Artificial Intelligence Course Project
Object Classification by Multi Layer Perceptron

Shakiba Fotouhi

December 2025

Contents

List of Figures

List of Tables

1 Introduction

The primary objective of this project was to implement multilayer perception (MLP) for the object classification task. The secondary objective is to find more difficult classes for classification task by the developed model. The developed Multi-Layer Perceptron (MLP) was evaluated across two datasets to test its generalization capability: the **CIFAR-10** dataset and the **Fashion-MNIST** dataset.

The CIFAR-10 dataset consists of 60,000 32x32 colour images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images [1].

Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes [2].

The primary objectives of this project were:

- To implement and train an MLP using the PyTorch framework.
- To evaluate the overall accuracy of the model on the test sets.
- Analyze class-wise performance to identify the most difficult classes for this architecture.

The complete source code, training history, and environment setup for this project are available for full reproducibility in the accompanying GitHub repository and interactive Google Colab notebook.

2 Methodology

2.1 Dataset and Pre-processing

Data pre-processing consist of a two-stage procedure tailored to each dataset used in this work. First, all images were converted into PyTorch tensors using the `transforms.ToTensor()` function. This step transformed the raw PIL Image objects—loaded internally by `torchvision.datasets`—into tensors while scaling pixel values from the 0–255 range to floating-point values between 0.0 and 1.0.

The second stage applied dataset-specific normalization. For CIFAR-10, which has three-channel RGB images of size 32×32, each color channel was normalized using a mean of (0.5, 0.5, 0.5) and a standard deviation of (0.5, 0.5, 0.5). On the other hand, Fashion-MNIST images consist of a single grayscale channel with size of 28×28, and a single-channel mean and standard deviation of 0.5 were used for normalization. These transformations ensured that both datasets were standardized appropriately before being fed into the MLP model.

The CIFAR-10 training and testing sets consist of 50,000 and 10,000 images, respectively. For Fashion-MNIST these numbers are 60,000 and 10,000. The goal is to classify these sets. The CIFAR-10 dataset classes include: Plane, Cat, Dog, Bird, Frog, Truck, Deer, Ship, Car,and Horse. The Fashion-MNIST Classes are: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot.

The CIFAR-10 dataset and the Fashion-MNIST dataset were downloaded using `torchvision.datasets` module and then loaded in batches of 64 using PyTorch's `DataLoader` with enabled shuffling for the training set.

2.2 Model Architecture

The implemented classifier was a four-layer MLP using `torch.nn.Module` (an input layer, two hidden layers, an output layer).

1. **Input Layer:** Image tensor (Channels × Height × Width) was converted into a single feature vector with an initial `nn.Flatten()` operation. For CIFAR-10 the size was set to 3,072 (3×32×32) and for Fashion-MNIST it was set to 784 (1×28×28).

2. **Hidden Layers:** Two layers, using the **Rectified Linear Unit (ReLU)** activation function to introduce non-linearity:

   - Hidden Layer 1: *Input*→512 neurons.
   - Hidden Layer 2: 512→128 neurons.

3. **Output Layer:** The 128-neuron output was mapped to 10 nodes in the final layer, which is the number of distinct classes in both datasets.

2.3 Training Parameters

- **Loss Function:** The CrossEntropyLoss (`nn.CrossEntropyLoss`) was chosen, which is primarily used for single-label multiclass classification problems, where each input sample belongs to exactly one class out of several possible classes.

- **Optimizer:** The Adam optimizer (`optim.Adam`) was utilized with a learning rate (α) of 0.001.
- **Epochs:** The model was trained for 10 epochs.

2.4 History and Metrics Tracking

The `train_model` function is modified to save the **training loss**, **test loss**, and **test accuracy** at the end of every epoch into dedicated history lists.

- **Training Loss** is calculated as the average loss over all batches processed in a given epoch.
- **Test Loss** and **Test Accuracy** are computed by the `evaluate_model_metrics` helper function, which iterates over the separate `testloader` with the model in `eval()` mode and disabled gradient calculation (`torch.no_grad()`). The test loss is accurately calculated as the weighted average across the entire test set.

These history lists are then passed to the `plot_learning_curves` function to visually inspect model performance over the course of training. This visualization, typically showing **Loss vs. Epochs** and **Accuracy vs. Epochs**, is critical for understanding the model's convergence behavior.

2.4.1 Final Evaluation and Confusion Matrix

The final step of the methodology is a thorough evaluation of the trained model's performance on the unseen test dataset using the `final_evaluation_and_confusion_matrix` function.

1. **Overall Accuracy Calculation:** The final Overall Test Accuracy is calculated across the entire test set.
2. **Confusion Matrix Generation:** The function gathers all true labels and corresponding predictions from the test set to generate a Confusion Matrix (CM) using the `confusion_matrix` utility from `scikit-learn`.
3. **Visualization and Analysis:** The CM is then passed to `plot_confusion_matrix`, where it is normalized by row (True Label total) before plotting using the `seaborn` library. Normalization ensures that the values within the matrix represent the **True Positive Rate** (recall/sensitivity) for each class.
4. **Class-wise Accuracy:** By analyzing the normalized diagonal elements of the CM, the class-wise accuracy is explicitly calculated and reported. This analysis is crucial for identifying the most difficult class (the one with the lowest classification accuracy), which highlights specific weaknesses in the model's feature extraction or representational capabilities for that category.

In order to ensure statistically reliable results and mitigate the influence of random weight initialization, the performance of the model was rigorously tested over ten independent runs for each dataset.

3 Results

3.1 CIFAR-10 Results

Following table shows overall accuracy and most difficult class to classify after ten epochs of training in ten runs.

Run	Most Difficult Class	Overall Accuracy (%)
1	Cat (34.80%)	53.56
2	Cat (35.00%)	53.37
3	Dog (34.30%)	52.63
4	Cat (23.00%)	53.43
5	Cat (33.70%)	53.47
6	Bird (33.80%)	53.66
7	Bird (34.40%)	53.14
8	Bird (40.40%)	53.50
9	Cat (39.10%)	53.69
10	Bird (37.30%)	53.37

Table 1: Overall accuracy and most difficult class of CIFAR-10 over ten runs

Overall Accuracy

The mean overall accuracy across the ten runs is: 53.38 % $\pm$ 0.31 %. The variance is relatively low, this means that the MLP behaves consistently.

The most difficult class across all runs was Cat with mean Accuracy of 35.57%.

Most Difficult Class to Classify

The results across the ten runs show a consistent pattern in class-level difficulty. The **cat** category appears as the lowest-accuracy class in five out of ten runs, making it the most difficult class for the MLP to distinguish. The **dog** and **bird** classes appear once and four times as the lowest-performing category. This aligns with expectations for CIFAR-10, where visual similarity between animal classes and the limited representational power of an MLP often lead to confusion among these categories.

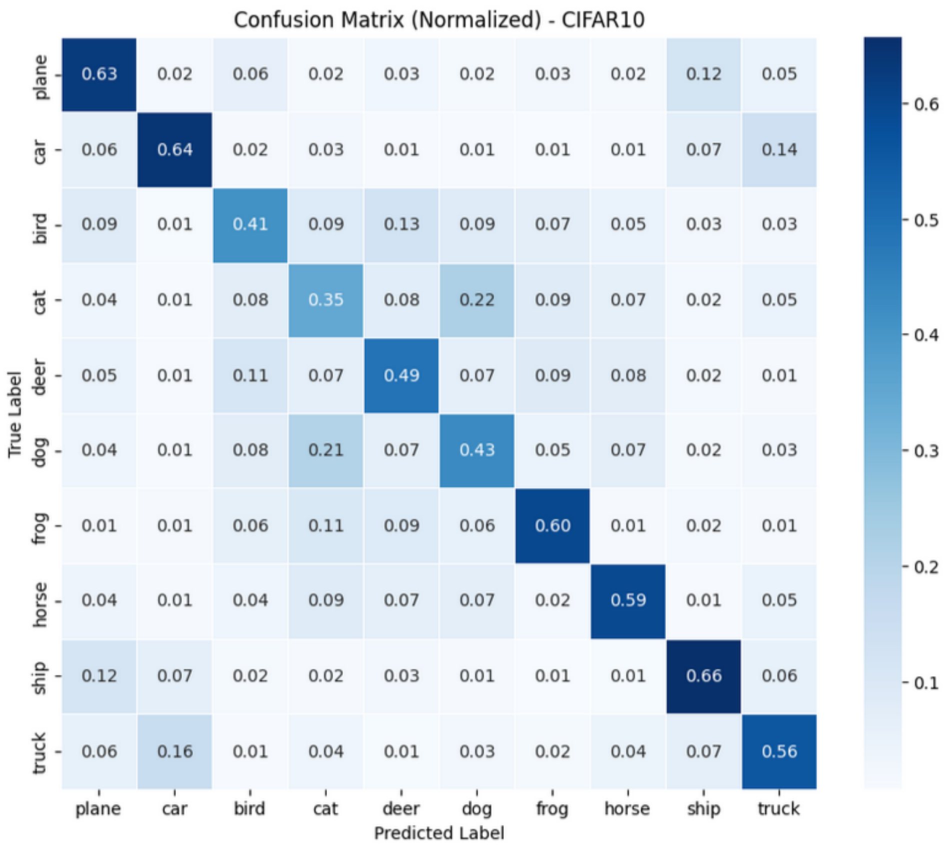
Class Name	Mean Accuracy
Cat	35.57%
Bird	39.67%
Dog	40.98%
Deer	46.96%
Truck	57.70%
Horse	60.11%
Frog	60.43%
Plane	63.69%
Ship	63.99%
Car	64.72%

Table 2: Mean class accuracies of CIFAR10 across ten runs sorted by difficulty

.

The most difficult classes for object classification were Cat, Bird, and Dog.

In contrast, Car, Ship, and Plane classes had highest mean accuracy. This results show that the model performs best for classifying objects with distinct shapes or backgound.



	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
plane	0.63	0.02	0.06	0.02	0.03	0.02	0.03	0.02	0.12	0.05
car	0.06	0.64	0.02	0.03	0.01	0.01	0.01	0.01	0.07	0.14
bird	0.09	0.01	0.41	0.09	0.13	0.09	0.07	0.05	0.03	0.03
cat	0.04	0.01	0.08	0.35	0.08	0.22	0.09	0.07	0.02	0.05
deer	0.05	0.01	0.11	0.07	0.49	0.07	0.09	0.08	0.02	0.01
dog	0.04	0.01	0.08	0.21	0.07	0.43	0.05	0.07	0.02	0.03
frog	0.01	0.01	0.06	0.11	0.09	0.06	0.60	0.01	0.02	0.01
horse	0.04	0.01	0.04	0.09	0.07	0.07	0.02	0.59	0.01	0.05
ship	0.12	0.07	0.02	0.02	0.03	0.01	0.01	0.01	0.66	0.06
truck	0.06	0.16	0.01	0.04	0.01	0.03	0.02	0.04	0.07	0.56

Figure 1: Confusion matrix of run no.1

The confusion matrix above shows that the model experienced more difficulty while classifying more similar looking objects. For instance for classifying cat images, 22% were misclassified as dogs and 21% of dogs were misclassified as cats.11% of true deer were mistaken for Bird and A high number of true birds were misclassified as Deer (13%) and Cat (9%).

Another confusion showed by the figure is related to Car and Truck classes. As represented, 16% of true trucks are incorrectly predicted as Car.

Training and Testing Curves

Following figure shows training and test loss per epoch on the left side and test accuracy per epoch on the right side.

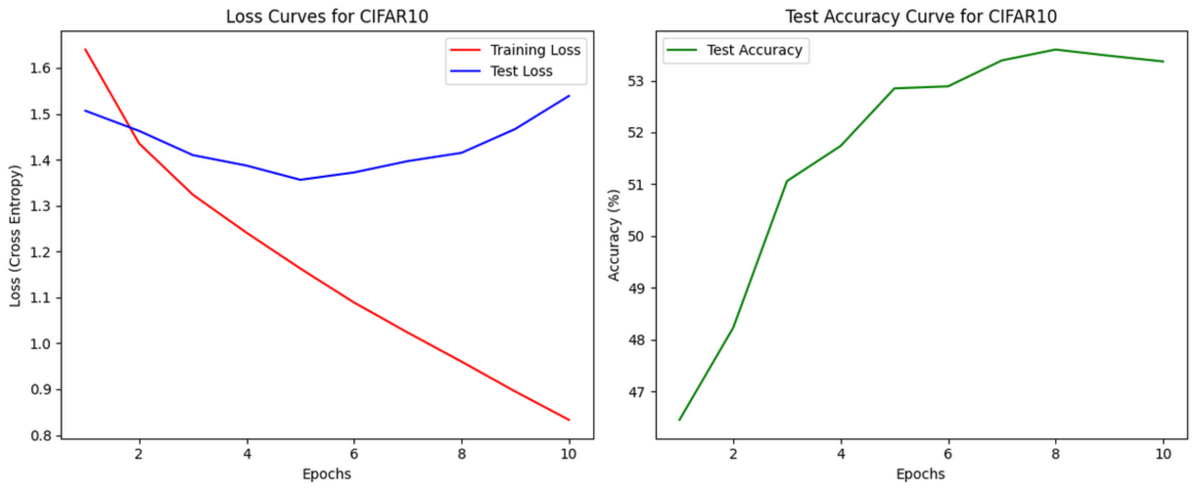


Figure 2: Loss curve and test accuracy per epoch for CIFAR10

The training loss sinks smoothly from roughly 1.65 to around 0.82 across 10 epochs. That's the model getting better at memorizing the training data. On the other hand, the test loss, is degrading. It declines for the first few epochs, bottoms out around epoch 4 or 5, and then starts climbing again.

Meanwhile, test accuracy climbs from about 46% to roughly 53–54% and then levels off.

Considering two curves together, combination of improved accuracy but worsening test loss indicate that the model overfitetd at around epoch 5 and 6.

3.2 Fashion-MNIST Results

Following table shows the results of overall accuracy and most difficult class for classification task over ten epoches of training and ten runs on Fashion-MNIST dataset.

Run	Most Difficult Class	Overall Accuracy (%)
1	Shirt (59.40%)	87.61
2	Shirt (68.20%)	88.44
3	Shirt (70.80%)	88.49
4	Shirt (71.30%)	89.14
5	Shirt (69.30%)	88.53
6	Shirt (60.20%)	87.90
7	Shirt (68.30%)	88.56
8	Shirt (62.80%)	88.67
9	Shirt (68.20%)	88.50
10	Shirt (64.10%)	88.70

Table 3: Overall accuracy and most difficult class of Fashion-MNIST over ten runs

Overall Accuracy

The mean overall accuracy across the ten runs was: 88.45% ± 0.42%
The most difficult class across all runs was Shirt with mean accuracy of 66.26%.

Most Difficult Class to Classify

Across the ten runs on the Fashion-MNIST dataset, the shirt category consistently emerged as the lowest-accuracy class, making it the most difficult item type for the MLP to recognize. This outcome is typical for Fashion-MNIST because shirts share visual characteristics with several neighboring categories—most notably t-shirt/top and coat—leading to frequent misclassifications. The limited feature-extractive capacity of an MLP further amplifies these ambiguities, causing the model to struggle when distinguishing subtle differences in garment shape and texture.

Class Name	Mean Accuracy
Shirt	66.26%
Pullover	81.65%
Coat	82.84%
T-shirt/top	84.49%
Dress	90.56%
Sneaker	94.23%
Ankle boot	94.91%
Sandal	95.72%
Bag	96.79%
Trouser	97.09%

Table 4: Mean class accuracies of Fashion-MNIST across 10 runs sorted by difficulty

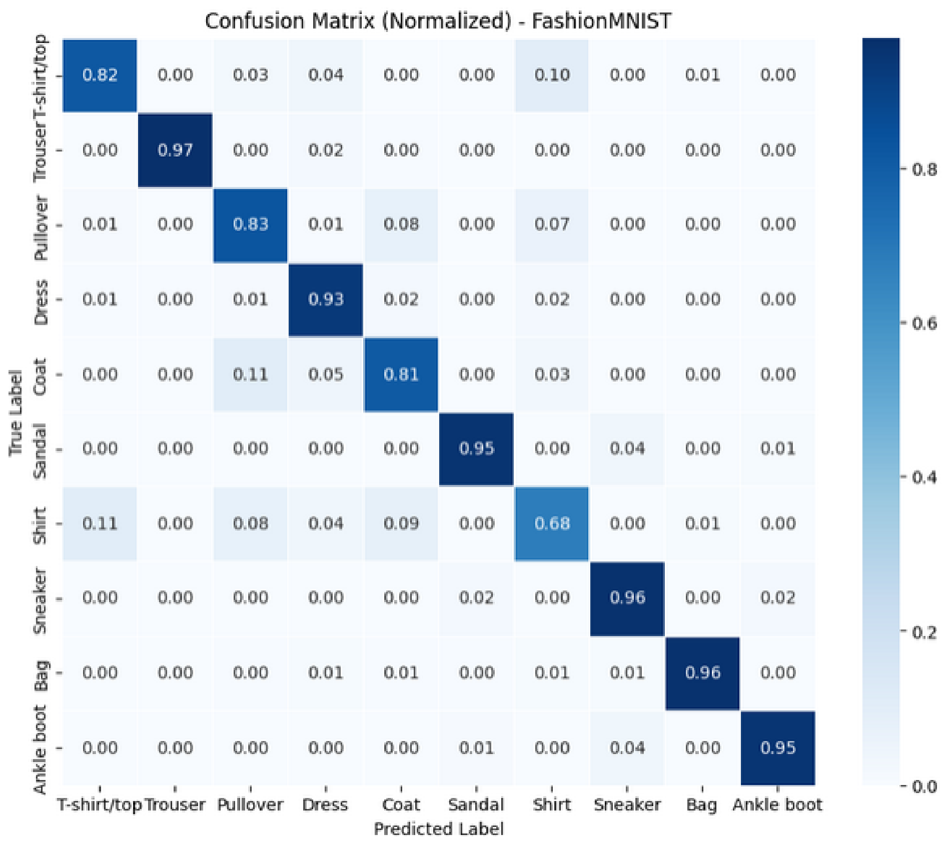


Figure 3: Sample confusion matrix of Fashion-MNIST

As shown in the confusion matrix, 11% of true shirt were misclassified as T-shirt/top, 9% were misclassified as Coat and 8% were misclassified as Pull over. Another confusion is related to Pullover and Coat classes. As shown by the confusion matrix, 11% of true coats were misclassified as Pullovers.

Training and Testing Curves

Following figure shows training and test loss per epoch on the left side and test accuracy per epoch on the right side.

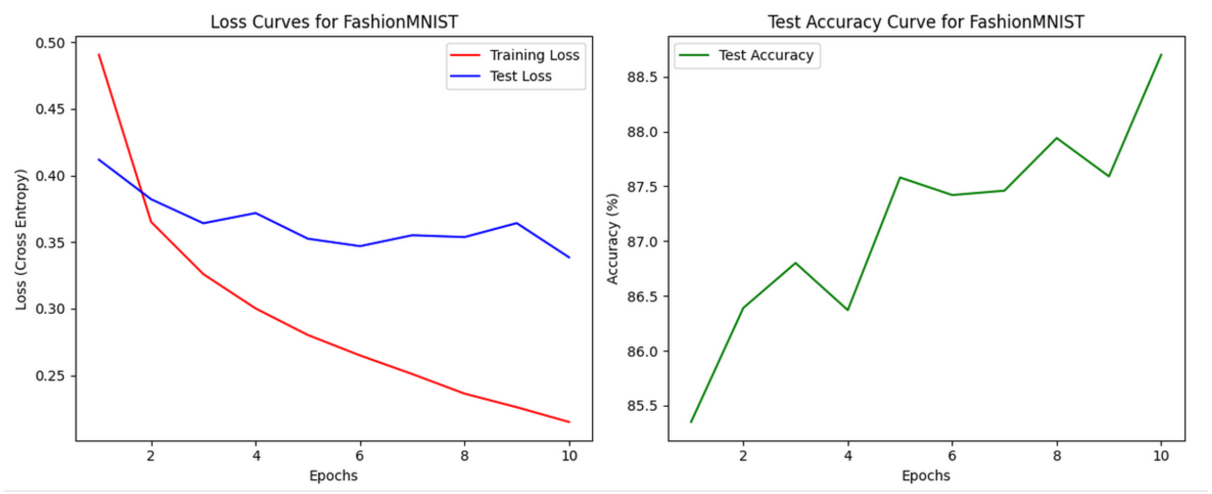


Figure 4: Loss curve and test accuracy per epoch for Fashion-MNIST

.

The training loss decreases steeply throughout all 10 epochs, indicating that the model is continuously learning and fitting the training data well.

The test loss initially decreased but began to flatten out and slightly fluctuated around epoch 5-6. It remained relatively close to the training loss, suggesting that the model was generalizing reasonably well to the unseen data without severe overfitting.

The Test Accuracy shows a strong overall upward trend, starting around 85.5% and finishing near 88.5% by epoch 10.

3.3 Comparative Analysis Between the Two Datasets

There is a significant difference in overall accuracy of two datasets (88.45% for Fashion-MNIST vs. 53.38% for CIFAR-10) which shows that the trained MLP performed much better on the grayscale, simpler, and high contrast Fashion-MNIST data than on the complex, colored CIFAR-10 images due to limited feature extraction capabilities. Which is why Convolutional Neural Networks (CNNs) were developed, as they preserve and process spatial information.

Both datasets are challenging in different ways; Fashion-MNIST is challenging due to subtle inter-class similarities (e.g., Shirt vs. T-shirt/top), while CIFAR-10 is challenging due to the complexity of features in 32×32 color images (e.g., Cat vs. Dog).

For CIFAR-10 dataset, by flattening the 32×32 color image into a 3,072-dimensional vector, model destroys all spatial correlation. This means the model must memorize features based only on where those features happen to appear in the flattened vector, which requires heavy reliance on the training data.

Since CIFAR-10 contains complex, real-world images with backgrounds, varying lighting, and subtle color details. These details act as noise when spatial context is lost. In the high 3,072-dimensional input space, the MLP fit this noise in the training set (driving the training loss down), but these complex rules fail spectacularly on the unseen test set (driving the test loss up).

In addition, the confusion between Cat, Dog, and Bird is difficult to resolve without preserved spatial relationships.

In summary, the complexity of these features in a non-spatial context causes the model to break down when attempting to generalize to new instances, leading to the rapid test loss increase seen in the CIFAR-10 curve.

However, considering the results for Fashion-MNIST dataset the loss of spatial correlation is less detrimental for two key reasons:

Firstly, Fashion-MNIST classes (like Trouser, Bag, or Sneaker) rely on gross, high-contrast shapes and outlines. These features are often recognizable even when the 28×28 grayscale image is flattened.

Secondly, the input vector is significantly smaller (784 features) than CIFAR-10's 3,072 features. This dramatically reduces the complexity of the first layer's weight matrix, making the learning task simpler and less prone to finding spurious patterns in the noise.

3.4 Brief Discussion of Limitations

The most significant limitation is the choice of the Multi-Layer Perceptron (MLP).

First of all, as explained previously the MLP model requires the image tensor to be converted into a single, flat feature vector. This operation fundamentally destroys the crucial spatial correlation and the two-dimensional grid structure of the image. When spatial context is lost, the model must essentially memorize features based only on their position in the flattened vector, which leads to the severe overfitting and poor generalization seen in the CIFAR-10 results.

The model's generalization ability would be significantly improved by replacing the initial MLP layers with a Convolutional Neural Network (CNN) architecture . CNNs use filters to extract local, translational-invariant features while preserving the spatial relationships between pixels, making them the standard choice for complex image recognition tasks.

Secondly, the training was strictly limited to 10 epochs. While 10 epochs were sufficient to show clear signs of overfitting on CIFAR-10, the simpler Fashion-MNIST task might have benefited from slightly extended training. Also, no rigorous hyperparameter tuning was performed on the learning rate (α=0.001) , optimizer choice (Adam) , or the size of the hidden layers (512-128 neurons). Systematic tuning could potentially improve the performance on the simpler Fashion-MNIST dataset beyond the observed 88.45% mean accuracy.

Lastly, both datasets are relatively small (50,000 to 60,000 training images), the MLP's limited capacity prevents it from utilizing the data as effectively as a CNN. An MLP cannot hierarchically build complex, generalized features like a CNN. If the project were extended to a much larger and more complex dataset, the MLP's fixed capacity would plateau quickly, whereas a CNN could leverage the increased data volume to learn richer, more robust features.

4 References

References

[1] "Cifar-10 dataset." https://www.cs.toronto.edu/~kriz/cifar.html. Accessed: 2025-12-6.

[2] "Fashion-mnist dataset." https://github.com/zalandoresearch/fashion-mnist. Accessed: 2025-12-6.