Homework Assignment #4: Multilayer Perceptrons (MLPs)

Vikrant Bhati

Introduction

For the assignment the objective was to analyze and implement multilayer perceptron (MLP) in different configurations for different datasets. A multilayer perceptron is a combination of multiple perceptron layers in different configuration layers such as a Input layer, hidden layers and a output layer which is capable of capturing the nonlinear relationship. And this ability aligns with the Universal Approximation Theorem, which states that with sufficiently number of MLP we can approximate any continuous function given enough neurons and appropriate activation functions.

Approach(es)

For the first task, I created two models for each dataset—CIFAR-10 and CIFAR-100—where each dataset consists of 10 and 100 classification categories, respectively. To preprocess the data, I normalized pixel values to the range [0,1], like Scikit-learn MinMaxScaler, to prevent large gradient updates. The output labels were encoded using One-Hot Encoding, where each class was represented as a binary vector with 1 indicating the correct classification.

Since both datasets contain images of size 32×32 pixels with three RGB channels (i.e., 32×32×3), I flattened the data using a Flatten layer before feeding it into the neural network. The model architecture consisted of multiple hidden layers with neurons to capture patterns and dropout layers for regularization to prevent overfitting. Given the need to model nonlinear relationships, the ReLU activation function was used in the hidden layers.

For training, I used the categorical cross-entropy loss function with Stochastic Gradient Descent (SGD) optimizer and ran the model for the prescribed number of epochs. Additionally, I applied a validation split of 0.2 to monitor model performance and detect potential overfitting using accuracy as the primary evaluation metric.

Further for the second question, I explored the impact of different hyperparameters on accuracy using the CIFAR-10 dataset. Three configurations were tested:

1. A 6-layer model (1 Flatten, 2 Dense, 2 Dropout, and 1 Output layer) using SGD and Adam optimizers, further tested with dropout rates of 0.4 and 0.6.
2. An 8-layer model (1 Flatten, 3 Dense, 3 Dropout, 1 Output layer).
3. A 10-layer model (1 Flatten, 4 Dense, 4 Dropout, 1 Output layer).

Each configuration was trained for 20 epochs, and I compared the impact of different depths and dropout rates on accuracy.

For the third question, I analyze how features are captured, and weights are updated when adding a Dense layer. To investigate this, I implemented a simple model in two scenarios:

- Without any Dense layer.
- With a Dense layer containing 512 neurons.

This experiment helped in visualizing the effect of additional trainable parameters on feature extraction and weight updates.

Experimental Results

Question1:

CIFAR10 Result: -

Model: "sequential_5"

Layer (type)	Output Shape	Param #
flatten_5 (Flatten)	(None, 3072)	0
dense_17 (Dense)	(None, 512)	1,573,376
dropout_12 (Dropout)	(None, 512)	0
dense_18 (Dense)	(None, 1024)	525,312
dropout_13 (Dropout)	(None, 1024)	0
dense_19 (Dense)	(None, 512)	524,800
dropout_14 (Dropout)	(None, 512)	0
dense_20 (Dense)	(None, 10)	5,130

Total params: 2,628,618 (10.03 MB)
Trainable params: 2,628,618 (10.03 MB)
Non-trainable params: 0 (0.00 B)

Fig: Model for CIFAR10 Dataset

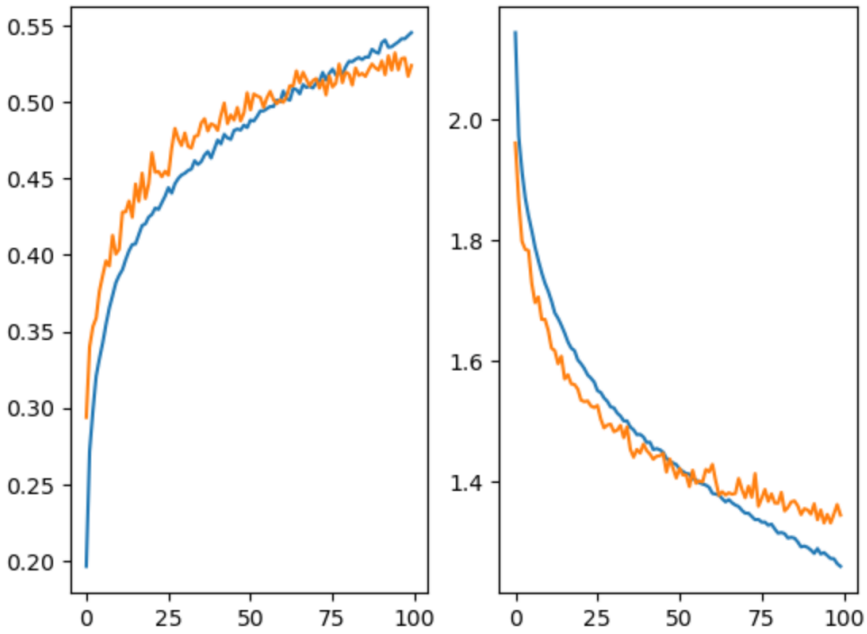

Fig a) Accuracy for validation and test set *Fig b) Loss for validation and test set*

Accuracy for CIFAR10: 0.5231999754905701

CIFAR100 Result: -

Model: "sequential_10"

Layer (type)	Output Shape	Param #
flatten_10 (Flatten)	(None, 3072)	0
dense_37 (Dense)	(None, 1024)	3,146,752
dropout_27 (Dropout)	(None, 1024)	0
dense_38 (Dense)	(None, 2048)	2,099,200
dropout_28 (Dropout)	(None, 2048)	0
dense_39 (Dense)	(None, 1024)	2,098,176
dropout_29 (Dropout)	(None, 1024)	0
dense_40 (Dense)	(None, 100)	102,500

Total params: 7,446,628 (28.41 MB)
Trainable params: 7,446,628 (28.41 MB)
Non-trainable params: 0 (0.00 B)

Fig: Model for CIFAR10 Dataset

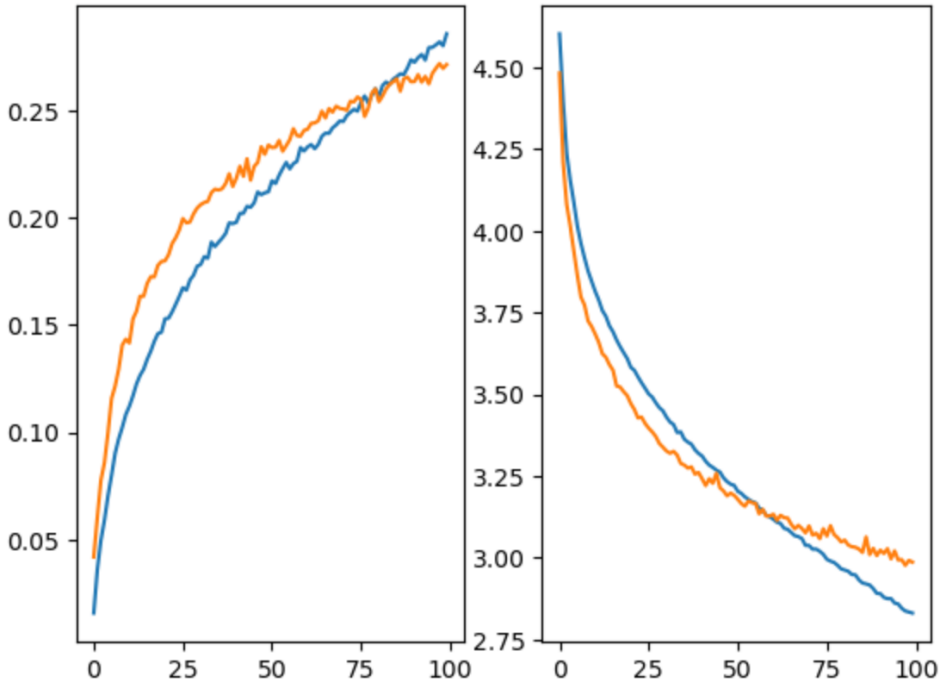

Fig a) Accuracy for validation and test set *Fig b) Loss for validation and test set*

Accuracy for CIFAR100: 0.27799999713897705

Results for Question 2:

Number of Layers	Optimizer	Epochs	Weights	Dropout rate	Trainable Parameters	Test Accuracy
8	SGD	20	[1024,512]	0.4	3,676,682	0.49619999527931213
8	Adam	20	[1024,512]	0.4	3,676,682	0.35179999470710754
8	SGD	20	[1024,512]	0.6	3,676,682	0.47040000557899475
8	Adam	20	[1024,512]	0.6	3,676,682	0.20479999482631683
10	SGD	20	[512,1024,512]	0.4	2,628,618	0.4722999930381775
10	Adam	20	[512,1024,512]	0.4	2,628,618	0.24580000340938568
10	SGD	20	[512,1024,512]	0.6	2,628,618	0.4032999873161316
10	Adam	20	[512,1024,512]	0.6	2,628,618	0.17990000545978546

12	SGD	20	[512,1024, 1024, 512]	0.4	3,678,218	0.4756999909877777
12	Adam	20	[512,1024, 1024, 512]	0.4	3,678,218	0.1972000002861023
12	SGD	20	[512,1024, 1024, 512]	0.6	3,678,218	0.2662000060081482
12	Adam	20	[512,1024, 1024, 512]	0.6	3,678,218	0.10000000149011612

Results for Question 3:

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 3072)	0
dense (Dense)	(None, 10)	30,730

Total params: 30,730 (120.04 KB)
Trainable params: 30,730 (120.04 KB)
Non-trainable params: 0 (0.00 B)

Fig: Model for CIFAR10 Dataset without Hidden layer

Model: "sequential_1"

Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 3072)	0
dense_1 (Dense)	(None, 512)	1,573,376
dense_2 (Dense)	(None, 10)	5,130

Total params: 1,578,506 (6.02 MB)
Trainable params: 1,578,506 (6.02 MB)
Non-trainable params: 0 (0.00 B)

Fig: Model for CIFAR10 Dataset with Hidden layer of 512 Neurons

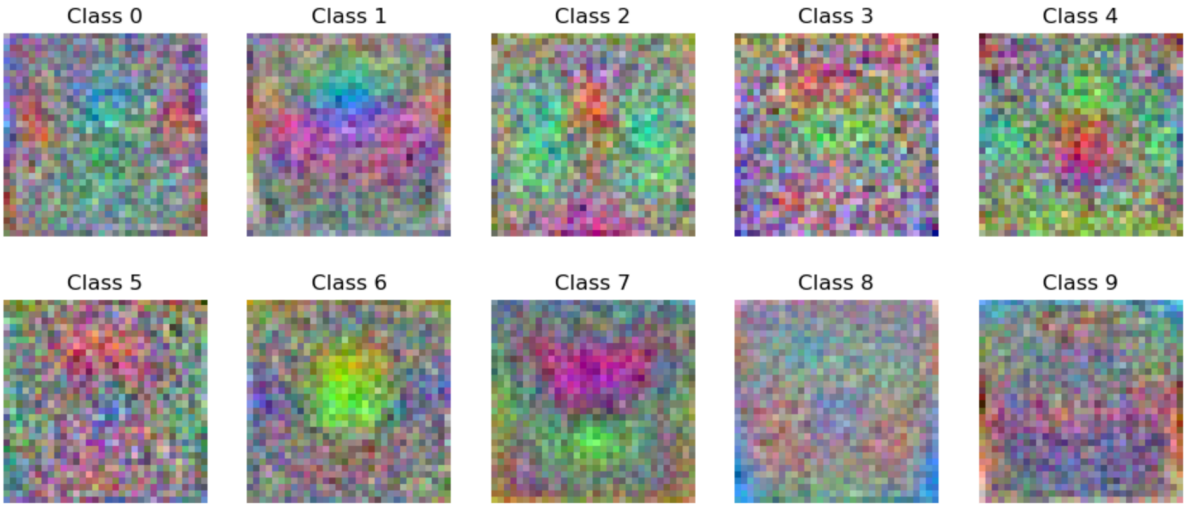

Fig: Output of Model for CIFAR10 Dataset without Hidden layer

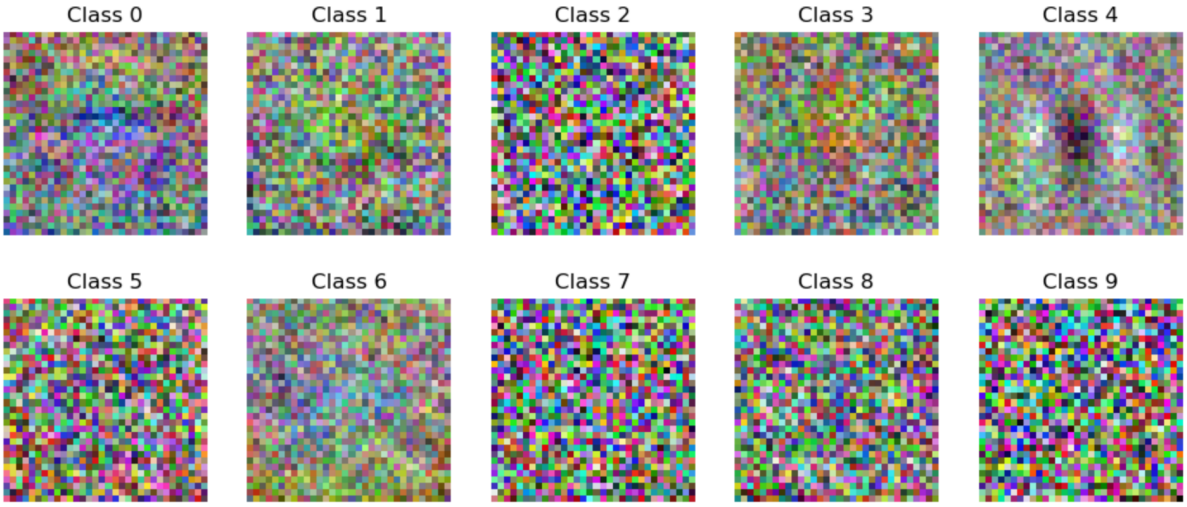

Fig: Output of Model for CIFAR10 Dataset with Hidden layer

Discussion

For question 1, the results indicated that my MLP model performed better on CIFAR-10 than on CIFAR-100. This was expected because CIFAR datasets contain image data, and flattening the images removes spatial information such as edges, curves, and textures, which are crucial for classification. Despite this, the MLP model achieved approximately 52% accuracy on CIFAR-10 and around 27% accuracy on CIFAR-100, demonstrating its ability to capture some level of non-linearity in the data. The lower accuracy for CIFAR-100 is attributed to the increased number of classes (100 vs. 10), making classification more challenging without spatial feature extraction.

For question 2, the results showcased that SGD outperformed the Adam optimizer. This can be explained because of SGD's ability to prevent overfitting and its effectiveness on larger datasets. Adam, while typically faster in convergence, many times leads to suboptimal generalization in deep networks. For dropout, the accuracy decreased as the dropout rate increased, which aligns with expectations. Higher dropout rates remove more neurons during training, which increases bias while reducing the variance, thereby acting like a regularization mechanism. Additionally, as the number of layers increased, the model exhibited a higher tendency of overfitting, leading to a decrease in overall performance. This suggests that deeper architectures require more advanced techniques, such as convolutional layers in CNNs.

For Question 3, analyzing the model's weights before and after adding a Dense layer of 512 neurons revealed that when the image was flattened and weights were examined, the model likely captured only low-level features like edges but failed to retain higher-order relationships. However, after adding a Dense layer with 512 neurons, the network was able to better capture the non-linear patterns, as evidenced by greater variation in weight distributions and fluctuations in RGB values. This suggests that increasing the number of neurons allowed the network to learn more abstract representations beyond simple edge detection.