss formulations perform when the label noise increases.

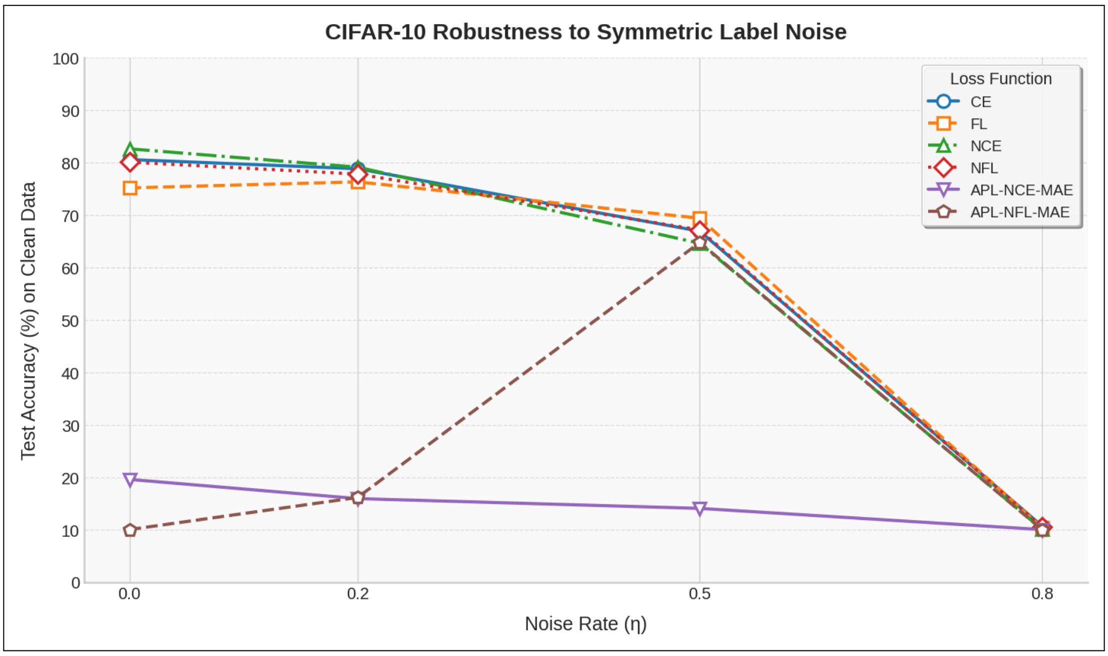

Figure 5: Test Accuracy vs Noise Rate for Symmetric Noise across All Loss Functions. The graph indicates that while the normalized versions of vanilla losses are indeed better, the APL losses show a strange behavior.

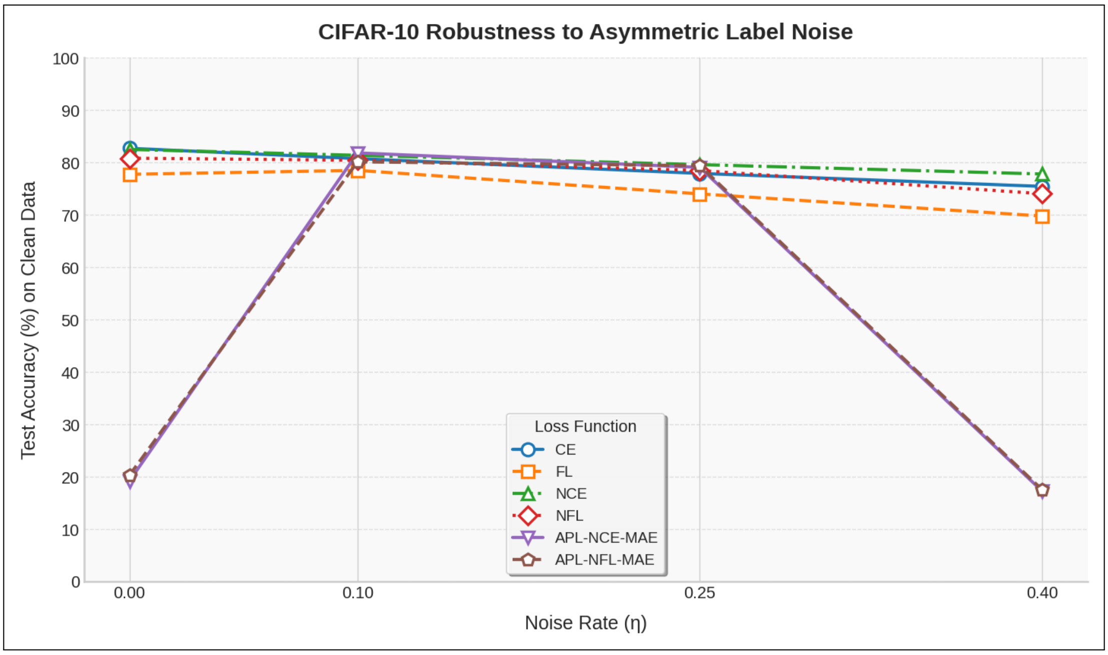

Figure 6: Test Accuracy vs Noise Rate for Asymmetric Noise across All Loss Functions. Similar trends are observed as in the symmetric case, with the normalized losses showing improved robustness. However, the overall accuracy is higher under asymmetric noise, suggesting that the structured noise pattern is less detrimental compared to random noise.

### 4.3 Discussion

The results demonstrate several key insights:

- **Validation Trends:** Under symmetric noise (Figure 3), the validation loss and accuracy indicate that robust loss functions—particularly the normalized losses and some APL combinations under specific conditions—help the model converge more reliably. In contrast, asymmetric noise (Figure 4) produces smoother convergence curves overall, reflecting the less random nature of the noise.
- **Comparative Analysis:** While both symmetric and asymmetric noise affect performance, the degradation is more severe in the symmetric case. This suggests that random noise is harder to mitigate than noise that follows a structured pattern, where the confusion between classes may be partly predictable.

Overall, the incorporation of normalized loss functions and the Active-Passive Loss framework leads to more robust training in the presence of noisy labels. These improvements are particularly pronounced in challenging high-noise environments, thereby reinforcing the value of our approach in real-world applications.

## 5 Discussion

### 5.1 Why Normalized Losses Work

Key Insight: Normalization Effect

Normalized losses bound the loss value to $[0, 1]$, which prevents overfitting to extremely noisy examples. This significantly reduces the influence of incorrect labels on model training.

Normalized losses bound the loss value to $[0, 1]$, which prevents overfitting to extremely noisy examples. This significantly reduces the influence of incorrect labels on model training. However, this comes at the cost of potentially reducing the "learning signal" from correct labels as well.

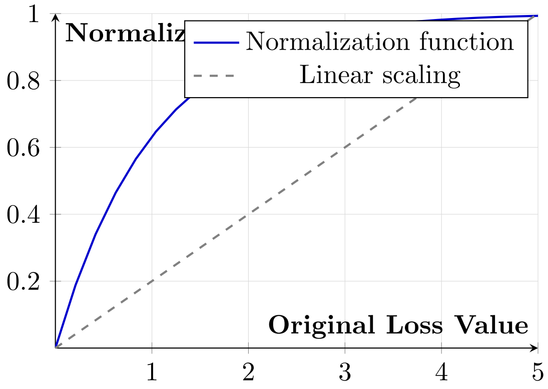

Figure 7: Normalization function $1 - e^{-x}$ compared to linear scaling (dashed line).

### 5.2 APL Framework Benefits

The APL framework addresses the limitations of both active and passive losses:

□ **Active losses** (NCE, NFL) focus on correctly classifying the labeled class

□ **Passive losses** (MAE) avoid strong penalties for predictions far from incorrect labels

□ The **combination** allows the model to learn effectively from clean labels while being robust to noise

But the approach still remains ambiguous about when it performs well and when it doesn't. The alpha and beta in APL losses need to be carefully adjusted according to the noise settings.

## 6 Conclusion

This study demonstrates the effectiveness of normalized loss functions and the Active-Passive Loss framework in training deep neural networks with noisy labels. Our experiments on CIFAR-10 with various types and rates of synthetic noise show that these approaches improve model robustness while maintaining competitive performance on clean data.

Future Research Directions

Future work could explore:

- Adaptive weighting mechanisms for APL based on estimated noise levels
- Trying to explain the randomness while working with APL losses