Mini Project 2: Classification of Image Data with Multilayer Perceptrons and Convolutional Neural Networks

Ethan Kreuzer 261050944 Yann Bourdé 260838140 Marie Ezra Marin 261053813

February 16, 2023

Abstract

In this project we implemented a functional multilayer perceptron (MLP) from scratch and observed its performance, with varying hyper-parameters, on image classification of the CIFAR-10 dataset. Additionally, we implemented a convolutional neural network (CNN) with similar dimensions and compared the performance of both models. Our best MLP implementation reached around 55% accuracy whilst the best CNN reached 68% accuracy. After running multiple experiments, we observed that a deeper and wider MLP results in better prediction accuracy, but at the cost of computational expense and longer training time. Furthermore, we observed that the choice of activation function is critical and ReLU performs the best. Additionally, the implementation of L1 and L2 regularization did not improve our results.

Introduction

As highlighted in the abstract the major task for this project is the implementation of the MLP class. The other tasks include implementing the CNN model and formatting the data appropriately. For the MLP implementation, there are numerous parameters to consider such as batch size and learning rate. For those two we decided to go with popular picks: 16 and 0.01 respectively. For the number of epochs, after running some tests and looking at the accuracy/loss values, we settled on 10 as the metrics were not evolving much after that point. For the performance of the MLP, we found that 2 hidden layers was always more or equally as accurate as 1 and 0 hidden layers. This was to be expected, however it was interesting to see that the gap between 1 hidden layer and 2 hidden layer was less significant than between 0 and 1 hidden layer. Additionally, in our extra experiments, we observed that 2 hidden layers of 1024 units achieved the greatest accuracy for MLP with a 2% increase compared with 256 unit layers. Furthermore, we used a CNN and ResNet50 models. We found the CNN to have the best performance of all the model we tested, achieving an accuracy of 68% on Test set. The pre-trained ResNet50 model we used had its best performance with 4 Convolutional Layers added on top of the pre-trained layers. The ReSnet model took the longest to train and still did not outperform the CNN model.

Datasets

For this project, we used two datasets: CIFAR-10 and . CIFAR-10 is a collection of 60 000 images (with labels) representing 32x32 color images of 10 different classes. There are 6000 images of each class which represents an equal amount for each class. The dataset is formatted in 6 batches of 10 000 images, 5 for training and 1 for testing. When downloaded, the images are randomly distributed amongst all 6 batches. Since images are in color (RGB) and are 32x32 pixels in dimension, there are 3072 input features to the MLP. Note that for this project, it was important to zero-center and scale the image data to [-1, 1] to help the models to converge.

Results

Experiment 1

Test accuracy:

- Accuracy, 0 hidden layers, M=256, activation ReLU: 39.3%
- Accuracy, 1 hidden layers, M=256, activation ReLU: 53.2%

- Accuracy, 2 hidden layers, M=256, activation ReLU: 53.1%

We observe that there is a sharp increase in the accuracy by adding 1 hidden layer to the network. However, the accuracy stays similar when adding a second layer with the current parameters.

Experiment 2

Test accuracy:

- Accuracy, 2 hidden layers, M=256, Leaky-ReLU: 53.9%
- Accuracy, 2 hidden layers, M=256, tanh: 39.7%

We observe that Leaky-ReLU performs very similarly to the 2 layer ReLU from experiment 1. However, the tanh model falls behind with an accuracy similar to the 0 layer ReLU model. Moreover, the tanh model took almost 30 minutes to compute compared with the 8 minutes of the normal ReLU and the Leaky-ReLU (for 10 epochs). Clearly, the tanh model is less desirable because it is slower and less accurate than both ReLU and Leaky-ReLU. These last two models have similar values, so both seems like a reasonable choice for this particular setup.

Experiment 3

Test accuracy:

- Accuracy, 2 hidden layers, M=256, ReLU, L1=0.01: 41.6%
- Accuracy, 2 hidden layers, M=256, ReLU, L2=0.01: 50.1%

We observe that the current implementation of L1 and L2 regularization has no real positive effect on the accuracy values; in fact it is the opposite. L1 regularization decreases the accuracy by approximately 13% and L2 by 3%. This indicates that the model is not overfitting and that regularization is not necessary in this case. A more optimal tuning of L1 and L2 values could potentially improve accuracy, but as it stands it does not seem necessary for this particular model and dataset.

Experiment 4

Test accuracy:

- Accuracy, 2 hidden layers, M256, ReLU, unnormalized data: 10%

We observe that when using unnormalized data, the accuracy drops significantly to 10% which is no better than randomly guessing images since there are 10 classes with even amounts of images. It is suspected that our model does not deal with unnormalized images correctly since this result is not expected. It could be due to the activation function and/or softmax implementation. In any case, we would expect a lower accuracy value, just not this low.

Experiment 5

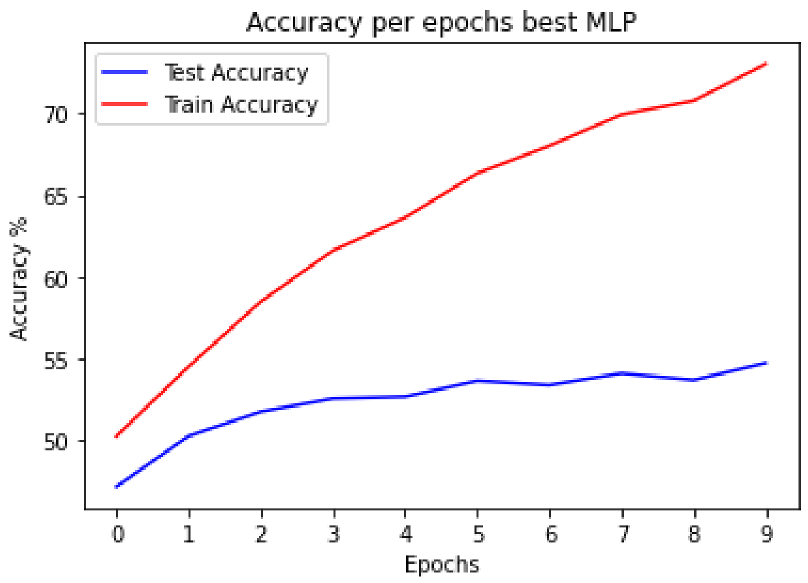

Figure 1: Best MLP train and test accuracy per epoch.

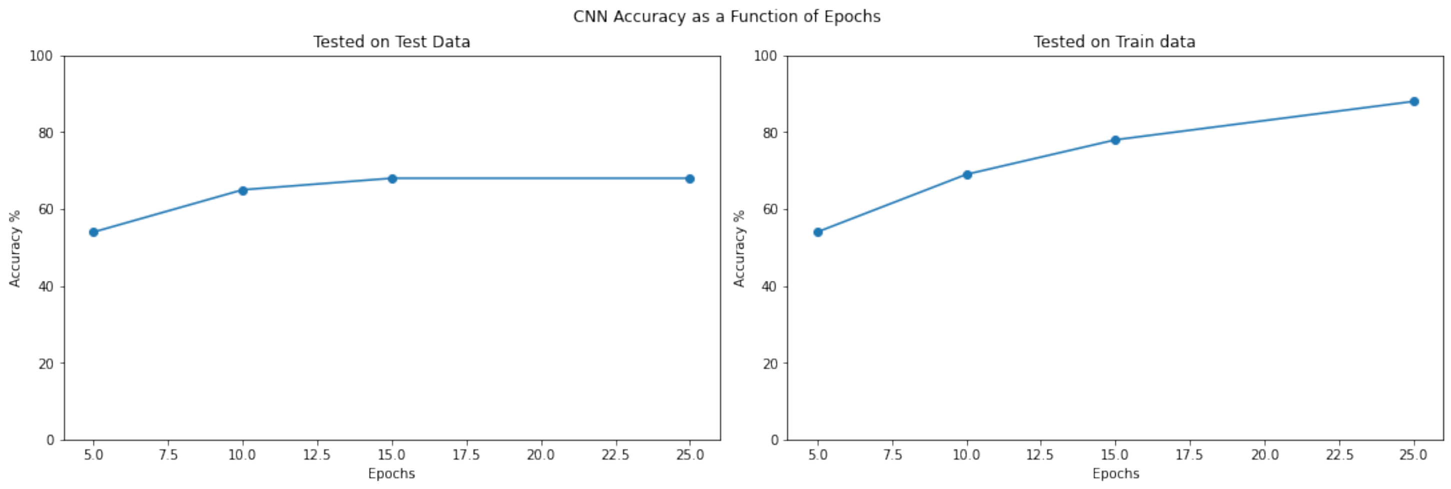

Figure 2: CNN Accuracy as a function of epochs.

The CNN model's Test accuracy performance was better than that of the previous MLP models. The CNN's worst Test accuracy score, 54%, was just as good as the MLP's best accuracy score. We observed that when increasing the epochs, the performance of the CNN on Test achieved 68% accuracy, which was considerably higher than the MLP performance. It is noteworthy, however, that CNN performance did not improve past 15 epochs, whereas the performance on Train continued to improve, reaching as high as 88% accuracy. This is indicative that the model began to overfit past 15 epochs.

Experiment 6

The pretrained model we used was the ResNet50 model. We used 256 neurons when adding fully connected layers to the model, to be able to compare with the previous CNN model. We observed that the model did improve as we added layers, the maximal accuracy on Test being achieved when 4 fully connected layers were added to the model. The models performance drastically declined beyond 4 fully connected layers, for both Test and Train accuracies. Once we had determined 4 fully connected layers was the appropriate amount, we plotted this model's performance across different epochs. The performance beyond 10 epochs was worse for both test cases, which was even earlier than the CNN model, which had its peak performance at 15 epochs. The performance of this model was better than our MLP model, but it was not better than the CNN's performance. Furthermore, it took longer to train than the CNN model.

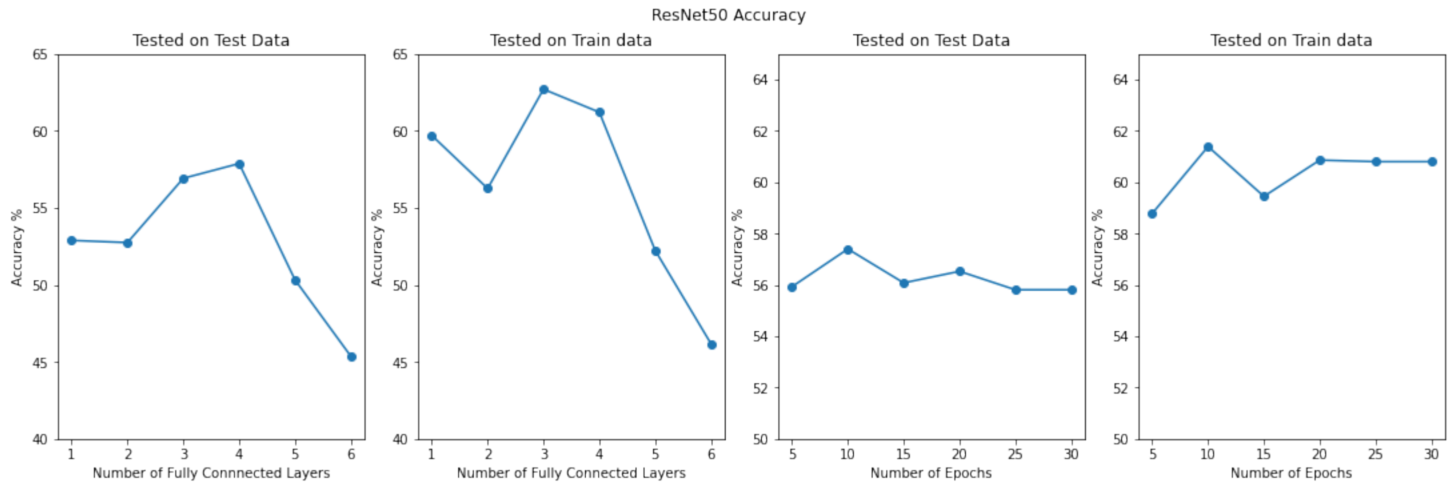

Figure 3: ResNet50 Accuracy as a function of connected layers and epochs.

Experiment 7

Please take a look at the results highlighted in the above sections.

Extra experiment: Different hidden layer width performance for MLP

Test accuracy:

- Accuracy, hidden 2 layers, M=32, ReLU: 48.6%
- Accuracy, hidden 2 layers, M=64, ReLU: 50.6%
- Accuracy, hidden 2 layers, M=128, ReLU: 52.8%
- Accuracy, hidden 2 layers, M=256, ReLU: 53.1%
- Accuracy, hidden 2 layers, M=512, ReLU: 53.9%
- Accuracy, hidden 2 layers, M=1024, ReLU: 55.6%

We can observe that as we increase the width of the model by increasing the size of the 2 hidden layers, the accuracy (on test data) increases steadily. However, it is important to note that that larger layers take a lot more time to compute. In this case, the M=1024 model took approximately 1 hour to train for 10 epochs with batch size 16, which is about twice as much time as all the other models combined!

Extra experiment: Using increasing training data for MLP

Test accuracy for our best MLP:

- Accuracy, hidden 2 layers, M=32, ReLU: 48.6%
- Accuracy 2 layers, M=256, ReLU, 10k: 45.5%
- Accuracy 2 layers, M=256, ReLU, 20k: 49.3%
- Accuracy 2 layers, M=256, ReLU, 30k: 51.2%
- Accuracy 2 layers, M=256, ReLU, 40k: 52.1%
- Accuracy 2 layers, M=256, ReLU, 40k: 53.1%

We observe that there is a clear increase in accuracy with increasing training data for the MLP.

Extra experiment: Exploring Convolutional Layer paramaters

We were interested how the outputs of the Convolutional Layers would affect performance. The inputs to the CNN model have dimension [32,3,32,32]: 32 batch size, 3 RGB channels and the images are 32x32. We wanted to see how the dimension of the output of the final Convolutional layer would improve accuracy. The following are the accuracies for the varying dimensioned outputs of the last Convolutional Layer.
Test and train accuracy for CNN with different layer size:

- 32,4,5,5: Test=59%, Train=63%
- 32,8,5,5: Test=63%, Train=69%
- 32,16,5,5: Test=67%, Train=73%
- 32,32,5,5: Test=66%, Train=74%

Discussion and Conclusion

In this project we explored the effect of width and depth of MLPs on their performance value with regards to image classification of the CIFAR-10 dataset. The first takeaway is that deeper networks result in more accurate predictions overall (at least in our case). Furthermore, using larger hidden layers does also improve accuracy but at the cost of computational expensiveness and longer training time. Additionally, increasing training data quantity does improve accuracy which explains why so many current neural network solutions strive to obtain more training data.

For the CNN, we observed that increasing the dimension of the output of the last Convolutional Layer, and thus the size of the input of the first Fully Connected Layer, increased the models performance. We also observed that the CNN, was the best model for classification, when comapred to ResNet50 and the MLP

By comparing the MLP results with different hyper parameter choices, we observed that there are plenty of ways to improve the performance of an MLP without changing its depth. We mostly observed that the choice of activation function is critical. Overall, ReLU performed better than the others in accuracy but also in speed, which is very valuable for these types of networks which tend to train for a very long time.

Finally, by comparing our best MLP model with a similar CNN model, we observed that MLPs are not particularly suited for image classification and that on average CNNs perform better in that domain although they are more computationally expensive.

Statement of Contribution

Yann: Dataset preparation, MLP implementation and tuning, tests experiments and report.
Marie: MLP implementation and extra test experiments and report.
Ethan: CNN implementation, test experiments between CNN and MLP and report.

References