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

Leo Chen 260984301 Dijian Guo 260433101 Amani Jammoul 260381641

March 9, 2023

**Abstract**

This paper explores the performance of various MLP and CNN models on image classification tasks. We evaluated effects of different activation functions, different width of hidden layers, different model depth, L1 and L2 regularization with different lambda values and the difference between using a normalized data set and an unnormalized data set. Unexpected results included higher accuracy with less MLP layers and a tanh activation function performing better than ReLU. Our results showed that that L2 regularization performed better than L1 and that there is no significant difference between using a normalized data set and an unnormalized data set. Finally, we concluded that CNN models consistently performed better than MLP models when it came to image classification tasks.

## 1 Introduction

This experiment aims to compare the performance of MLP and CNN models and their variance on image classification tasks. The data set contained 60 000 images with the 32x32 format where each image has 3 channels of different colours. We implemented an MLP model that uses mini-batch stochastic gradient descent. It uses reverse mode, including backpropagation, to get the gradients and update weights. Our weights are initialized randomly over a normal distribution. In our experiments, we considered different activation functions, such as ReLU, tanh and leaky-relu, and examined the effect of varying the width of the hidden layers, verified the effect of L1 and L2 regularization with different lambda value and investigated the difference between using a normalized dataset and an unnormalized dataset. Our experiments with various hidden layers show that a smaller number of layer has a higher accuracy, which is not what we expected. Experiments on activation functions show that tanh was the best choice, which again was unexpected as we presumed either relu or leaky-relu would be the best. Our experiments on L1 and L2 regularization showed that L2 performs better than L1 and that there is no significant difference between using a normalized data set and an unnormalized data set. Our main results showed that CNN models consistently outperform MLP models when it comes to image classification tasks. We also found that adding more layers to the CNN model can improve its performance but it significantly slows down learning.

## 2 Datasets

All of our experiments are run on the CIFAR-10 image dataset [1], which contains 60000 images of 10 different modes of transportation and animals. There is no overlap between them, meaning each image falls under exactly one class. We used PyTorch to load the data, which returned a normalized set of training (50000 images) and test (10000 images) data. This train/test split was used throughout our experiments. After loading, we counted the number of images that fall under each class. We found that there is an equal amount for each, i.e. there are exactly 5000 images for each of the 10 classes in the training set and 1000 images for each in the test set.

## 3 Results

### 3.1 Hyperparameter Tuning

In order to determine which values to set for our hyperparameters, we used skit-learn's RandomizedSearchCV model section tool. The parameters included the learning rate, batch size, and number of maximum epochs for the mini batch stochastic gradient descent. The parameter space we ran this search on was: learning rate = [0.0001, 0.001, 0.01, 0.1], batch size = [16, 32, 64, 128], number of epochs = [10, 50, 100, 1000]. After implementing an

MLP_Estimator class, which was necessary to have in order to use this tool, the search used random combinations and used 5 folds for each of the 10 candidates, resulting in 50 fits in total. The score was based on accuracy. In the end, the best parameters found were: {n_epochs: 100, learning_rate: 0.01, batch_size: 32}.

Unfortunately, after starting our first experiments, we noticed that this learning rate and batch size did not work when fitting the entire training set. Therefore, we resulted to decreasing these two values a little to learning rate = 0.0001 and batch size = 16 in order for it to work properly without outputting NaN values. We now assume that the reason the higher learning rate worked during the tuning is because when this is done with multiple folds, a smaller subsection of the dataset is fitted and therefore using the entire set, which contains 50000 data points, is too large.

## 3.2 Performance vs Epoch

Figure 1 shows plots of test performance on the MLP model with different number of hidden layers (0, 1, 2) as the number of epochs grows. The performance is evaluated with the softmax cross entropy loss. The behavior is different for each model, where the minimum cost is at a different epoch iteration for each one. With no hidden layers, the optimal cost is found after the very first iteration. With 1 hidden layer, the cost decreases and is optimal after 17 iterations, after which the cost consistently increases. With 2 hidden layers, the optimal cost is reached after 8 iterations, after which it increases dramatically.

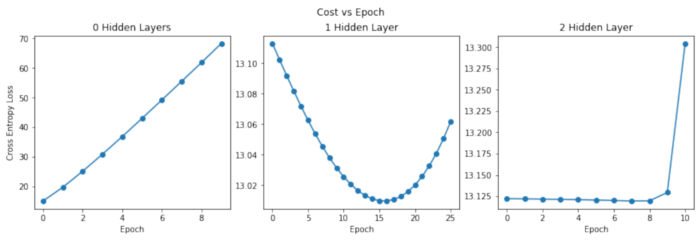


Figure 1: Cost vs Epoch for 3 different models

## 3.3 Early Stopping Regularization

We noticed this behavior early on in our project, which is why we decided to add early stopping regularization. This stops the gradient descent loop once the loss is found to start increasing.

## 3.4 3 MLP Models with ReLU

Table 1 shows the train and test accuracies after running our MLP model with different number of hidden layers. All models used ReLU activation and have 256 units in their hidden layers.

To our surprise, the highest accuracy is a result of the 0 hidden layer MLP, while the lowest results from the 2 hidden layer MLP. Our expectation was the opposite, that having no hidden layers would result in the worse performance and that adding more layers would improve it. It seems like with our model, more depth reduces performance. This could be due to the fact that the data is simple enough to be learned well with a 0-layer MLP.

| | 0 Hidden Layers | 1 Hidden Layer | 2 Hidden Layers |
|---|---|---|---|
| Train Accuracy | 28.02% | 24.77% | 11.88% |
| Test Accuracy | 28.43% | 25.46% | 11.77% |

Table 1: Train and Test Accuracies for the 3 different models

## 3.5 Tanh and Leaky-ReLU

Table 2 shows the train and test accuracies after running a 2 hidden layer MLP with various activation functions, namely tanh and leaky-ReLU. Again, to our surprise, the model trained using tanh had the best accuracy. We expected ReLU to do better since tanh has a vanishing gradient problem whereas ReLU does not. One result that we did expect is that leaky-ReLU performed better than regular ReLU. This would be because this activation avoids having dead neurons, meaning neurons that get stuck in the negative state and cease to contribute to the output. Since we do not ensure that our weights are all initialized to non-negative values, the relu activation function could result in dead neurons.

| | Hyperbolic Tangent (tanh) | Leaky-ReLU | ReLU |
|---|---|---|---|
| Train Accuracy | 19.24% | 14.66% | 11.88% |
| Test Accuracy | 19.49% | 14.43% | 11.77% |

Table 2: Train and Test Accuracies for various activation functions

### 3.6 Evaluating Effect of Width

Figure 2 is a plot of test accuracy versus network width. For this experiment we used a 2 hidden layer MLP with relu activation and adjusted the number of units in the hidden layers. The results indicate that a width of 512 is optimal, however there is no direct correlation between accuracy and width. A higher width generally results in better performance, however not all the time, as a width of 256 has the worst result.

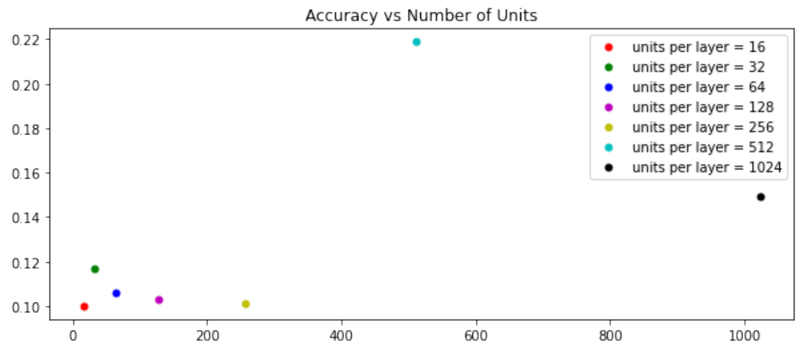


Figure 2: Accuracy vs Width

### 3.7 L1 and L2 Regularization

| | Lambda=1 | Lambda=0 | Lambda=0.1 |
|---|---|---|---|
| L1 Normalization | 12.18% | 8.92% | 9.76% |
| L2 Normalization | 13.73% | 17.67% | 18.55% |

Table 3: Test Accuracies for various lambda values in L1 and L2 normalization

Table 3 shows the accuracy of a 2 hidden layer MLP each having 256 units with ReLU activation plus L1 and L2 normalization. To get a wider range of results, three lambda values were used [1, 0, 0.1]. From these results, we can say that L2 normalization outperforms L1 normalization for every lambda value that was tried. Then, for the L1 norm, the accuracy is varying up and down with no clear direction for the different lambdas. On the other hand, L2 norm seems to perform the worst with lambda=1, even though this accuracy is better than all the ones from L1 norm. Then it does much better, around 28.7% increase in accuracy from a lambda of 1 to a lambda of 0. Finally it increases again by a little bit, around 5% increase from a lambda of 0 to a lambda of 0.1. Finally, in comparison with the original two hidden layer MLP, the L1 norm doesn't help the accuracy much, but the L2 norm greatly increases the accuracy, from 11.77% to 18.55% with lambda=0.1.

### 3.8 Unnormalized Images

| | Normalized Data | Unnormalized Data |
|---|---|---|
| Test Accuracy | 11.77% | 10.49% |

Table 4: Test Accuracies of normalized and unnormalized images

Table 4 shows the accuracy of normalized images and unnormalized images with an MLP having 2 hidden layer each with 256 units and ReLU activation function. Turns out the final accuracy we get is not so different between the two, with only a 1.28% difference. The normalized images does give the higher accuracy, which is

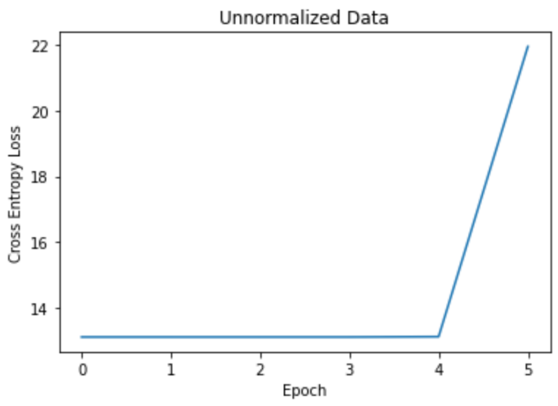
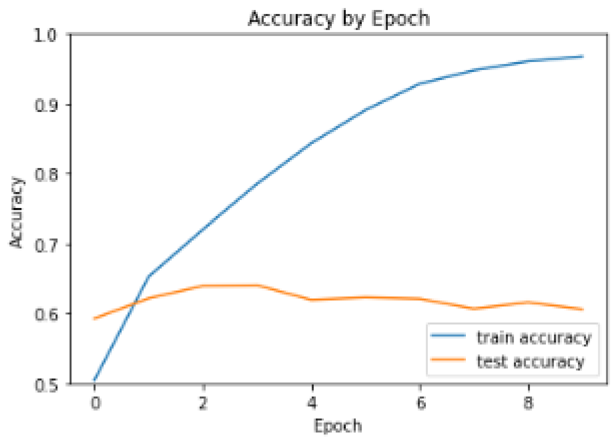


Figure 3: Cost vs Epoch of unnormalized images

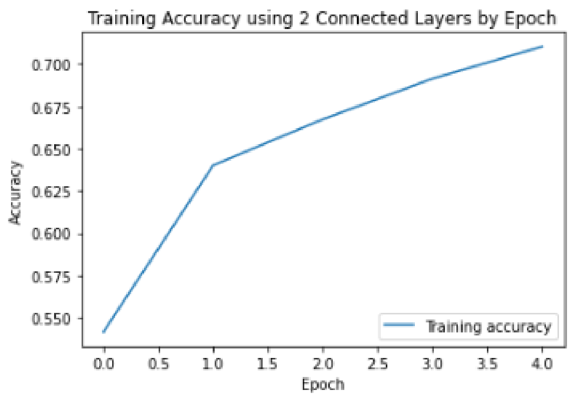
what we would expect at least. Furthermore, figure 3 can be compared with figure 1 on the 2 hidden layer MLP as they use the same hyperparameters. They appear to share a similar trend, with very little change and then a spike upwards.

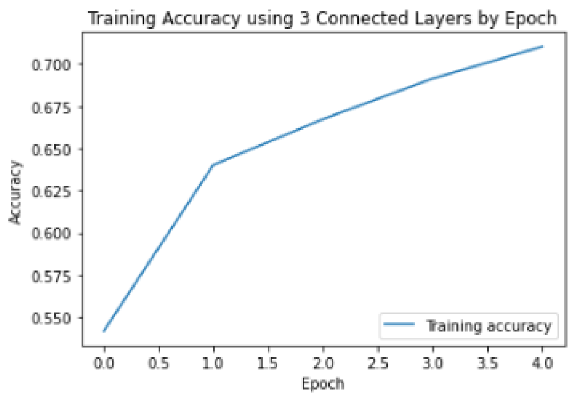
### 3.9 Convolutional Neural Network



The figure above demonstrates the accuracy of a CNN with 2 convolutional and 2 fully connected layers. The table shows the accuracy of the model by each epoch iteration. It shows that it has a greater accuracy rate than the MLP model created in part I. This observation can be justified because the MLP model typically does not know the structure of the image and processes the images in 1x1024 format while CNN can work with the actual shape of the image of 32x32.

### 3.10 Using Pre-trained Model





The figure above demonstrates the accuracy of a pre-trained CNN model with 3 additional untrained connected layers. The table shows the accuracy of the model by each epoch iteration. It shows that it has a greater accuracy rate than both the MLP model created in part I and the experimental CNN model. This result is reasonable because the MobileNetV2 model we are using is pre-trained and has a lot more convolutional layers than our experimental model. Moreover, it is pre-trained with more data than CIFAR-10 can provide. This experiment was also clearly slower and more time consuming than the experimental CNN due to the extra layers which prompted us to reduce the epoch number to 5. The average time was 110ms/step for our experimental CNN while the average time was 162ms/step for this experiment. Finally, we experimented with the number of connected layers added after the pre-trained model. Results concluded that the performance would increase for every layer added. Our model accuracy increased by 1 percent when we went from 2 to 3 layers. However, the average time was increased by 10ms/step.

## 4 Conclusion

In conclusion, our experiment showed that CNN models outperform MLP models when it comes to image classification tasks. We also found that adding more layers to the CNN model can improve its performance, but it significantly slows down learning. Additionally, the results of the experiments on L1 and L2 regularization show that L2 performs better than L1 and that there is no significant difference between using a normalized data set and an unnormalized data set. For future investigations, we suggest tuning other parameters, such as activation functions and the gamma in leaky-relu, as well as investigating other pre-trained models and using different types of data sets.

## 5 Statement of Contribution

In this project, Leo worked mainly on part one, acquiring the data and processing it. Dijian and Amani worked on the MLP model. Then we all worked together to complete the experiments and the report.

## 6 Bibliography

[1] Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.