COMP 551
Project write-up for
Mini-Project 2
Multilayered Perceptrons and Convolutional Neural Networks

Yajiv Luckheenarain and Anna Decterov

Multilayer perceptrons (MLPs) and Convolutional Neural Networks (CNNs) are two popular machine learning models used for image classification tasks. To enhance the performance of these models, we preprocessed the images using PCA whitening, which involves decorrelating and normalizing the pixel values to reduce the redundancy and enhance the signal-to-noise ratio in the data. Both MLPs and CNNs were trained on the preprocessed CIFAR-10 dataset, which consists of 60,000 32x32 color images in 10 different classes. The MLP models achieved an accuracy of approximately 30%, while the CNN achieved an accuracy of approximately 65% in pytorch and 30% in tensorflow. However, with a softmax function used at the top layer, the tensorflow model reached an accuracy of 61%. This indicates that the CNN model performed significantly better in classifying the images. Overall, CNNs have been shown to outperform MLPs in image classification tasks. If a model with pre-trained weight in the convolutional layers is used, an equivalent accuracy can be achieved. However, it takes very long to train.

I. Introduction

MACHINE learning has revolutionized the field of computer vision, enabling computers to classify, recognize, and even generate images. Two of the most popular machine learning models for image classification tasks are multilayer perceptrons (MLPs) and convolutional neural networks (CNNs).

An MLP is a densely connected artificial neural network with one or more hidden layers that learns complex non-linear relationships between inputs and outputs. MLPs are feedforward networks where information flows from the input layer through the hidden layers to the output layer.

CNNs are a type of neural network designed for image recognition, inspired by the visual cortex in animals. CNNs have convolutional, pooling, and fully connected layers that enable the network to learn hierarchical features of the input image, from simple edges and shapes to complex objects.

We compare the performance of MLPs and CNNs on the CIFAR-10 dataset after applying various preprocessing techniques and experimenting with different MLP architectures. Our goal is to gain a deeper understanding of the strengths and limitations of these two powerful machine-learning models for image classification.

To improve our machine learning models' performance on CIFAR-10, we flattened the images to 1x3072 and preprocessed them using mean normalization, PCA, and whitening. These techniques involved subtracting the mean value of each pixel, reducing data dimensionality while retaining variance, and reducing correlations between features, respectively.

These pre-processing steps helped us reduce the redundancy and noise in the data, which in turn improved the accuracy and speed of our machine-learning models. In addition, they helped prevent overfitting, which can occur when a model becomes too complex and starts fitting the noise in the training data rather than the underlying patterns.

We experimented with various architectures and regularization techniques to train our models on the preprocessed CIFAR-10 dataset. Specifically, we used our MLP class to train models with different numbers of hidden layers and activation functions, and we also experimented with L1 and L2 regularization. Furthermore, we trained models without any normalization of the images before training. The output layer activation function for our MLP models was softmax, as this was a multi-class label classification task. We used stochastic gradient descent with a mini-batch size of 8, a learning rate of 0.01, and 20 epochs to train our MLP models.

We compared our MLP models with the implementation of CNNs using the pytorch and tensorflow machine learning libraries. By experimenting with the number of layers, kernel sizes and activation functions, we were able to achieve a much higher accuracy than that of the MLP.

Overall, our combination of preprocessing techniques, MLP architectures, and regularization techniques enabled us to achieve high accuracy on the challenging CIFAR-10 image classification task.

II. Datasets

Before performing any preprocessing on the CIFAR-10 dataset, we first verified the dimensions of the data by printing the shape of the data. The training set contains 50,000 images of size 32x32x3, while the test set contains 10,000 images of the same size. We also printed out some images from the CIFAR-10 dataset to get an idea of the type of data we were working with. The images are of various objects such as airplanes, cars, and animals, and are presented in RGB colour. By visualizing the images, we can better understand the complexity of the dataset and the challenges that lie ahead in accurately classifying them. The CIFAR-10 dataset is widely used for benchmarking computer vision and machine learning algorithms due to its large size, diverse range of objects, and the challenges it poses in image classification tasks.

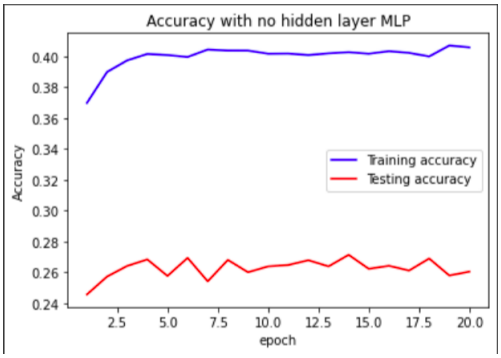

Fig. 1. Experiment 1.1: Accuracy as a function of epoch for the trained MLP with no hidden layers

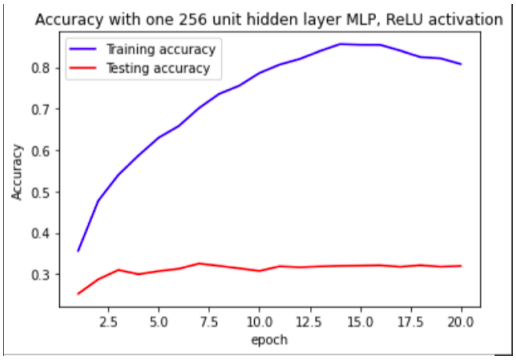

Fig. 2. Experiment 1.2:Accuracy as a function of epoch for the trained MLP with 1 hidden layer of 256 units and ReLU Activation

III. Results

A. Experiment 1

We trained three different MLP models on the CIFAR-10 dataset, we compared their test accuracies to see how non-linearity and network depth affected their performance. The model with no hidden layer performed the worst, followed by the model with a single hidden layer with 256 units and ReLU activations. The third model with 2 hidden layers with 256 units and ReLU activations consistently outperformed the other two models, achieving the highest accuracy on the test set. Our results showed that the accuracy of each model fluctuated by epoch.

These results suggest that adding depth to the network can improve its accuracy, and that adding a single hidden layer may not provide sufficient non-linearity to capture the underlying patterns in the data. These results are as expected. The MLP with no hidden layer approximately capped at 40% accuracy on the training set and 27% accuracy on the testing set. The MLP with one hidden layer had approximately respectively 80% and 30% accuracies on its training and testing set. The MLP with two hidden layers eventually started overfitting to the training set with accuracies of about 90% while peaking at a 35% accuracy on the testing set.

B. Experiment 2

We also conducted an additional experiment to compare the performance of different activation functions in MLPs.

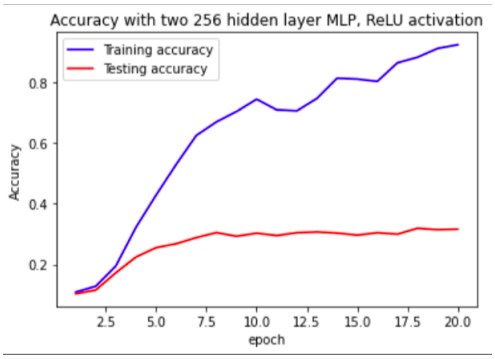

Fig. 3. Experiment 1.3 :Accuracy as a function of epoch for the trained MLP with 2 hidden layer of 256 units and ReLU Activation

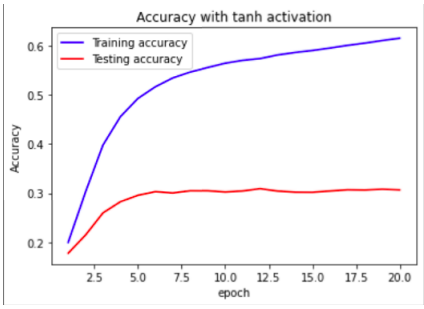

Fig. 4. Experiment 2.1: Accuracy as a function of epoch for the trained MLP with 2 hidden layer of 256 units and tanh Activation

We took the MLP model with 2 hidden layers and ReLU activations from Experiment 1 and created two additional models with the same architecture, but with their activation functions replaced by tanh and Leaky-ReLU. During training, we monitored the accuracy of the models on both the training and testing sets. We found that the tanh activation (Experiment 2.1) MLP performed at about 31% accuracy, which was lower than the ReLU MLP with hidden layers. The Leaky ReLU MLP (Experiment 2.2) peaked towards 32%. Our results suggest that ReLU and Leaky-ReLU are adequate activation function for MLPS trained for image recognition tasks on the CIFAR-10 dataset.

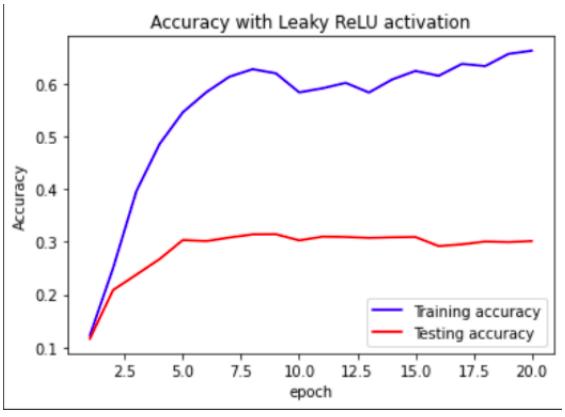

Fig. 5. Experiment 2.2 :Accuracy as a function of epoch for the trained MLP with 2 hidden layer of 256 units and Leaky-ReLU Activation

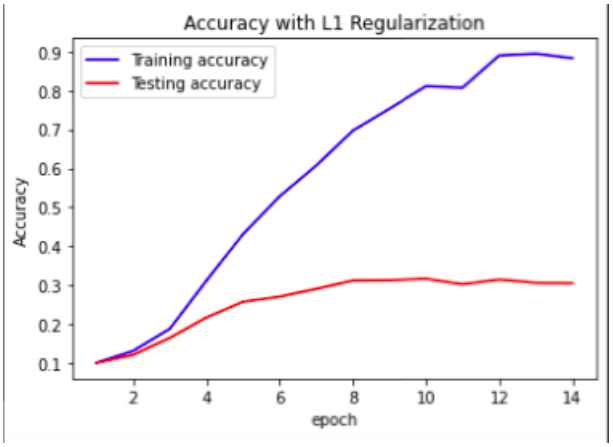

Fig. 6. Experiment 3.1 :Accuracy as a function of epoch for the trained MLP with L1 regularization

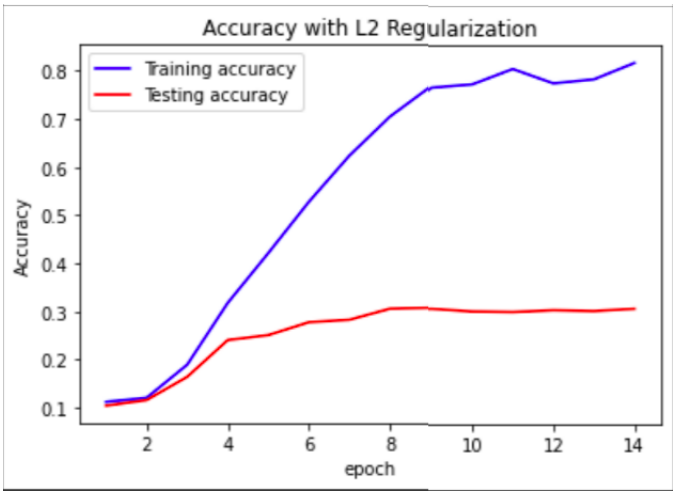

Fig. 7. Experiment 3.2 :Accuracy as a function of epoch for the trained MLP with L2 regularization

C. Experiment 3

Hyperparameter tuning was done to determine good values for L1 and L2 penalties on the training of our MLP models. These values were respectively found to be optimized at 0.001 and 1e-05. We created an MLP with 2 hidden layers, each having 256 units with ReLU activations. However, this time, we added L1 and L2 regularization to the network with the values that were previously found and trained the MLP in this way. Our goal was to investigate how these regularizations affect the accuracy of the model. The results of the experiment showed that adding regularization to the network reduced overfitting and improved the accuracy on the test set. Moreover, we found that L2 regularization generally outperformed L1 regularization in terms of test accuracy. Peak accuracies on each set for L1 and L2 regularization are referred in Table 1, respectively as Experiment 3.1 and 3.2.

D. Experiment 4

We trained an MLP with 2 hidden layers, each having 256 units with ReLU activations on the CIFAR-10 dataset.

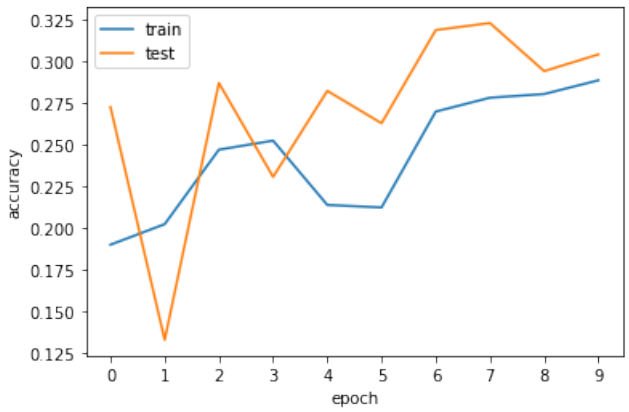

Fig. 8. Experiment 3.5.1 :Accuracy as a function of epoch for the trained tensorflow CNN with top layer activated by ReLu

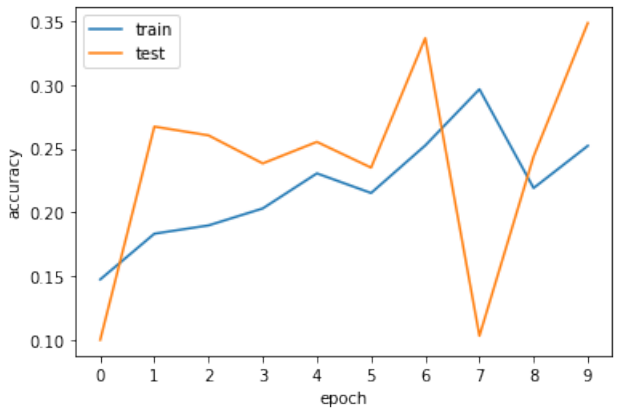

Fig. 9. Experiment 3.5.2 :Accuracy as a function of epoch for the trained tensorflow CNN with top layer activated by softmax

However, this time, we trained the network with unnormalized images. Compared to the same MLP trained in experiment 1, this MLP performed horribly. Its accuracy varies unpredictably without any convergence towards higher accuracy distinguishable.

E. Experiment 5

In this experiment we implemented different CNNs using python libraries. We first created a CNN using tensorflow. However, we noticed that the model didn't respond well to the pre-trained images. This may be due to added noise produced by the whitening of the images. However, the model worked fine with the original images so that is what we used for the rest of the models created in this experiment. At first, we were not able to achieve satisfactory accuracies with the tensorflow model when we were activating all layers with ReLu. In fact, it was peaking at around 30%. The accuracy graphs for some models were also irregular and not showing a steady increase in accuracy with the number of epochs.

However, after applying a softmax activation function to the top layer of the network, we were able to achieve an accuracy of around 60 We then created an additional CNN using pytorch to compare those two models. This CNN reported accuracy of

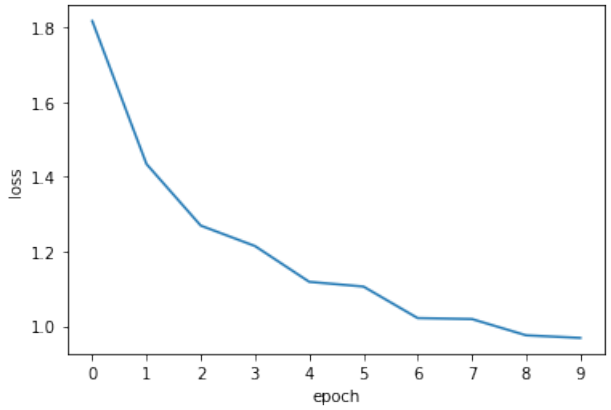

Fig. 10. Experiment 3.5.3 : Loss function for the Pytorch CNN Model

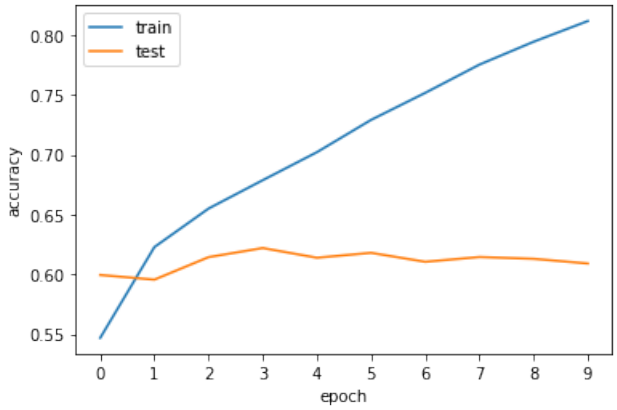

Fig. 11. Experiment 3.6 :Accuracy as a function of epoch for the optimal CNN using ResNet50 pretrained weights

65%, even with a RELU activation at the top layer. It also required less training time than its tensorflow equivalent. It was able to finish training in 6 minutes while its tensorflow equivalent required a training time of up to 15 minutes. However, it was a lot more difficult to work with.

F. Experiment 6

In this experiment, a Resnet50 model was loaded using the keras tensorflow library. It is a convolutional neural network that is 50 layers deep. It is considered as one of the most efficient neural network architectures. The convolutional layers were frozen and fully connected layers were added. The weights of the fully-connected layers were then trained. The pre-processed images did not work well with this model, yielding an accuracy of only 40%. As a result, the original images of the dataset were used. Three different models were created, but they all yielded similar results of an accuracy peaking at 62%. We tried to vary different parameters with every run, but every time we got equivalent accuracies of around 60%. As a result, in our case, the MLP containing only 2 fully-connected layers was the most efficient since it only took 50 min to run.

Contrary to our expectations, our model yielded a worse accuracy than our most performant CNN model, but it still performs much better than our best MLP. A possibility of what could be done to give this model and also other models a higher accuracy would be to upsample the images since they are relatively small. In terms of training time, it was much longer, having a peak running time of 1 hour. This is very long compared to the peak running time of 15 min for the CNN, and even the peak running time for the MLP which was 40 minutes.

G. Extra

As an extra experiment, we trained MLPs with three hidden layers and observed their accuracies as a function of epoch. However, we observed that these MLPs were less accurate than our previous model. This was likely due to not allowing enough epochs for training, which impacted the convergence of the SGD to a local minimum. The algorithm did not have sufficient time to explore the vast parameter space and reach an optimal solution. Additionally, the fully connected nature of the models resulted in a significant increase in the number of weights to be trained with only one additional layer. Consequently, the optimization problem became more complex, and the algorithm required more time to search for the optimal values of the weights. Therefore, it is crucial to consider the number of epochs required to ensure that the model converges to an optimal solution, particularly for more complex models such as those with multiple hidden layers.

H. MLP Accuracies

Table 1: MLP Peak Accuracy (%)

	Training Set	Testing Set
Experiment 1.1	40.7	27.1
Experiment 1.2	85.6	32.5
Experiment 1.3	92.3	31.9
Experiment 2.1	61.5	30.9
Experiment 2.2	66.3	31.5
Experiment 3.1	89.4	31.7
Experiment 3.2	81.9	30.8
Experiment 4	10	10
Experiment 5 Pytorch	68	65
Experiment 5 Keras with ReLu	31	37
Experiment 5 Keras with Softmax	85	62
Experiment 6	85	62
Extra	28.9	22.1

IV. Discussion

Our experiments demonstrate that CNNs are better suited for image recognition tasks on the CIFAR-10 dataset than MLPs. However, there are several potential next steps that could be taken to further this experiment.

While we used several preprocessing techniques on the data, there are other preprocessing techniques that could be

explored. For example, data augmentation could be applied to increase the size of the training set and improve the generalization of the models.

We only trained the models using a single dataset. It would be interesting to investigate the performance of MLPs and CNNs on other datasets and tasks to determine if our findings generalize to other domains.

We explored the MLP's accuracy as a function of the L1 and L2 penalties, which is a critical hyperparameter in the training process. However, other hyperparameters such as batch size, learning rate, number of hidden layers, units per layer, and activation functions could be tuned to further improve the performance of the models.

Finally, since the pictures used are so small, a good idea would be to upsample the images. This would provide more range for the convolution kernels and create more parameters to be used by the CNN.

V. Conclusion

In conclusion, our findings suggest that CNNs are a promising choice for image recognition tasks on the CIFAR-10 dataset, but further research is needed to fully explore their potential and optimize their performance. Future experiments could focus on the optimization of these hyperparameters and the exploration of different datasets and tasks.

VI. Statement of Contributions

We were two members for this project as the third team member is auditing this course. Yajiv wrote the MLP class and trained the MLP models. Anna implemented the CNN models and trained them through varying parameters. We both computed and plotted the model's accuracies.

References

[1] Backpropagation from scratch with Python, *Pyimagesearch*, https://pyimagesearch.com/2021/05/06/backpropagation-from-scratch-with-python/

[2] CS231n Convolutional Neural Networks for Visual Recognition, *CS231n: Deep Learning for Computer Vision*, Stanford, https://cs231n.github.io/neural-networks-2/datapre