Assignment 2

Group members:
ShuoZhang, 530791539, szha0882
Zichong Zeng, 540075768, zzen0696
Niklas Schmitt, 540940765, nsch0022

Abstract

This paper presents two algorithms specifically designed to handle label noise, a common issue in real-world datasets that can negatively impact model performance. Label noise refers to inaccuracies in the labeled data of supervised learning, which can lead to biased model predictions, reduced generalization, and lower accuracy. Models trained on noisy data may learn incorrect patterns and struggle to generalize to clean data effectively. To address these challenges, we designed two robust deep learning algorithms. The first is a label noise robust multilayer perceptron (MLP), and the second is a label-noise robust convolutional neural network (CNN). To enhance model robustness, we incorporated a transition matrix in the MLP and designed a robust loss function in the CNN.

The organization of this paper is as follows: First, we introduce related work to familiarize readers with label-noise learning. Next, we provide a detailed explanation of the methods we used. Finally, we analyze experimental results to draw a compelling conclusion.

1 Introduction

Label noise will significantly impacts the performance of supervised learning models, leading to issues like reduced accuracy, increased bias, unstable training, and higher computational costs. Label noise can mislead models into learning incorrect patterns and obscure the true data distribution, thus decreasing generalization ability.[21] Furthermore, noisy labels can bias models towards incorrect patterns, especially in imbalanced datasets, which can intensify this bias. Deep learning models are particularly vulnerable to label noise, which can result in unstable training and overfitting, reducing adaptability to new data. Traditional methods to address label noise, such as data cleaning and detection processes, are computationally expensive. Consequently, effectively handling label noise is crucial in real-world applications like medical diagnostics, sentiment analysis, autonomous driving, and financial fraud detection. Efficiently managing label noise not only enhances the reliability and practical value of models but also increases their business value.[14]

As mentioned in the abstract, we designed two algorithms to address label noise. In the MLP, we incorporated a label noise transition matrix that characterizes the distribution of label noise in the dataset at the softmax output layer.[14] Specifically, the transition matrix adjusts the softmax output, allowing the MLP to recognize the presence of label noise and improve robustness.[4] In the robust CNN, we implemented a specially designed loss function. This series of loss functions enhances model robustness in datasets with label noise by employing various mechanisms and strategies (such as asymmetric penalty and reverse learning[23]) to minimize the impact of noisy labels, allowing the model to learn clean features more effectively.[3]In addition, we design a method to estimate the transition matrix and verify its effectiveness.

2 Previous work

Previous research has proposed several approaches to address the challenges posed by label noise in supervised learning.The transition matrix-based approach is an intuitive option. G. Patrini suggested incorporating a label noise transition matrix into the loss function to mitigate the adverse effects of noisy labels on model performance. This approach adjusts the loss calculation to account for noise, enhancing the model's robustness and reliability in learning from imperfect data.[14] Goldberger introduced a noise adaptation layer that uses a transition matrix to capture and account for label noise. This layer enables the model to adjust dynamically to noisy labels throughout the training process, improving its robustness and accuracy.[4]

There are some the advantages of this method: 1.Effectiveness The transition matrix models the noise distribution explicitly, helping the model better understand and handle label noise. 2.Flexibility:It is compatible with deep learning models and requires minimal modification, making it adaptable to various noise distributions. 3.Strong interpretability: The main diagonal of the noise transition matrix represents the probability of each class label remaining unchanged, indicating the likelihood of correctly labeled samples. The off-diagonal elements describe the probability of labels being misclassified as other classes.

And disadvantages: 1.Difficulty in accurate Estimation: In practical applications, accurately estimating the label noise transition matrix can be challenging, especially in cases with multiple classes and complex noise patterns. 2.High Computational Complexity: As the number of label classes increases, the number of parameters in the transition matrix also grows significantly, leading to higher computational complexity and memory overhead. 3.Limited Adaptability to Changing Noise Environments: In real-world scenarios, the type and distribution of noise may shift over time.However, the transition matrix is usually static, making it difficult to adapt to dynamic noise conditions.

Another approach is to design robust loss functions that reduce the impact of label noise on model training. G.Patrini studied the performance of various loss functions under label noise and identified some that are relatively robust to noise.[14] Zhang and Sabuncu introduced the Generalized Cross Entropy (GCE) loss, which adjusts for different levels of noise through asymmetric modulation of prediction confidence.[23] Ma further proposed Normalized Loss Functions, which make the loss less sensitive to outliers, thus improving model stability in noisy label environments.[12]

Advantages of this method: 1.No Need for Explicit Noise Distribution Estimation: Unlike transition matrix-based methods, robust loss functions do not require explicit modeling or estimation of the label noise distribution. This makes them effective even when the noise distribution is unknown or hard to estimate. 2.Adaptable to Different Types of Noise: Robust loss functions are generally adaptable to various types of noise, offering flexibility across different noise patterns. 3.Easy to Implement: Most robust loss functions can be directly implemented within existing frameworks without requiring complex adjustments to the model structure or the addition of new modules.

Disadvantages: 1.Sensitivity to Parameter Selection: Some robust loss functions require parameters to control the impact of noise. The choice of these parameters can significantly affect model performance, often necessitating additional hyperparameter tuning, which increases implementation complexity and training time.

2.Potential Impact on Convergence Speed: Certain robust loss functions may slow down model convergence, as they reduce gradient updates for specific samples during optimization. As a result, more training iterations (epochs) may be needed to achieve the target accuracy, further extending training time.

3 Methods

3.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs), a subset of Feedforward Neural Networks (FNNs), are particularly notable for their ability to process input in matrix form, such as images. Unlike traditional neural networks that require flattening input data, CNNs can retain the spatial hierarchy of image pixels, allowing objects within an image to be recognized regardless of their positional variance. This capability is rooted in the observation that pixels are typically more closely correlated with their immediate neighbors than with distant ones. To exploit this characteristic, CNN neurons focus on a pixel and its adjacent pixels to gather localized information. Filters, applied through a sliding-window technique, scan the image by capturing regions based on a predefined kernel size, such as 2×2 or 3×3. These filters maintain consistent weights across different positions, which reduces the parameter count substantially in comparison to an FNN [5].

In deeper CNNs, multiple convolutional layers are stacked, each generating outputs corresponding to the number of filters used. For color images (RGB), these filters are independently applied to each color channel, though they may remain two-dimensional. To further enhance the network's efficiency and control for overfitting, CNNs commonly incorporate pooling layers. These pooling layers aggregate the convolutional output by preserving only the strongest activations (e.g., maximum values), effectively reducing dimensionality while retaining essential spatial features [11].

3.2 Proposed Acticture ture CNN

According to several papers, a small CNN without a pretrained backbone as ResNet can be utilized to classify images reliably on Fashion MNIST and Difar-10. Research has proposed models consisting of a three-layer CNN followed by four fully connected layers, which achieve about 91% accuracy without preprocessing [9]. Yet, another good model is a two-layer CNN batch normalization and skip connections; it achieves 92.54% accuracy. Moreover, a fine-tuned CNN can reach up to 93.69% accuracy at time complexity of 3.8 million operations [7]. These results illustrate CNNs' potential in handling Fashion MNIST dataset's complexities. To ensure computational efficiency, we kept our CNN fixed to two convolutional and two fully connected layers. For the optimization of performance, hyperparameter tuning was conducted over the learning rate, dropout, kernel sizes, channel counts, batch size, batch normalization, epochs, L1 regularization, and the loss functions being varied.

For the loss functions, we considered cross entropy, a commonly used classification loss that calculates the difference between predicted probabilities and true labels. However, cross entropy is known to be unstable in the presence of noisy labels, as noted by Ma et al. [13]. Theoretically, they show that any loss function can be made more robust to label noise by just applying some simple normalization, providing an insight toward designing new, resilient loss functions. Following these principles, they analyzed the behavior of a variety of loss functions on the noisy Fashion MNIST dataset. Of those, ReverseCrossEntropy combined with NormalizedFocalLoss turned out to be the best-performing losses.

3.3 Multi-Layer Perceptron (MLP)

Multi-layer perceptron is a full connection, feed-forward neural network classifer [5]. Figure 1 hows the architectural design of the model. There are two hidden layers go after one input layer and follows by a output layer. Each hidden layer has 64 nodes and the output layer has 4 nodes.

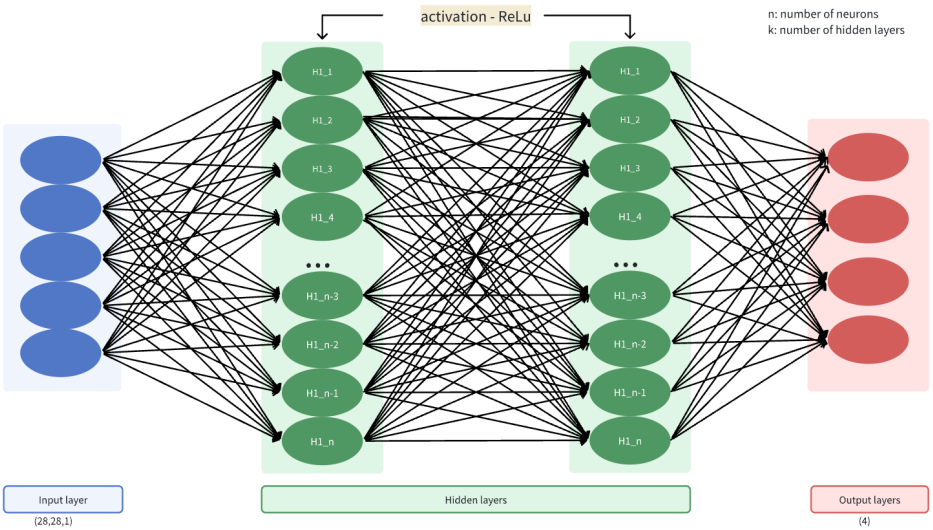

Figure 1: MLP Structure

When the model receives the input data, the input layer will flatten the input to a 2D matrix before being fed to the network. The first hidden layer receives the data from the input layer, then after applying ReLU as the activation function.It return 0 if input is negative, and return input itself if the input is positive.

$$\text{ReLU}(x) = \max(0, x)$$

After that, the data will be fed to the next hidden layer. Before the output layer, we design a droupout function which is a regularization technique applied to hidden layers to prevent overfitting by randomly dropping units in the MLP [17]. We used PyTorch as the deep learning framework to implement the model. When training the model, MLP will learn the classification to reduce the cross-entropy loss function by changing the weight between neural network nodes. Additionally, the Adam algorithm is used to act as an optimizer since Adam is typically used to being fairly robust to hyperparameter settings [5].

3.4 Transaction Matrix Estimation Method

Our method primarily relies on the theory of label noise modeling and high-confidence sample selection. Specifically, it uses the label noise transition matrix theory, combined with the assumption of high-confidence sample representativeness, to estimate the label noise distribution.

3.4.1 Label Noise Modeling

The theoretical assumption behind label noise modeling is that label noise can be described using a label transition matrix T. The transition matrix T defines the conversion relationship between the true label y and the observed noisy label $\tilde{y}$. Each element T_{ij} in the transition matrix represents the probability that a sample with a true label i is incorrectly labeled as j in a noisy environment. Specifically,

$$T_{ij} = P(\tilde{y} = j \mid y = i)$$

By estimating this transition matrix, we can simulate the effect of label noise during training, allowing the model to capture the true label distribution even in noisy conditions and thereby enhancing model robustness.

3.4.2 High-Confidence Anchor Sample Assumption

In estimating the transition matrix, we utilize high-confidence anchor samples to approximate the noise distribution of true labels. According to the High-Confidence Anchor Sample Assumption, high-confidence samples are closer to clean samples, and predictions on these samples are more likely to be accurate. [6].. By using the average prediction probabilities across different classes for these high-confidence samples, we estimate the noise distribution. This estimation assumes that high-confidence samples have lower prediction error and thus provide a more reliable reflection of the actual label noise.

3.4.3 Theoretical Basis for Transition Matrix Estimation

The theoretical foundation for estimating the transition matrix using the average predicted probability of high-confidence samples comes from the Law of Large Numbers in statistics. The Law of Large Numbers states that, as the sample size increases, the sample mean converges to the true mean.[18] Therefore, with a sufficient number of high-confidence samples, the average predicted probability can serve as a reliable estimate of the label noise transition matrix.

3.4.4 Detail process

Step1 We train a CNN classifier on a dataset with noisy labels, where the model classifies each sample. The trained model generates a predicted probability distribution for each sample, which serves as the basis for selecting high-confidence samples in subsequent steps. Let the output of the model f_θ on a sample x be:

$$p = f_\theta(x) = \text{softmax}(z)$$

where z is the raw output (logits) of the model on sample x, and $p \in R^C$ represents the probability distribution of the sample across C classes.

Step2 Define a confidence threshold and select only those samples with prediction confidence above this threshold. These samples are considered "high-confidence" predictions and are more likely to be clean samples.[8]. For each sample (x, y) ,if the maximum predicted probability meets the following condition, it is selected as a high-confidence anchor sample:

$$\max_j p_j \geq \tau$$

where p_j represents the predicted probability for class j and τ is the confidence threshold.In addition to meeting the high-confidence condition, the predicted class for the sample must match the true label:

$$\arg\max_j p_j = y$$

This further ensures that the selected samples are likely to be "clean" samples that the model predicts accurately.

Step3 Using the selected anchor samples, we estimate the label noise transition matrix by calculating the model's average predicted probability distribution over these samples. Firstly we organize Anchor Samples by Class. Specifically, we group the high-confidence anchor samples by their true label. For example, place all high-confidence samples with a true label of class A in the A class group, and all high-confidence samples with a true label of class B in the B class group, and so on.[6] For each class k, the set of anchor samples is denoted byA_k:

$$A_k = \{(x, y) \mid y = k, \max_j p_j \geq \tau, \arg\max_j p_j = y\}$$

Each group of samples represents a relatively "clean" subset for that class, reducing the influence of noise on the samples. Secondly,we calculated the average predicted probability for each Class.For the anchor samples in each class, calculate the average predicted probability distribution. The mean predicted probability for the anchor samples of class k is denoted by q_k

$$q_k = \frac{1}{|A_k|} \sum_{(x,y)\in A_k} f_\theta(x)$$

where $|A_k|$ represents the number of anchor samples in class k, and $q_k \in R^C$ is the average predicted probability distribution for class k. This average reflects the probability distribution of samples with a true label of k being predicted as other classes in a noisy environment.

Finally, we construct the Label Noise Transition Matrix. Fulfill each row of the transition matrix T with the average predicted probability for each class, thereby constructing the label noise transition matrix T.The element in the ith row and jth column of the transition matrix T, represents the probability that a sample with a true label i is mislabeled as class j

$$T = \begin{bmatrix} T_{00} & T_{01} & \cdots & T_{0(C-1)} \\ T_{10} & T_{11} & \cdots & T_{1(C-1)} \\ \vdots & \vdots & \ddots & \vdots \\ T_{(C-1)0} & T_{(C-1)1} & \cdots & T_{(C-1)(C-1)} \end{bmatrix}$$

At the end, the constructed transition matrix T describes the distribution of label noise, representing the conversion relationship between true labels and observed labels.

Step4: For optimizing the transition matrix estimator, we primarily focus on optimizing the CNN classifier, similar to our approach with the MLP. The parameters are updated using cross-entropy loss.

4 Experiment

This section highlights the key findings identified in our analysis of the various algorithms and explains them. It includes an introduction to the optimization approach used to find the optimal set of hyperparameters for best model performance. Additionally, the dataset is presented along with insights into the evaluation metrics. Furthermore, this section details the experimental setup employed in the study and analyzes the results of the different variants.

4.1 Dataset

In this experiment, we used three different datasets for a classification task involving noisy data. Both are subsets of the original dataset. The Fashion-MNIST dataset [20] comprises 28×28 grayscale images of 70,000 fashion products across 10 categories, with 7,000 images per category. Relative frequnecy of occurance is the same for the diffrent categroies. To conserve computational resources, we limited this study to 24,000 images for training and 4,000 images for testing, using only 4 classes. The dataset was utilized twice with different levels of label noise: one subset with 70% correct labels and another with 40% correct labels, as defined by transition matrices.

The CIFAR-10 dataset, a subset of labeled small-scale image collections, consists of 60,000 32x32 color images grouped into one of 10 classes [10]. In our experiment, we used a trimmed-down version with 16,000 images for training and another 4,000 images for clean testing and used 4 classes as well. According to our training data estimates, about 80% of the training labels are accurate.

The present study makes use of two datasets: each presenting a relatively balanced class distribution, where all classes occur at approximately equal frequencies.

4.2 Transaction Matrix Estimation Process

4.2.1 Applying Matrix Estimation to CIFAR-10 Dataset

We estimate the label nose transition matrix of CIFAR-10 using anchor point method which introduced in section 3.4. And the outcome is

$$\begin{bmatrix} 0.8654 & 0.0930 & 0.0286 & 0.0130 \\ 0.0236 & 0.8846 & 0.0902 & 0.0016 \\ 0.0325 & 0.0123 & 0.8505 & 0.1047 \\ 0.1319 & 0.0047 & 0.0745 & 0.7889 \end{bmatrix}$$

4.2.2 Validating Estimation Method on FashionMNIST Dataset

To validate the reliability of our estimation method. We validated the effectiveness of our designed transition estimator in FashionMNIST0.3 data set. The true matrix of this data set is

$$\text{true\_matrix} = \begin{bmatrix} 0.7 & 0.3 & 0 & 0 \\ 0 & 0.7 & 0.3 & 0 \\ 0 & 0 & 0.7 & 0.3 \\ 0.3 & 0 & 0 & 0.7 \end{bmatrix}$$

and our estimation is

$$\text{estimated\_matrix} = \begin{bmatrix} 0.7702 & 0.1063 & 0.0316 & 0.0919 \\ 0.0264 & 0.7507 & 0.2105 & 0.0123 \\ 0.0165 & 0.0178 & 0.8154 & 0.1503 \\ 0.1804 & 0.0182 & 0.0507 & 0.7507 \end{bmatrix}$$

We quantified the estimation error by calculating the Mean Squared Error (MSE) and Mean Absolute Error (MAE), with the results as follows: $(\text{MSE} = 0.0075, \text{MAE} = 0.0691)$ The low estimation error demonstrates that our method can accurately estimate the label noise transition matrix, proving the effectiveness of this approach.

4.3 Testing Procedure

We used a Bayesian Optimization framework with Python library Optuna [1] in order to find the best-performing model on the evaluation set.

Inside the optimization loop, we provided the optimizer with the mean accuracy of a 10-fold cross-validation to minimize the chance of obtaining a result by chance. We tried at least 25 diffrent compinations of the Hyperparameters (HP)

We performed this optimization procedure only on the CNN, as previous research has shown that CNNs generally outperform Multi-Layer Perceptrons (MLPs) on image data due to their ability to capture spatial hierarchies in images [**?**].

4.3.1 Baysian Optimization

In Bayesian Optimization (BO), an optimization problem can be represented as minimizing an objective function $f(x)$, where $f(x)$ could represent a specific metric, such as accuracy in classification tasks, with the goal of finding optimal hyperparameters [19]. BO is a form of black-box optimization, as it requires no access to gradients or Hessians of the target function $f(x)$. It focuses on global rather than local optimization.

To locate an optimal solution, BO uses an acquisition function $\alpha(x|D)$, balancing exploration (probing unexplored regions) and exploitation (refining promising regions). The acquisition function depends on a probabilistic model $p(y|x, D)$, which estimates the objective function based on past observations D. Common choices for these models include Gaussian Process Regression (GPR) [15] and the Tree-Structured Parzen Estimator (TPE) [2].

The algorithm iteratively refines $p(y|x, D)$ by selecting new queries x , which is in our case the hyperparameters, that maximize $\alpha(x|D)$, updating D with each query result $y = f(x)$. This process continues until a termination criterion is met, often a time or iteration limit, leading to an optimized solution with minimal risk of getting stuck in local minima.

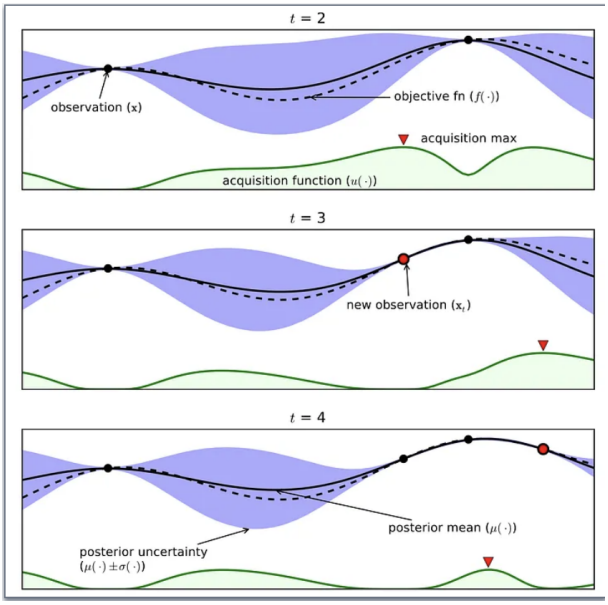

Figure 2: Baysian Optimization Steps

The biggest advantage of Bayesian Optimization is that it provides a guided approach, which often performs better than an uninformed grid search [19]. A drawback is that since the optimization framework frequently changes multiple hyperparameters at once, it can be hard to get a feeling for how each parameter individually affects the evaluation metrics [19].

4.4 Evaluation Metrics

In this experiment, we compared the performance and the robustness of different Artificial Neural Network (ANN) architectures. The top-1 accuracy metric was used to evaluate each classifier's performance, defined as the percentage of examples classified correctly out of all test examples. The top-1 accuracy is calculated as follows:

$$\text{Top-1 Accuracy} = \frac{\text{Number of correctly classified examples}}{\text{Total number of test examples}} \times 100\% \tag{1}$$

4.5 Result evaluation

This section covers the results of the different algorithm variants on the various datasets. We also performed a HP optimization for the CNN using the Optuna framework, which provides built-in visualization functions. These visualizations help in understanding the behavior of different HP and their influence on the performance of the algorithm.

4.5.1 Optimisation History

The following figures are optimization histories of the model for different trials. The y-axis represents the accuracy on the evaluation set, while the x-axis represents the trial number in which every accuracy result was achieved.

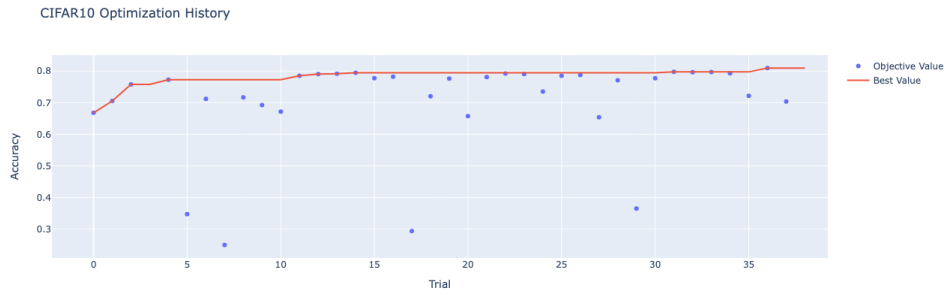

Figure 3: Baysian Optimization Steps

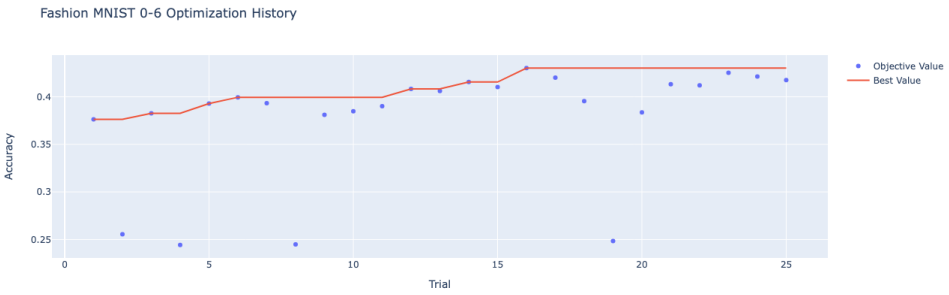

Figure 4: Baysian Optimization Steps

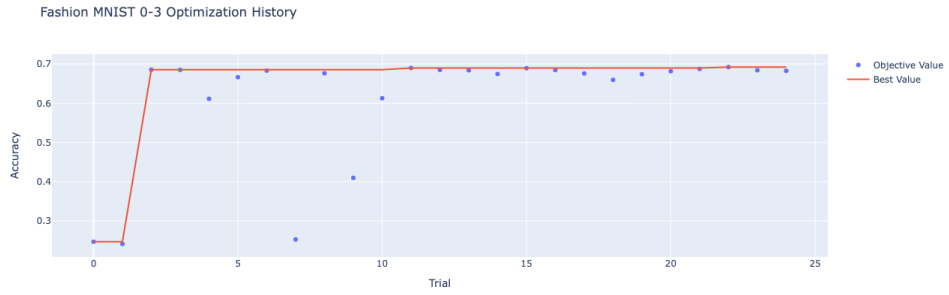

Figure 5: Baysian Optimization Steps

In all experiments, we see that the theoretical limit of maximal accuracy on noisy datasets is reached relatively fast, whichever the dataset. This theoretical limit assumes the provided and estimated transition matrices to be correct. For instance, for the Fashion MNIST 0.6 dataset, this limit is about 40%, since only 40% of the data are correctly labeled. It follows that the algorithm cannot classify data that are wholly mislabeled, and therefore, this

limit cannot be significantly exceeded. This pattern is also followed in other datasets, where the observed accuracy remains around, and does not go very far beyond, the theoretical limit. The slight deviations above this threshold can probably be attributed to chance.

We note that the CNN classifier is remarkably stable across varying hyperparameter settings for each dataset: for both the CIFAR dataset and the Fashion MNIST 0.3 dataset, the standard deviation in accuracy over varying hyperparameter settings is only 0.15; the standard deviation across trials for the noisiest dataset, Fashion MNIST 0.6, further reduces to just 0.05. This might mean that hyperparameter optimization didn't help much with model accuracy, as the highest possible accuracy was always achieved in the first few trials. In these noisy environments, hyper-parameter tuning based on accuracy becomes less relevant, as the model's ability to learn meaningful patterns is bounded from the start by data quality. The model will eventually hit a plateau defined by the theoretical limit of achievable accuracy, beyond which further optimization has a minimal effect.

4.5.2 Hyperparameter Importances

In high-dimensional data analysis, the importance of individual HPs might be hard to determine due to complex interactions and dependencies among parameters [1]. If there are many hyperparameters, the importance of each one might be hidden, just like in Simpson's Paradox, where apparently strong trends in lower-dimensional slices of data might disappear or even reverse when looking at a full dataset .

The optimization framework we used enables us to look into the importance of hyperparameters by measuring the relationship between each parameter and the evaluation accuracy as well as the recored standard deviation. A tree-based estimator is thereby applied to capture non-linear dependencies and interactions among hyperparameters [1]. The tree model ranks every hyperparameter in terms of its contribution to the general error reduction in evaluation accuracy. It thus enables us to find out which parameters are most relevant during optimization.

4.5.3 Hyperparameter Importances on Accuracy Results

As shown in the appendix A.1, the hyperparameter (HP) importance for model accuracy varies across datasets. For the Fashion MNIST 0.6 dataset with more noise, dropout rate and learning rate emerge as the most influential hyperparameters. The implication here is that, under noisier conditions, both regularization by dropout and learning stability through learning rate are crucial for better performance. On the other hand, in the less noisy Fashion MNIST 0.3 dataset, only one hyperparameter has an importance above 20%: whether batch normalization layers follow each convolutional layer. This suggests that when the noise is lower, regularization of the activations by batch normalization becomes a primary factor to help the model generalize. Here again, the learning rate is a critical factor for the CIFAR-10 dataset. Besides, the batch size has a significant impact on performance in this dataset, which indicates that the amount of data processed at each update step affects how well the model can learn from the rich and detailed images in CIFAR-10.

We visualized the most important hyperparameters in a simple 2D plot, with accuracy on the y-axis and the relevant hyperparameter on the x-axis in the plots below. Capturing clear trends is challenging due to the multidimensional nature of the experiment. However, a higher density of points can indicate that the optimizer identified an interesting region or local optimum.

4.5.4 Learning Rate

For the learning rate, which was an essential hyperparameter for both the Fashion MNIST 0.6 and CIFAR-10 datasets, we saw the following effect:

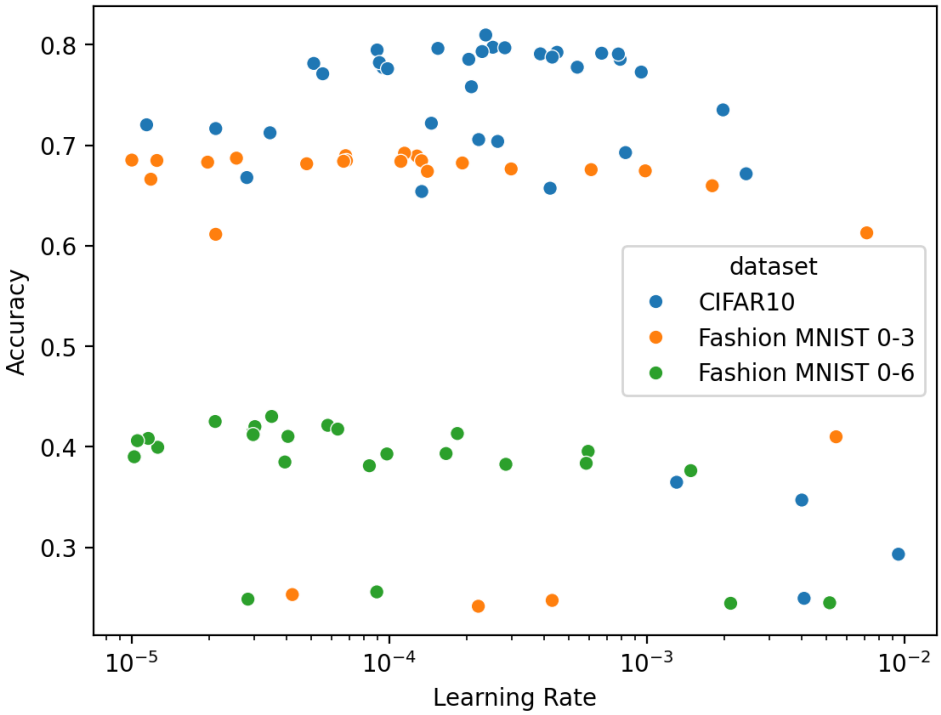

Figure 6: Accuracy vs. Learning Rate

As shown in Figure 6, there is a higher density of points between learning rates of 10^{-4} and 10^{-3}. The accuracy also peaks in this range compared to other ranges.

A similar trend can be seen for the Fashion MNIST dataset, although it is not as pronounced as in CIFAR-10. To be precise, the range of 10^{-5} to 10^{-3} seems to give better accuracy, with a slight decrease in performance as the learning rate increases beyond this interval. This again suggests that the generally smaller learning rates are better for stability and the performance on these noisy datasets.

4.5.5 Dropout Rate

It is also interesting to look at the dropout rate, another important factor in model performance, and compare the results from the CIFAR-10 dataset with 80% correct labels to the Fashion MNIST 0.6 dataset, having only 40% correct labels in its training set. We find that for the more stable CIFAR-10 dataset, a lower dropout rate between 10% and 25% is preferred, while for the noisier Fashion MNIST 0.6 dataset, a higher dropout rate between 40% and 60% is more frequent and gives slightly better results. That is the difference it makes, because in noisier datasets, a higher dropout rate acts as a stronger regularizer to help the model avoid overfitting to incorrect labels by encouraging it to focus on robust patterns rather than spurious correlations. For less noisy datasets like CIFAR-10, a lower dropout rate allows the model to retain more from every training pass, which optimizes learning stability and general accuracy without over-regularization.

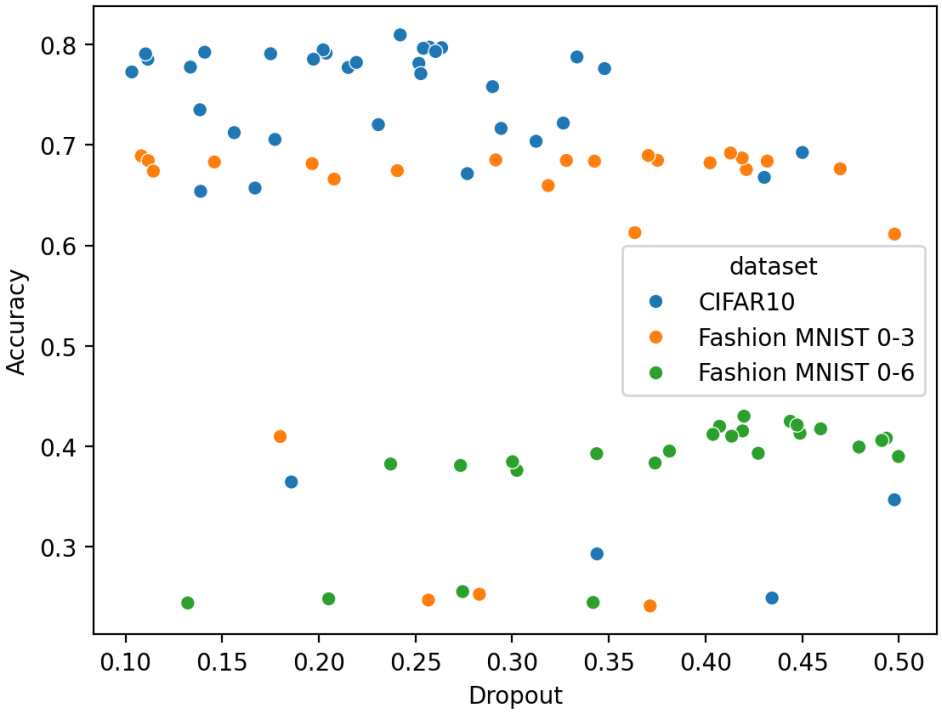

Figure 7: Enter Caption

4.5.6 Hyperparameter Importances on Accuracy Standard Deviation

These hyperparameters affect the standard deviation of test accuracy because they control the ways in which the model learns from data and handles noise. The **dropout rate** enforces variability by randomly knocking out neurons during training, which causes divergence of the learned representations across folds. The **learning rate** affects convergence stability-higher rates may result in fast but unstable learning, leading to increased variability of accuracy across cross-validation folds. The **number of epochs** affect overfitting since the higher number of epochs, the sensitivity towards noise increases hence increased variability in accuracy on the test set.

We can see that the inforamtion that we can get from the follwing plots are not that insightful compaired to the plots above. the reason might be that in the optimization there is not a cost for the standativation of the test results, which leads to the fact that we can not see spaces where there is a lower standard deciation or just a higher density of the points

4.5.7 Epochs

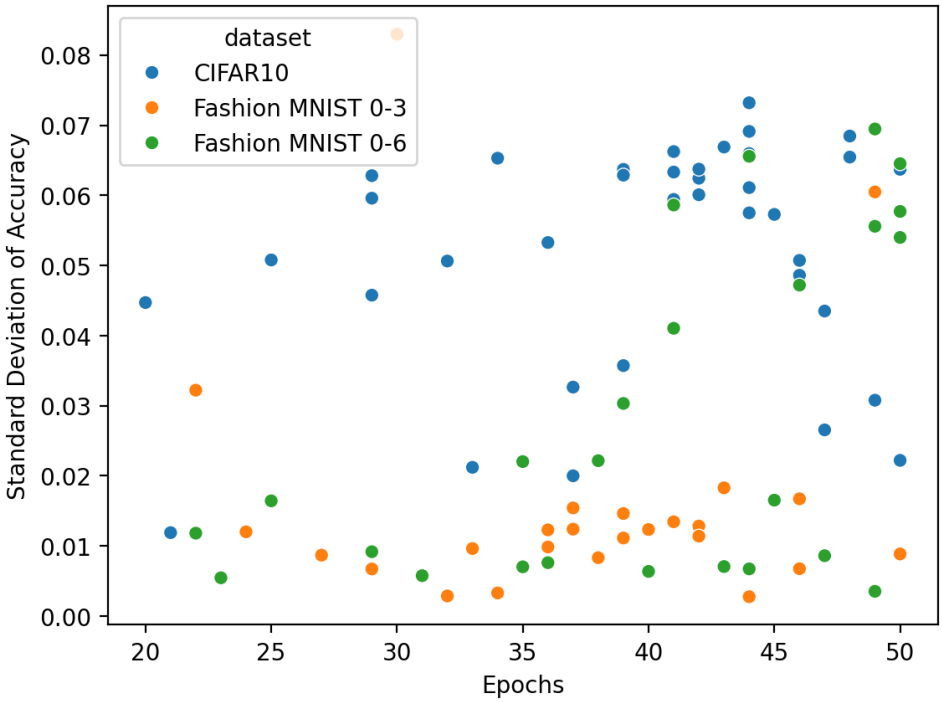

Figure 8: Enter Caption

Our hypothesis that the higher number of epochs will have a higher standard deviation seems true. There is a bit of an upward slope in the std as we increase the number of epochs. This probably comes from the slight overfitting that the model does on the noisy training dataset.

4.5.8 Dropout and Learning Rate

On the noisiest Fashion MNIST dataset, represented by green points, we can observe that higher dropout rates tend towards higher standard deviations in accuracy. However, in other datasets, a clear trend is hard to be found in this visualization. Moreover, for the learning rate, no conclusive trend can be concluded from the data as to whether the smallest learning rates always cause the highest standard deviation or not A.2.

4.6 Results on Unseen test data

4.6.1 DIFAR10 unkown flip rate

The best performance, tuned CNN, tested on the test set of the DIFAR Dataset, gave an accuracy of about 89%. It can be seen from the confusion matrix that this model gets confused among the classes ”birds” and ”cats” probably because both share many visual features, such as open backgrounds, which can be a cause of misclassification. Some of the ”airplane” images were misclassified as ”birds” since both often appear in the sky and may develop structural similarities.

Compared to state-of-the-art models, the best performance, for example, was recorded by DINOv2, reaching an accuracy of 99.5%, while it was trained on a cleaner dataset, further showing the improvements to be made with more advanced architecture.

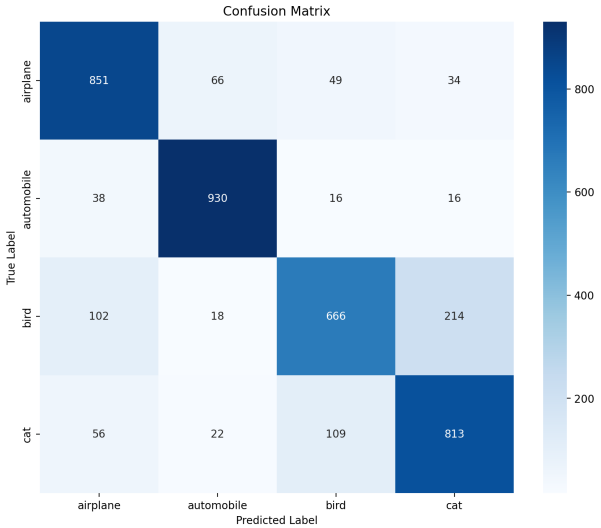

Figure 9: Confuion MAtrix for DIFAR

4.6.2 Fashion Minist kown flip rate

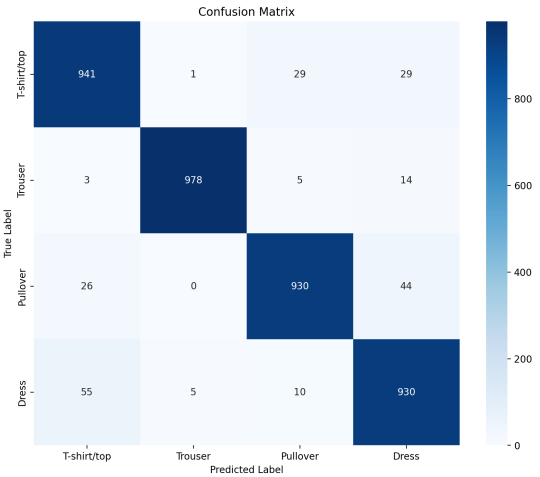

Figure 10: Confusion Matrix for DIFAR

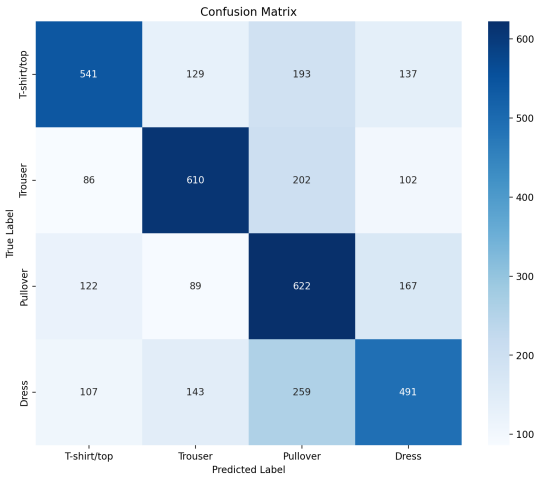

Figure 11: Confusion Matrix for DIFAR

This model CNN architecture did reasonably well for the Fashion MNIST dataset that had less noise-it was able to achieve a test accuracy of about 94.48 %. Not bad, given it is a relatively small model. By comparison, the state-of-the-art solution was FEMNIST, which achieved an accuracy of 99.06 %, but on the clean training dataset. There were no instances of any particular category doing substantially worse on the test set. On the noisier version

of the dataset, however, the accuracy reduced to 56.6%. This decline is expected since the added noise will act to desiderate class distinctions in the model, thus impacting performance.

4.7 Standard Deviations on same Hyperparameters

Standard deviation under the same hyperparameter setting, 10-fold cross-validation on the test set is relatively stable for all datasets. In particular, for the Fashion MNIST dataset, we see that as the noise of the dataset increases, the mean of standard deviation almost doubles. Such an increase in variability is theoretically expected when considering noisy labels.

High label noise levels inject randomness into the learning process since the model tries to fit to the mislabeled data or the data that is not consistent. This type of noise interferes with a classifier's generalization ability, causing more fluctuation in accuracy concerning the validation fold. This finding is consistent with previous work on noisy datasets, which shows that label noise affects model convergence and stability significantly [22, 16].

Unlike the behavior observed in the Fashion MNIST datasets, the CIFAR-10 dataset presents a higher standard deviation even though it has a lower level of noise estimated to about 20% by the our proposed transition matrix. The difference can be attributed to the nature of the dataset itself and the complexity of its images.

CIFAR-10 is inherently more difficult for classification because it has richer, more varied image content and higher dimensionality. Each image in CIFAR-10 represents a full-color scene with diverse textures and objects that may bring variability in how the model learns the features. On the other hand, grayscale images in Fashion MNIST are much simpler than CIFAR-10 because they relate to a uniform structure of shapes; hence, CIFAR-10 requires the model to discern more complex patterns and hence more susceptible to changes in learning as it tries to grasp useful features across folds.

Dataset	Mean	Standard Deviation	Minimum	25th Percentile	Median	75th Percentile	Maximum
Fashion MNIST 0.3	0.0163	0.0180	0.0028	0.0087	0.0120	0.0146	0.0830
Fashion MNIST 0.6	0.0280	0.0236	0.0035	0.0071	0.0165	0.0540	0.0695
CIFAR-10	0.0521	0.0163	0.0119	0.0447	0.0594	0.0637	0.0732

Table 1: Cross-validation statistics for each dataset across fixed hyperparameters

4.8 Personal Reflection

We can see that our label-noise robust CNN performed worse than the MLP with the transition matrix. The reason is that the noise in FashionMNIST 0.6 is too severe, and merely changing and designing the loss function cannot effectively learn the patterns in such noisy data. Therefore, we believe that both methods have their own specific scopes of application.

5 Conclusion

Dataset Name	Model	Mean Val Accuracy	Mean Accuracy	Standard Deviation
FashionMNIST03	MLP	0.6818	0.9390	0.0034
FashionMNIST06	MLP	0.3701	0.8150	0.0075
CIFAR10	MLP	0.6100	0.6826	0.0093
FashionMNIST03	CNN	0.8096	0.8911	0.0655
FashionMNIST06	CNN	0.4302	0.5660	0.0695
CIFAR10	CNN	0.6920	0.9448	0.0154

Table 2: Comparison of Different Models on Various Datasets

In this study, we aimed to develop robust methods for handling label noise in classification tasks, using a label-noise robust CNN with a specially designed loss function and an MLP enhanced with a transition matrix. Our experiments on noisy datasets like FashionMNIST and CIFAR-10 demonstrated the effectiveness of both methods in moderate noise conditions.

When noise levels are moderate, such as in the FashionMNIST 0.3 and CIFAR-10 datasets, the label-noise robust CNN, with its specially designed loss function, demonstrates greater robustness. However, when noise is more

severe, the MLP with transition matrix approach has an advantage. Additionally,, due to the large number of parameters that need updating, the label-noise robust CNN requires a longer optimization time.

For future work, we propose exploring adaptive methods for transition matrix estimation and investigating other robust loss functions to enhance model resilience across varying noise levels. Additionally, we will extend this study to more complex datasets would provide further insights into the generalizability of these methods.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019.

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, page 2546–2554, 2011.

[3] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. *arXiv preprint arXiv:1712.09482*, 2017.

[4] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. *arXiv preprint arXiv:1609.03683*, 2016.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[6] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Ivor W Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8527–8537, 2018.

[7] Md Mosharaf Hossain, Sheikh Ghafoor, and Ramakrishnan Kannan. Fawca: A flexible-greedy approach to find well-tuned cnn architecture for image recognition problem. 08 2018.

[8] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2304–2313, 2018.

[9] Shivam Kadam, Amol Adamuthe, and Ashwini Patil. Cnn model for image classification on mnist and fashion-mnist dataset. *Journal of scientific research*, 64:374–384, 01 2020.

[10] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. Accessed: November 1, 2024.

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romano, Sarah Erfani, and James Bailey. Normalized loss functions for deep learning with noisy labels. *Proceedings of the 37th International Conference on Machine Learning*, pages 6543–6553, 2020.

[13] Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romano, Sarah Erfani, and James Bailey. Normalized loss functions for deep learning with noisy labels, 2020.

[14] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. *arXiv preprint arXiv:1609.03683*, 2017.

[15] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2005.

[16] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

[17] Arnaud Rosay, Kévin Riou, Florent Carlier, and Pascal Leroux. Multi-layer perceptron for network intrusion detection: From a study on two recent data sets to deployment on automotive processor. *Annales des télécommunications*, 2022.

[18] Sheldon Ross. *Introduction to Probability and Statistics for Engineers and Scientists*. Academic Press, 2014.

[19] Shuhei Watanabe. Bayesian optimization techniques and applications. *Optimization in Machine Learning*, 2023.

[20] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[21] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2017.

[22] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2017.

[23] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*, 2018.

A Additional Data

A.1 Hyperparameters Importance Accuracy

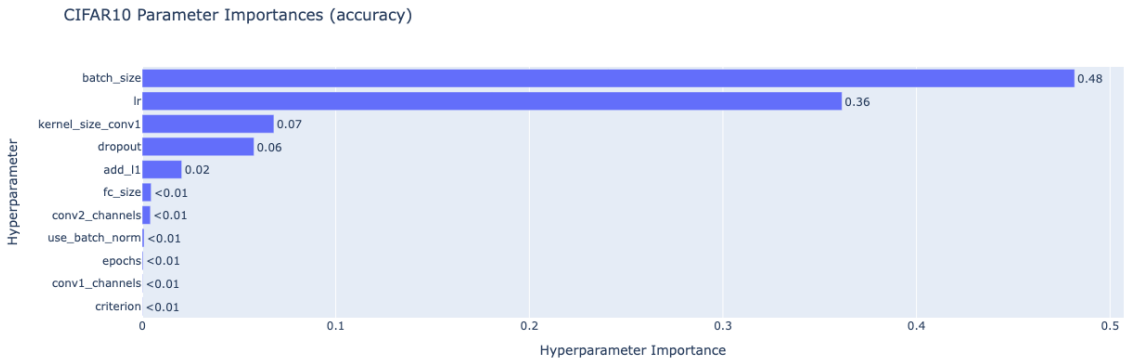

Figure 12: HP Importance on CIFAR-10

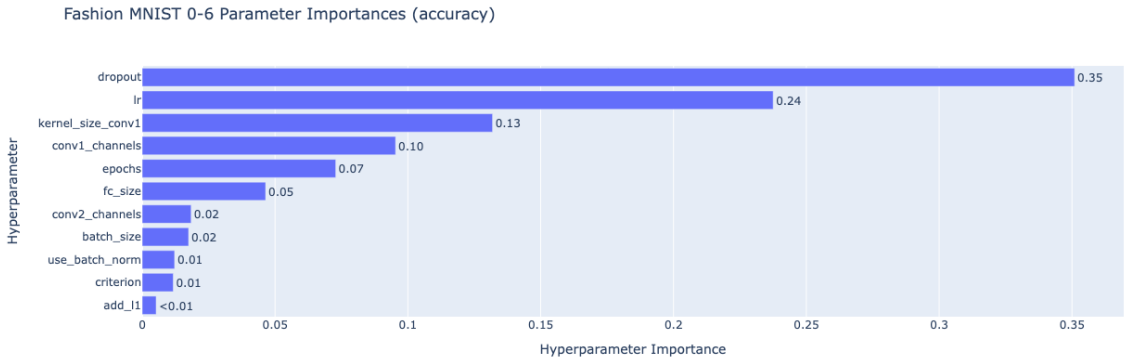

Figure 13: HP Importance on Fashion MNIST 0-6

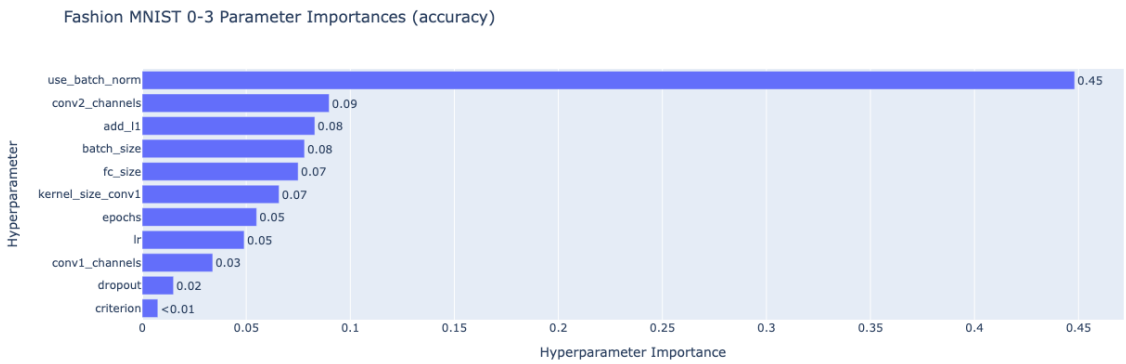

Figure 14: HP Importance on Fashion MNIST 0-3

A.2 Hyperparameters Importance Accuracy Standard Deviation

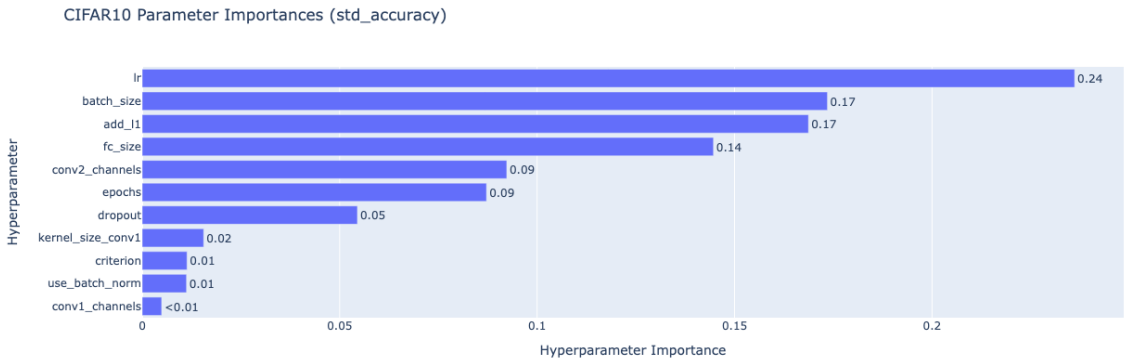

Figure 15: HP Importance on CIFAR-10

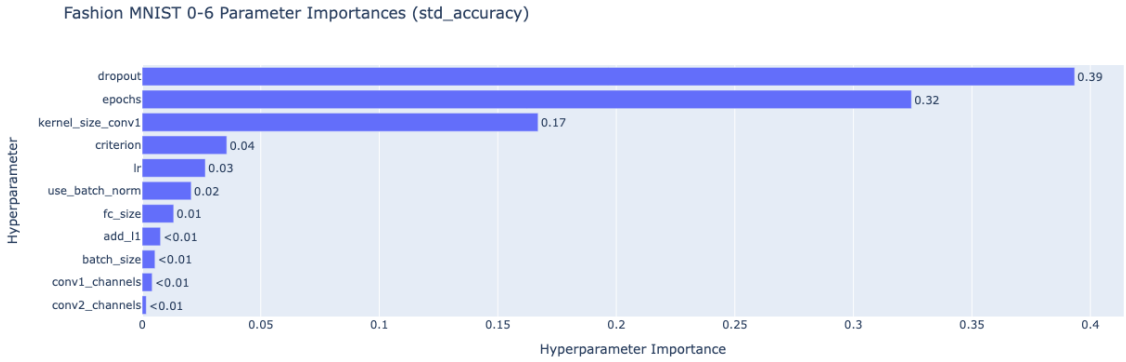

Figure 16: HP Importance on Fashion MNIST 0-6

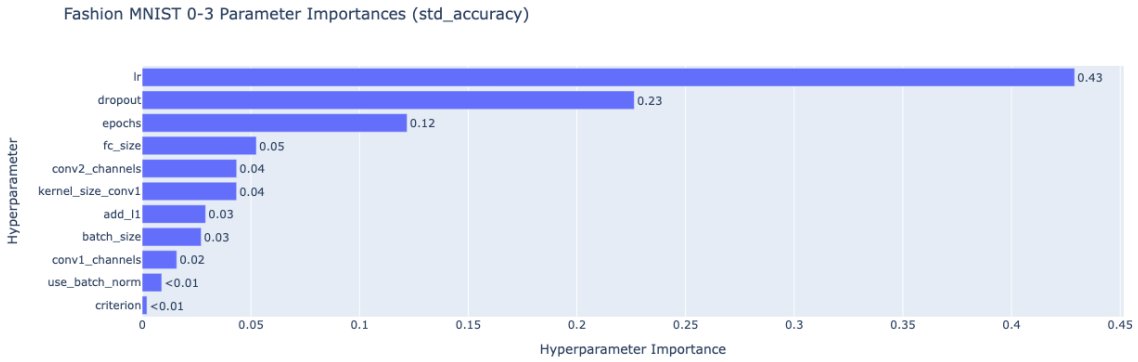

Figure 17: HP Importance on Fashion MNIST 0-3

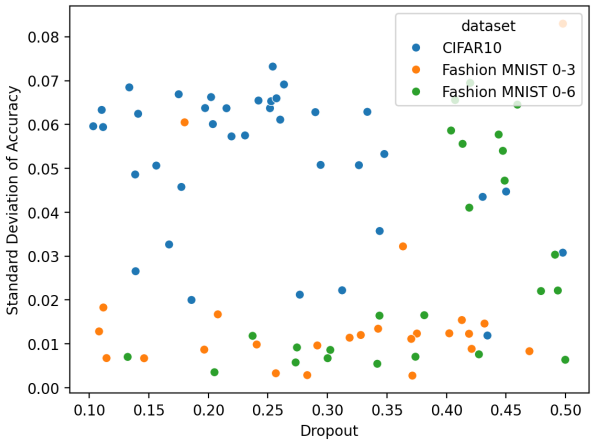

Figure 18: Dropout Rate vs. Accuracy Standard Deviation

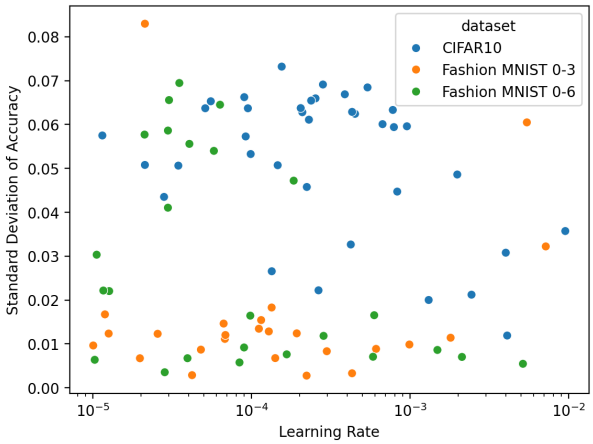

Figure 19: Learning Rate vs. Accuracy Standard Deviation