Image data classification with MLP and CNN

1 Abstract

This research project delves into the practical applications and comparative analysis of Convolutional Neural Networks (CNN) [1] and Multilayer Perceptrons (MLP) [5] using the FashionMNIST and CIFAR10 datasets. The primary aim was to discern the disparities between MLP and CNN models and explore the advantages of convolutional layers within CNNs. In the context of the FashionMNIST dataset, a comparison was conducted between the performance of a MLP model and a CNN solution utilizing the Stochastic Gradient Descent (SGD) optimizer. The findings unequivocally favored the CNN model. Subsequently, when evaluating the CIFAR10 dataset, the study extended its analysis by testing various optimization techniques, specifically SGD and Adam. Notably, the Adam optimizer delivered superior results. This research illuminates the practical use of CNNs, showcasing their superior performance in image tasks, enhancing our understanding of deep learning models' adaptability to diverse datasets.

2 Introduction

This research project focuses on a comparative analysis of MLP and CNN for image classification applications. To facilitate this comparison, we employ the "Fashion MNIST" dataset.In our initial experiments with the Fashion MNIST dataset, we observed that the CNN model achieved an impressive accuracy of 90%, significantly outperforming the MLP model. This superior performance can be attributed to the CNN's inherent capacity to capture and interpret spatial relationships between pixels in images.Building upon these findings, we extended our investigation to the CIFAR10 dataset, where we systematically compared the performance of different optimizers and hyperparameter configurations. Notably, the Adam optimizer demonstrated a remarkable accuracy of 92%, surpassing the results obtained with the SGD optimizer. This enhanced performance may be attributed to the adaptive learning rate properties of the Adam optimizer.Furthermore, we explored the utilization of a pretrained ResNet model with additional fully connected layers. The results of this modified ResNet model were compared with those of the previously studied MLP and CNN models.Our analysis includes extensive visualizations, graphs, and tables, providing a thorough examination of machine learning algorithms on out-of-distribution datasets.

3 Datasets

3.1 Fashion MNIST

The Fashion MNIST dataset contains 60,000 training and 10,000 testing grayscale images of fashion items, each represented as a 28x28 pixel vector. These items are categorized into ten classes, that includes include "T-shirt/top," "Trouser," "Pullover," "Dress," "Coat," "Sandal," "Shirt," "Sneaker," "Bag," and "Ankle boot". We normalized pixel values to [0, 1] and applied Principal Component Analysis to reduce data to two components for visualization in a scatter plot, as depicted in Figure 1.

3.2 CIFAR-10

The CIFAR-10 dataset encompasses 60,000 color images, meticulously distributed into ten classes, each hosting 6,000 images. These images exhibit a modest resolution of 32x32 pixels. The classes within CIFAR-10 are denoted as "Airplane", "Automobile", "Bird", "Cat", "Deer", "Dog", "Frog", "Horse", "Ship" and "Truck". Similar to the Fashion MNIST dataset, we executed a normalization process on the CIFAR-10 dataset. Here, PCA reduced the data to two principal components, similar to the Fashion MNIST case. A scatter plot was generated to visualize the data in two dimensions, with class labels represented by colors as shown in Figure 2.

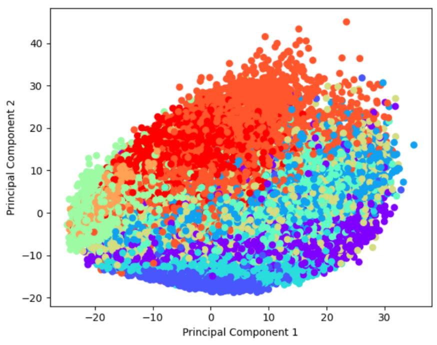

Fig.1 Fashion MNIST Visualization

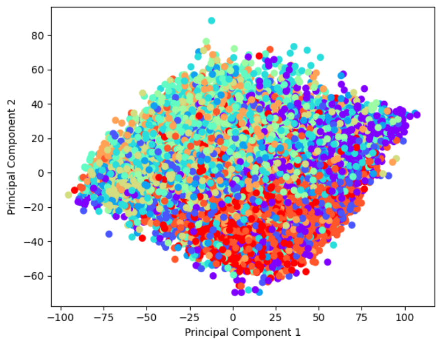

Fig.2 CIFAR10 Visualization

4 Results

4.1 Fashion MNIST

4.1.1 Experiments on Weight Initialization

Weight initialization plays a crucial role in the training and performance of a neural network. They have a significant impact on the convergence and performance of the neural network.

Xavier Initialization: In Xavier initialization, weights are sampled from a normal distribution with mean, $\mu = 0$, and variance, $\sigma^2 = \frac{2}{F_{in}+F_{out}}$. Empirical results suggest that this method facilitates faster convergence when compared to other initialization techniques. The respective training curves and testing accuracies are tabulated below.

All-zeros Initialization: Opting for an all-zeros initialization renders the neural network symmetric, making it the most unstable initialization approach. Such symmetry can hinder the network from learning efficiently as it may not update its weights effectively. The training and testing accuracies are observed to be nearly zero. Relevant curves and accuracy figures are presented below.

Uniform Initialization: For uniform initialization, weights are selected from a uniform distribution spanning the range $[-1, 1]$. Observations indicate that with this approach, the model successfully converges to an optimal solution. Pertinent training curves and testing accuracy figures can be found below.

Gaussian Initialization: In this strategy, weights are initialized based on a Gaussian (or normal) distribution with a mean of 0 and a standard deviation of σ. The associated training curves corroborate that the model tends towards an optimal convergence point. Both the training progression and the testing results are elaborated on below.

Kaiming Initialization: Kaiming initialization is predominantly favored for training deep neural networks, particularly those utilizing ReLU or analogous activation functions. This method can also employ a normal distribution, characterized by $\mu = 0$ and $\sigma^2 = \frac{2}{F_{in}}$. The forthcoming sections will shed light on the training curve and testing accuracies obtained using this method.

Initialization Type	Xavier	All Zero	Uniform	Gaussian	Kaiming
Train accuracy	0.9695	0.1002	0.8805	0.8973	0.9811
Test accuracy	0.8844	0.0991	0.8517	0.8519	0.8817

Table 1: Train and Test accuracies on different types of initialization

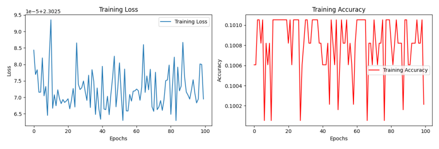

Fig.3 Zero Initialization

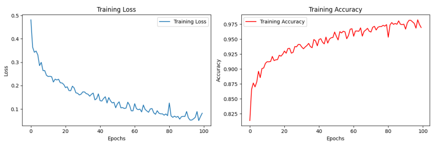

Fig.4 Xavier Initialization

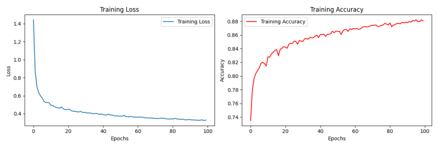

Fig.5 Uniform Initialization

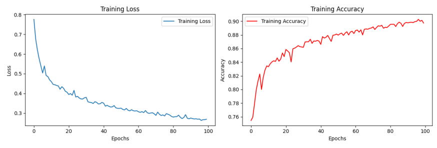

Fig.6 Gaussian Initialization

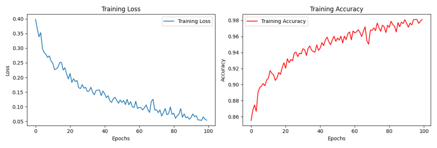

Fig.7 Kaiming Initialization

4.1.2 Comparison with different number of hidden layers

Three MLP models were evaluated on the Fashion MNIST dataset:

1. **Linear Model:** Direct mapping from inputs to outputs without hidden layers.
2. **Single Hidden Layer Model:** One hidden layer with 128 units using ReLU activation.
3. **Dual Hidden Layer Model:** Two hidden layers, each with 128 units and ReLU activation.

Observations

- The **Linear Model**, lacking non-linearity, is expected to have limited accuracy due to its inability to capture complex patterns in the dataset. We can clearly observe that in the training and testing accuracies.
- The **Single Hidden Layer Model** introduces non-linearity via ReLU, enabling it to recognize more intricate patterns, likely leading to better performance than the linear model.
- The **Dual Hidden Layer Model**, with even more capacity, could potentially achieve the highest accuracy, but there's a risk of overfitting. The training and testing accuracies are mentioned below.

Number of Layers	0 layers	1 layers	2 layers
Train accuracy	0.8677	0.9334	0.9656
Test accuracy	0.8525	0.8880	0.8893

Table 2: Train and Test accuracies on different number of layers

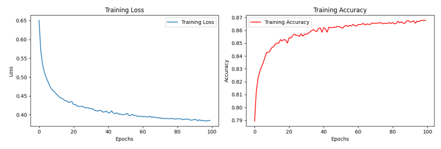

Fig.8 Zero-layer

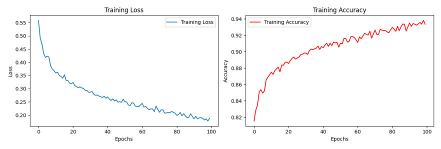

Fig.9 One Layer

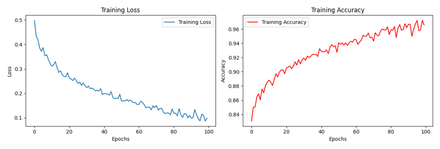

Fig.10 Two Layers

4.1.3 Comparision of different Activation Functions

From the previously discussed architectures, the MLP model with two hidden layers was selected. Two variants of this model were created by replacing the ReLU activations:

1. **Model with Leaky ReLU:** This model incorporates the Leaky ReLU activation function.
2. **Model with Softplus:** The Softplus activation function is employed in this variant.

Observations:

- **Model with Leaky ReLU:** The Leaky ReLU activation aims to address the dying ReLU problem. By allowing a small gradient when the unit is inactive, it ensures that neurons remain slightly activated.
- **Model with Softplus:** Softplus serves as a smooth approximation to the ReLU function. It is differentiable everywhere and can help avoid dead neurons.
- We observed that the model performed better in the case of leaky Relu compared to Softplus because of its ability to allow a small gradient for negative input values leading to better performance than Softplus

Type of Activation	Leaky Relu	SoftPlus
Train Accuracy	0.9678	0.9084
Test Accuracy	0.8908	0.8794

Table 3: Train and Test accuracies on different activation functions

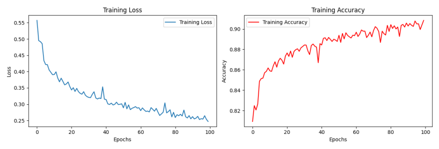

Fig.11 SoftPlus Activation Function

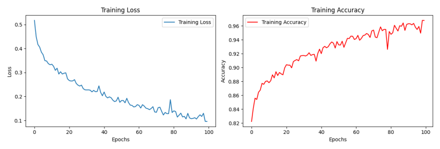

Fig.12 Leaky$_ReluActivationFunction$

4.1.4 Analysis of Regularization Techniques on MLP

An MLP model with two hidden layers, each containing 128 units and employing ReLU activations, was designed and trained. To assess the effects of regularization on performance, the network was subjected to independent L1 and L2 regularization methods.

- **L1 Regularization:** Also known as Lasso regularization, it adds a penalty equivalent to the absolute value of the magnitude of coefficients. L1 can lead to feature sparsity, where some feature weights become exactly zero, which may make the model simpler and more interpretable, but could also lead to a compromise on the model's capacity.
- **L2 Regularization:** Known as Ridge regularization, it penalizes the squared magnitude of coefficients. L2 tends to shrink the weights of less important features but rarely makes them exactly zero. This regularization helps in preventing overfitting by constraining the model's weight space without causing complete sparsity.

Regularization Type	L1 Regularization	L2 Regularization
Train accuracy	0.8865	0.9480
Test accuracy	0.8720	0.8855

Table 4: Train and Test on L1 and L2 regularizations

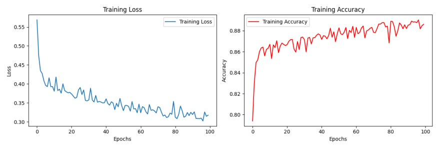

Fig.13 L1 regularization

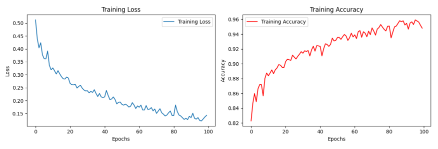

Fig.14 L2 regularization

4.1.5 Effect of Image Normalization on MLP Performance

Using unnormalized images means retaining pixel values in their original range, for instance, 0-255. Such values can lead to exponentially large activations, particularly when ReLU functions are in play. This explosive surge in values can either significantly slow down the convergence or, in some cases, prevent convergence altogether. As a consequence, the accuracy of the model trained with unnormalized images is lower than that of a model trained on normalized images. The accuracies for training set is **0.1001** and accuracy for Test set is **0.0911**

4.1.6 Fashion MNIST on CNN

In the experiment, a convolutional neural network (CNN) was constructed using PyTorch, consisting of 2 convolutional layers and 2 fully connected layers. While the hyperparameters of the convolutional layers were flexibly chosen, the units in the fully connected layers were set to 128. The ReLU activation function was employed across all layers. This CNN was then trained on the Fashion MNIST dataset. To determine the efficacy of the CNN, its accuracy was compared to a multi-layer perceptron (MLP). The results revealed that the CNN increased the accuracy in comparison to the MLP, highlighting the benefits of using convolutional layers for this dataset. The accuracies for training set is **0.9541** and accuracy on test set is **0.9147**. The training curves are shown in the appendix section

Model	Accuracy	Training time per epoch	Epochs
MLP with Kaiming initialization	80%	1.6s	100
CNN with SGD optimizer	87%	5.5ms	20

Table 5: Comparison of CNN with SGD and MLP

Model	Accuracy	Training time	Epochs
MLP with Kaiming initialization	80%	3 min	100
CNN with SGD optimizer	87%	2 min	20
CNN with Adam optimizer	95%	2 min	10
ResNet50 model with 3 FC	80%	1 hr	10
Vgg16 model with 3 FC	70%	40mins	65

Table 6: Comparison of Different Models

4.2 CIFAR-10

In the first experiment, we compared the performance of CNN using the SGD optimizer with that of a MLP model on the CIFAR-10 dataset. The CNN [2] [3] model was faster and achieved higher accuracy. The results are shown in Table 2.In the second experiment, we fine-tuned the SGD optimizer with different momentum values ranging from 0.4 to 0.9 and found that the best performance was achieved with a momentum value of 0.8, resulting in an accuracy of 87%.Next, we replaced the SGD optimizer with the Adam optimizer, which led to a notable improvement in accuracy, achieving 95%. Also, Adam optimizer converged in 10 epochs, which was notable lesser than SGD optimizer. Later We modelled Adam optimizer with different momentum values, beta_1 ranging from 0.8 to 0.95 and and beta_2 ranging from 0.9 to 0.98 and achieved highest accuracy of 95% with beta_1=0.9 and beta_2=0.98. In terms of stability, both MLPs and CNNs provides stability with SGD, provided careful consideration of network architecture and regularization techniques. For CNNs, their spatial understanding capabilities make them particularly stable in image-related tasks when used with the right practices. Finally, in the last experiment, we utilized a pre-trained ResNet50 model and modified it by removing all fully connected layers and adding three fully connected layers with 128 neurons in each layer. This modified ResNet50 model achieved a remarkable accuracy of *%80, surpassing all the previous models tested..* The comparison models are shown in Table 2.

4.3 Creativity

4.3.1 Comparison of ResNet50 model with Vgg16 model

In our research, we conducted a comparative analysis of pretrained ResNet50 and Vgg16 models [6], considering training time and accuracy. Vgg16 attained a 70% accuracy level in just 40 minutes, whereas ResNet50, although taking 1 hour to train, achieved a superior accuracy of 80%. It's important to note that Vgg16 is characterized by a simpler and shallower architecture, while ResNet50 stands out as a significantly deeper network that leverages residual blocks to tackle training difficulties encountered in very deep neural networks.

4.3.2 Effect of Varying Width and Depth of the network

In our research, we investigated the impact of varying the depth and width of neural networks. [4] Our findings revealed that altering the depth of the network has a more significant influence on accuracy compared to changing the network's width. Increasing the depth of the neural network had a pronounced effect on the model's performance, underscoring the importance of depth in capturing complex patterns and representations within the data.

5 Discussion and Conclusion

Our study on Fashion MNIST and CIFAR-10 datasets explored the impact of various neural network configurations. [7] Initiating with model weight initialization methods like Xavier and Kaiming yielded better outcomes compared to zero weights. Modifying MLP architectures by adding layers and structural changes enhanced accuracy, aiding in complex data comprehension. Different activation functions, including leaky ReLU and Softplus, exhibited distinct influences on results, with leaky ReLU outperforming others. The use of L1 and L2 regularization techniques mitigated overfitting, with effectiveness contingent on specific settings.Notably, the absence of image normalization led to low testing accuracy, emphasizing the importance of this preprocessing step. In contrast, CNNs outperformed MLPs, particularly on datasets like CIFAR-10, underscoring their suitability for image data. Experimenting with optimizers, momentum in SGD, and the use of Adam optimizer had significant effects. Adam's regularization capabilities were especially valuable.Lastly, comparing our models to established ones like Resnet-50 and VGG-16 provided valuable insights into their performance.

6 Statement of Contribution

Shwetali Shimangaud compared MLP and pretrained ResNet50 with CNN on CIFAR-10. Dheeraj Vattikonda developed a custom MLP with different hyperparameters for FashionMNIST. Chaitanya Tekane performed exploratory data analysis and generated informative visuals. The team collaboratively contributed to the report.

References

[1] Leon O Chua and Tamas Roska. The cnn paradigm. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 40(3):147–156, 1993.

[2] Nie Jinliang. Cifar10 image classification based on resnet. 23(1):412–415, 2019.

[3] Neha Sharma, Vibhor Jain, and Anju Mishra. An analysis of convolutional neural networks for image classification. *Procedia computer science*, 132:377–384, 2018.

[4] Yeonjong Shin. Effects of depth, width, and initialization: A convergence analysis of layer-wise training for deep linear neural networks. *Analysis and Applications*, 20(01):73–119, 2022.

[5] Hind Taud and JF Mas. Multilayer perceptron (mlp). *Geomatic approaches for modeling land change scenarios*, pages 451–455, 2018.

[6] Dhananjay Theckedath and RR Sedamkar. Detecting affect states using vgg16, resnet50 and se-resnet50 networks. *SN Computer Science*, 1:1–7, 2020.

[7] Jia Xin, Tey Jia Yi, Voon Pei Yi, Pu Jun Yu, and Zailan Arabee Abdul Salam. Convolutional neural network for fashion images classification (fashion-mnist). *Journal of Applied Technology and Innovation (e-ISSN: 2600-7304)*, 7(4):11, 2023.

A Diagrams for CIFAR10

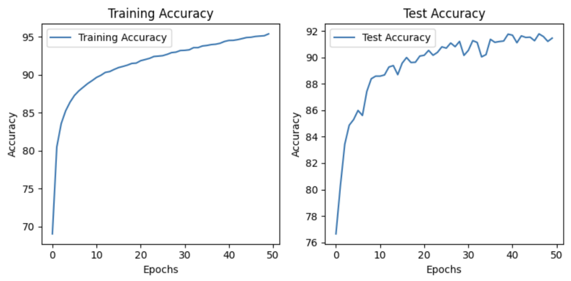

Fig.15 Training and Testing curves trained on CNN on Fashion MNIST dataset

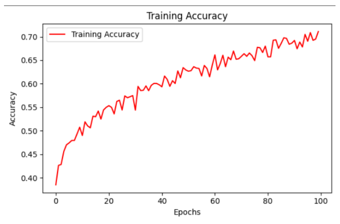

Fig.16 MLP: Accuracy

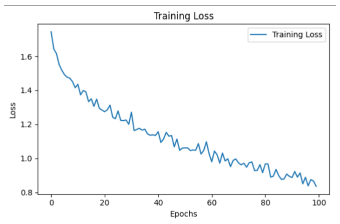

Fig.17 MLP: Loss

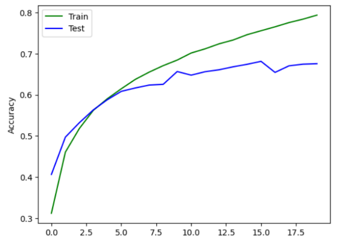

Fig.18 SGD Accuracy: momentum=0.4

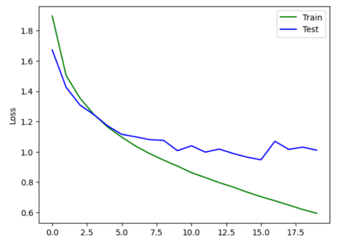

Fig.19 SGD Loss: momentum=0.4

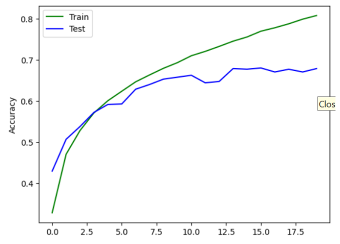

Fig.20 SGD Accuracy: momentum=0.5

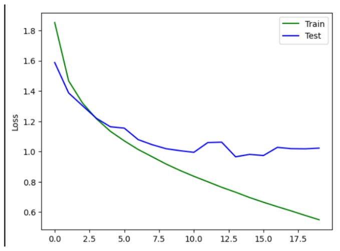

Fig.21 SGD Loss: momentum=0.5

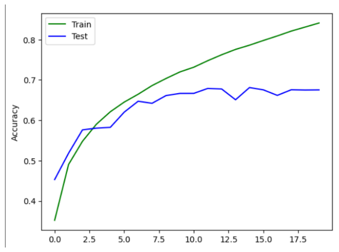

Fig.22 SGD Accuracy: momentum=0.6

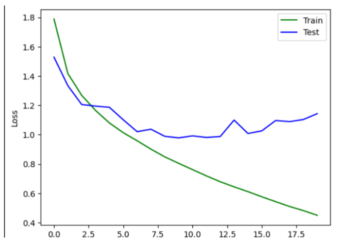

Fig.23 SGD Loss: momentum=0.6

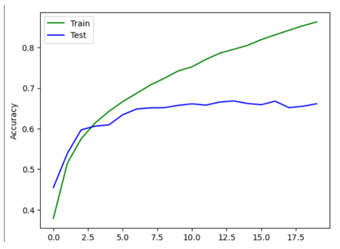

Fig.24 SGD Accuracy: momentum=0.7

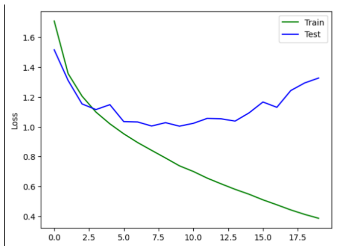

Fig.25 SGD Loss: momentum=0.7

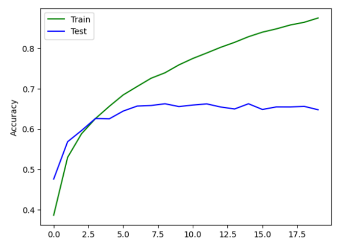

Fig.26 SGD Accuracy: momentum=0.8

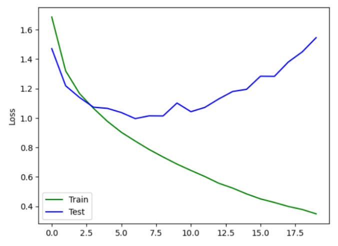

Fig.27 SGD Loss: momentum=0.8

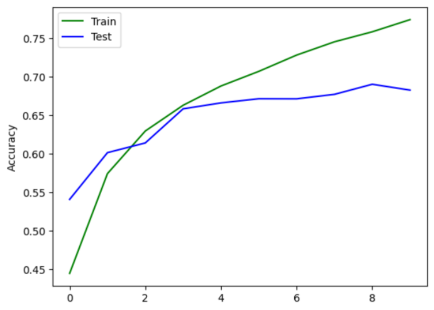

Fig.28 Accuracy: Adam optimizer

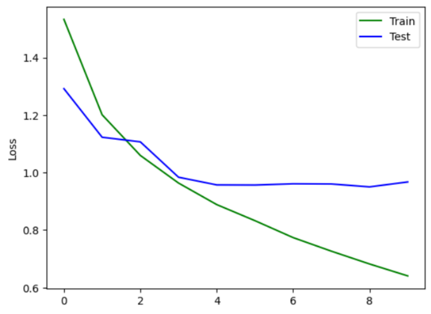

Fig.29 Loss : Adam optimizer