COMP 551: Applied Machine Learning
Mini-Project 3

Patrick Brebner 260889870, Thomas Hillyer 260680811, Dannie Fu 260874695

April 19, 2020

Abstract: In this project, we investigated the performance of two common neural networks for image classification on the CIFAR-10 dataset. A Multi-layer Perceptron and a Convolutional Neural Network were trained on the data and their accuracy was evaluated as a function of training epoch number. We found that convolutional neural networks outperformed the more basic multi-layer perceptron model on this dataset with overall accuracies of 69 % and 52.27 %, respectively.

1. Introduction

Neural networks are a set of interconnected algorithms that are loosely based on the neurons of the human brain. They can interpret sensory data, recognize patterns, and perform clustering or classification. The training data is translated from real-world data including images, sounds, and text. A neural network consists of multiple interconnected layers which must include an input layer, one or multiple hidden layers, and an output layer, all containing a designated number of nodes. By increasing the depth of the network (Increasing number of hidden layers), and using non-linear activation functions, the neural network can be an exceptional and flexible tool for machine learning. Two common examples of neural networks are the Multi-layer Perceptron (MLP) and the Convolutional Neural Network (CNN). In this project, we investigated the performance of the Multi-layer Perceptron model and Convolutional Neural Network model for image classification on the CIFAR-10 dataset.

MLP is a type of feed-forward neural network that combines multiple perceptrons, a type of linear classifier. MLPs are composed of an input layer to receive data, an output layer to make predictions, and in between, an arbitrary number of hidden layers that provide the computational power. These layers can be fully or sparsely connected and have a wide variety of activation functions. Although MLPs with one layer are capable of approximating any continuous function, it's typical that depth of a network is more valuable than width. The general process of training an MLP, along with all other networks, is to perform whats called a forward pass followed by a backward pass. The forward pass runs a batch of input data through the network and the resulting output is compared to the ground truth to calculate a loss. The backward pass takes this loss and uses backpropagation and the chain rule to calculate partial derivatives (gradients) of the loss with respect to the various weights of the network. These gradients are used to update the weights and the process can be repeated for the remainder of training data or multiple epochs of data.

CNN is one of the main neural networks used for image recognition/classification with object detection and facial recognition as some of the areas where CNNs are used most. As with all neural networks, CNNs process the input data through multiple layers to produce an output. The CNN model will pass each input image through a series of convolution layers followed by fully connected layers before applying an output classification function, such as a softmax function for multi-class classification. The convolution layers, which provide the CNNs their unique benefits, include a filter (kernel) and often include pooling. The essential idea of the convolution layers is that they will convolve on a given image spatially to detect features like edges and shapes. The two main advantages of convolution layers are parameter sharing and sparsity of connections. Parameter sharing occurs since the parameters of the filter are used over all inputs and sparsity of connections occurs since each output value depends only on the filter and the small region of inputs used for convolution. Due to these advantages, using convolution layers can drastically reduce the number of parameters compared to a fully connected network while maintaining a high level of accuracy.

In this project, we evaluated the classification performance of an MLP with 2 layers and a CNN with 2 convolution layers and 3 fully connected layers on the CIFAR-10 dataset. Hyper-parameters were adjusted to optimize the performance of each model. We found that CNNs outperformed the more basic MLP model on this dataset, with accuracies of 69 % and 52.27 %, respectively.

2. Related Work

CIFAR-10 is a popular dataset for image classification and therefore there have been many papers evaluating model performance. In his paper, Akwaboah presented three different CNN approaches for classifying the CIFAR-10 datasets. The first two networks, which had 8 and 7 layers respectively, performed relatively well, however showed drastic over-fitting. The first network achieved a performance accuracy of 86.18% and 71.81% on the training and test set, and the second network achieved performance accuracies of 95.37% and 67.07% respectively for 20 epochs. The third network addressed the over-fitting by employing dropout and L2 regularization, achieving accuracies of 73.21% and 75.43% for 40 epochs. [1]

In another paper, Aboulnaga et al. studied the performance of different classifiers, such as Logistic Regression, Support Vector Machines (SVM), Random Forests, K-Nearest Neighbours (KNN), and a CNN, on the CIFAR-10 dataset and built an ensemble of classifier to achieve a better performance. They used feature engineering methods, like Principal Component Analysis (PCA), to reduce over-fitting. They found that using PCA to reduce overfitting in KNN and ensembling it with CNN increased the best model accuracy from 93.33% to 94.03% [2].

3. CIFAR-10 Dataset

The CIFAR-10 dataset is a common dataset used for image classification. The CIFAR 10 dataset contains 60,000 32x32 colour images (i.e. 3 channels), 50,000 of which are training images and 10,000 test images.There are 10 classes of images: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck, which are each assigned an integer value from 0 to 9 in the dataset. There are 6,000 images per class and each class of image is also mutually exclusive, meaning that there is no overlap between classes like "automobile" and "trucks". The dataset has 5 training batches and one test batch. The test batch has 1,000 randomly selected images for each class. The training batches contain the remaining images and may contain more images from one class than another.

4. Proposed Approach

This section outlines the specific approach taken to build and optimize both of the models for image classification. On top of preparing the data for a neural network, each model has many hyper-parameters than can be tuned to improve the performance of the network.

4.1. Multi-Layer Perceptron

Prior to creating the MLP, the input data must be formatted correctly for the network to be able to run. The CIFAR-10 dataset contains 32x32 colour images, meaning the dimension of one input image is (32,32,3), as there are 3 RGB channels (red, green, blue). We flattened this to a (N x 3072) input, where N is the number of samples. The input images were also normalized such that their values fall between (0,1). The labels of the images were one hot encoded to change the values from 0 to 9 to a binary array.

There are two main parameters to adjust when creating an MLP: number of hidden layers, and number of nodes per layer. In this project, we built a simple 2-layer MLP (i.e. one hidden layer) and adjusted the width of the network (i.e. the number of nodes in the hidden layer). For each hidden layer of the network, we also adjusted the type of activation function, such as ReLu, logistic, tanh etc, to investigate its impact on the classification accuracy.

At the last layer, we used the softmax function to map the output of the network to probabilities over the predicted output classes, and then took the maximum of those values as our final predicted output class. In the implementation of the softmax function, we shifted the data passed in by subtracting the maximum value before performing the exponential function. This was to ensure that the softmax function did not produce overflow due to high input values.

The weights of the network were initialized to be random with uniform distribution, where the range is from [-1/sqrt(3072), 1/ sqrt(3072)]. Mini-batch Stochastic Gradient Descent was used for optimization of the weights. The main hyper parameters for this algorithm are learning rate, batch size, and some termination condition such as number of iterations to run.

4.2. Convolution Neural Network

To load and prepare the dataset for the CNN, we used the pytorch.torchvision package, which is a convenient data loader for common datasets such as CIFAR-10. This eliminates the need of writing a lot of boiler plate code. The torchvision dataset was loaded, transformed to a tensor, and normalized between -1 and 1. Building the network was mainly done with the pytorch torch.nn module and torch.optim for optimization. A GPU was used to enable faster network training speeds.

One of the most crucial choices when designing a neural network is deciding the type and number of hidden layers. When building a convolutional neural network its typically advised to start with a standard CNN architecture, such as LeNet or AlexNet, and make adjustments as necessary to improve performance. These standard network structures generally have good results so they should be used as inspiration. Therefore, our initial CNN, inspired by LeNet-5, had two convolution layers (with max-pooling) followed by 3 fully connected layers. With this configuration, overfitting was already starting to occur so additional layers were likely unnecessary. Regardless, a few other configurations, such as adding a third or fourth convolution layer were tested but they did not produce better results. Although there are many choices for activation functions, Relu was used because it's widely considered as the most effective. At the output layer, the softmax function was used to determine the predicted class.

The other main hyper-parameters of a CNN include: filter/kernel size, padding value, stride value, and pooling layer parameters. As with the number and type of layers, there are some strategies/patterns used to select and optimize these hyper-parameters. Filters are a matrix of weights which are derived while training the network. In general, filters are odd size matrices and smaller filters will gather more fine details of the image. For this reason, and because our images are quite small, we only experimented with small filters of either 3x3 or 5x5. Padding is useful if you want to maintain image size or if you believe there is a lot of useful information at the edge of the image. In the CIFAR-10 dataset, the main object of the image is almost always in the center of the frame so padding is not necessary for this task. Stride dictates how many spaces to shift the filter during convolution. Since a stride greater than 1 decreases the size of the image much faster, it was left as 1 for the network filters. Finally, max pooling is commonly used in CNNs with a 2x2 size and stride of 2.

When building the neural network and increasing its complexity it can become easy for the CNN to overfit, especially when using the training data for multiple epochs. To avoid overfitting, there are several techniques that can be employed such as early stopping and data augmentation. Early stopping can either be done manually, by looking at the results over time, or set up automatically to stop once a certain criteria is met. Data augmentation works on the principle that a larger dataset results in better generalization. To increase the size of the dataset we can add reasonable transformations to the input, such as rotating the image or altering the images brightness, contrast, or saturation. With different variations of the training images every epoch the network is able to generalize much better and avoid overfitting. Both early stopping and data augmentation were used to get the best results from the CNN.

Finally, there are several loss and and optimization algorithms that can be used for CNN. For the loss function we used the cross entropy loss implemented with the pytorch torch.nn module. Two optimization algorithms with learning rates of 0.001 were tested, the stochastic gradient descent with momentum of 0.9 and the Adams algorithm. Both performed well however stochastic gradient descent achieved a test accuracy that was a few percent higher while reaching this accuracy in about 10 fewer epochs. The only advantage of the Adams algorithm was it seemed to help with overfitting although if using early stopping and data augmentation this benefit was minimal. For these reason stochastic gradient descent was used for the CNN.

5. Results

In this section the performance of each model is discussed and compared. Table 1 shows the overall classification accuracy of the MLP and CNN model as well as the classification accuracy for each class. We can see that the convolutional neural network greatly outperformed the multi-layer perceptron model.

Results		
	Multi-layer Perceptron	Convolutional Neural Network
Class	Accuracy (%)	Accuracy (%)
Plane	66.4	81
Automobile	58.0	89
Bird	38.6	51
Cat	26.8	39
Deer	31.5	53
Dog	43.3	64
Frog	79.6	84
Horse	63.0	75
Ship	52.2	77
Truck	63.3	80
Overall	52.27	69

Table 1: Final Model Test Accuracy

5.1. Multi-Layer Perceptron

Overall, the 2-layer MLP performed decently on the CIFAR-10 dataset. The performance of the 2-layer MLP had a highest accuracy of around 52.27%, as seen in Table 1. This model had 2056 nodes in the hidden layer, and used the relu activation function. The class with the lowest accuracy was the cat class, however we can see, in Figure 1 in the appendix, that the other classes had many misclassified instance between similar classes. For example, class 9, which was "truck" had the majority of its misclassified instances as class 1, which was "automobile".

We adjusted several parameters of the MLP model: first we looked at how changing the number of nodes in the hidden layer affected the results, we then looked at how changing the activation impacted the performance. For the MLP implementation, we did not use any methods, such as GridSearch cross validation, to select the best hyper parameters. This would be recommended to do in future improvements of this model.

5.1.1. Changing number of nodes in hidden layer

One of the main parameters of an MLP is the number of nodes in a hidden layer. We tested the model using 4 different number of nodes in the hidden layer: 100, 256, 1024, 2056. We used the Relu activation function for the hidden layer and the softmax function for the outer layer, a batch size of 50, and a learning rate of 0.01 for gradient descent. Figure 2 in the appendix shows the classification accuracy over epochs for each test. The overall accuracy for the model using 100, 256, 1024, and 2056 nodes was 49.34%, 50.95% , 51.79%, and 52.27% respectively. We can see that for lower number of nodes in the hidden layer, we are losing some information, given our input has a dimension of 3072.

5.1.2. Changing activation function

We tested three different activation functions for the hidden layer: Relu, tanh, and the logistic sigmoid function. Figure 3 in the appendix shows the training and testing accuracies over 20 epochs using a batch size of 50 and learning rate of 0.01 when using the three activation functions. The overall accuracies for each model was 50.95%, 50.63%, 42.64%. In general, Relu is thought to be better as it eliminates the vanishing gradient problem that occurs with the logistic sigmoid function. We can see that both the Relu and tanh activation function yield similar results, while the logistic sigmoid function has a lower accuracy in comparison.

5.2. Convolutional Neural Network

The Convolutional Neural Network performed very well on the image classification of the CIFAR-10 dataset, as expected. The final convolution neural network had 2 convolution layers with 5x5 filters and 2x2 max-pooling, and 3 fully connected linear layers. This network structure performed well, even with minimal hyper-parameter tuning. The full details of this network can be seen in Table 2 of the appendix. Overfitting was an issue with the CNN as illustrated in Figures 4 and 5 in the appendix. With no data augmentation, overfitting started to occur at around the 3 or 4 epoch mark. Data augmentation helped reduce overfitting and the model had more consistent train and test accuracies. Another option would of been early stopping at the 3 or 4 epoch mark to address the overfitting problem. Overall, as seen in Table 1, the final CNN model had an accuracy of 69%. Performance varied across classes, with the cat class having the lowest accuracy. This is likely caused by the cat images being mistaken for another class with correlated features.

6. Discussion and Conclusions

With recent advances in deep learning, image classification/recognition has become a popular machine learning task. In this project, we used two models, multi-layer perceptron and convolutional neural networks, to perform image classification on the CIFAR 10 dataset. As expected, the convolutional neural network had a much higher classification accuracy than the 2-layer multi-layer perceptron. We saw that overfitting was a problem in the CNN model; however, this problem was fixed by performing data augmentation. In the MLP model, the model with 2056 neurons and a Relu activation function at the hidden layer yielded the best result.

Future work could be focused on exploring how to improve the MLP classification, for example by increasing the depth of the network. In this project, we saw that with a smaller number of neurons in the hidden layer, we lost some information and therefore had a lower classification accuracy. With a higher number of neurons, we had a higher chance of overfitting the data as there were more parameters to model the data. Therefore, we can expect by increasing the number of layers as well, that we will likely have overfitting. Further experiments could be done to find the optimal number of neurons and layers. In general, to find the optimal hyper-parameters of the model, it is recommended that some kind of grid search cross validation be used in the future. For the CNN, there is a wide variety of network structures and hyper-parameters available. In this project we manually explored several options however it was still not a robust search of the complete parameter space. Developing an automatic algorithm that fully explores this space would be ideal for future models. Finally, other experiments that could be done to improve the classification accuracy of both models is to use ensemble methods, as was done in the paper by Aboulnaga et al., where CNN was combined with KNN to improve the performance.

In conclusion, we explored two models for image classification: multi-layer perceptron and convolutional neural network. We found that the convolutional neural network had a better performance, which was to be expected as it is one of the most popular techniques for image classification. The 2-layer multi-layer perceptron model performed decently, but it still needs further exploration to optimize the parameters.

7. Statement of Contributions

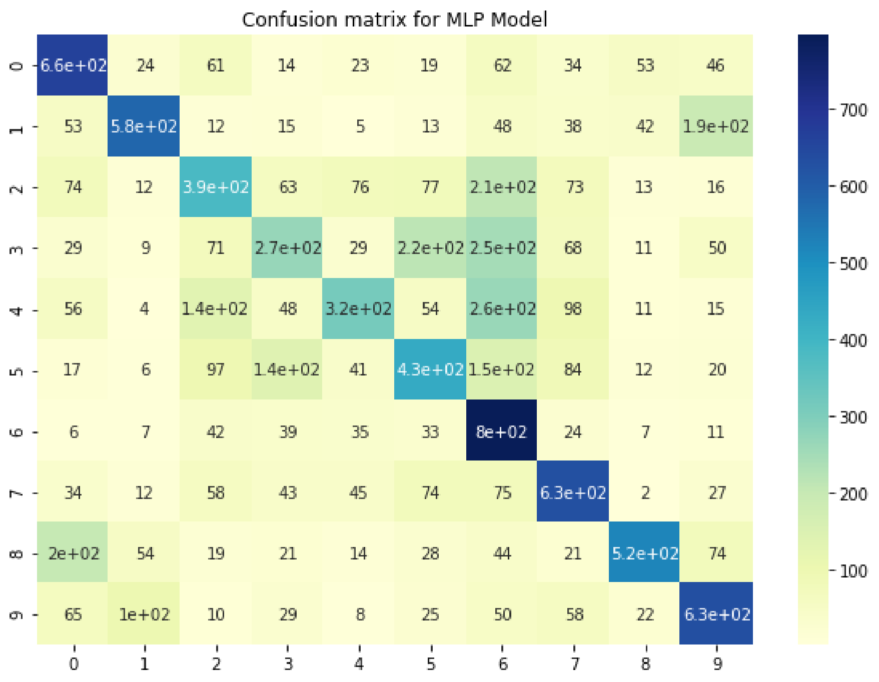
Dannie worked on the Multi-layer Perceptron Model and the report. Patrick worked on the Convolutional Neural Network and the report. Thomas helped debug Multi-layer Perceptron Code.

References

[1] A. D. Akwaboah, Convolutional neural network for cifar-10 dataset image classification (11 2019).

[2] Y. Abouelnaga, O. S. Ali, H. Rady, M. Moustafa, Cifar-10: Knn-based ensemble of classifiers, in: 2016 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE, 2016, pp. 1192–1195.

Appendix



	0	1	2	3	4	5	6	7	8	9
0	6.6e+02	24	61	14	23	19	62	34	53	46
1	53	5.8e+02	12	15	5	13	48	38	42	1.9e+02
2	74	12	3.9e+02	63	76	77	2.1e+02	73	13	16
3	29	9	71	2.7e+02	29	2.2e+02	2.5e+02	68	11	50
4	56	4	1.4e+02	48	3.2e+02	54	2.6e+02	98	11	15
5	17	6	97	1.4e+02	41	4.3e+02	1.5e+02	84	12	20
6	6	7	42	39	35	33	8e+02	24	7	11
7	34	12	58	43	45	74	75	6.3e+02	2	27
8	2e+02	54	19	21	14	28	44	21	5.2e+02	74
9	65	1e+02	10	29	8	25	50	58	22	6.3e+02

Figure 1: Confusion matrix for the MLP model

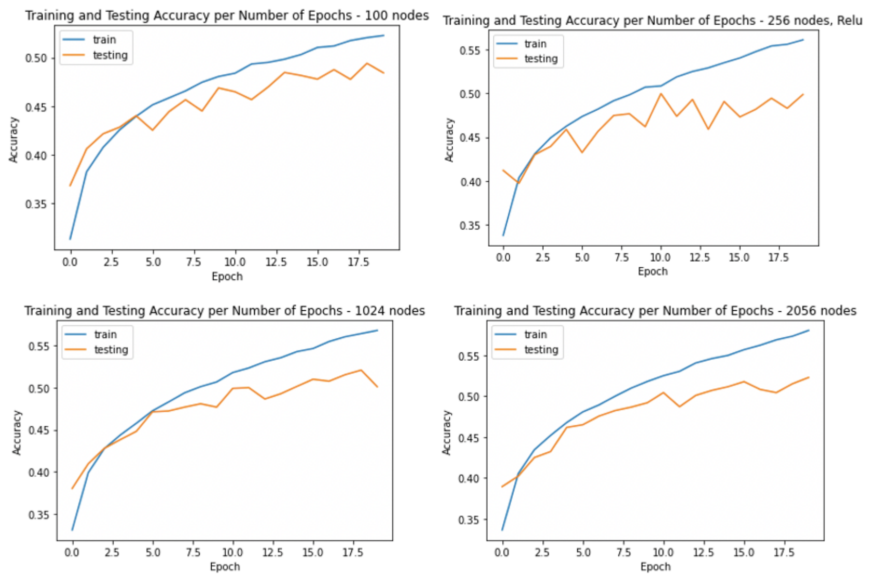


Figure 2: Training and testing accuracy over epochs for various network widths.

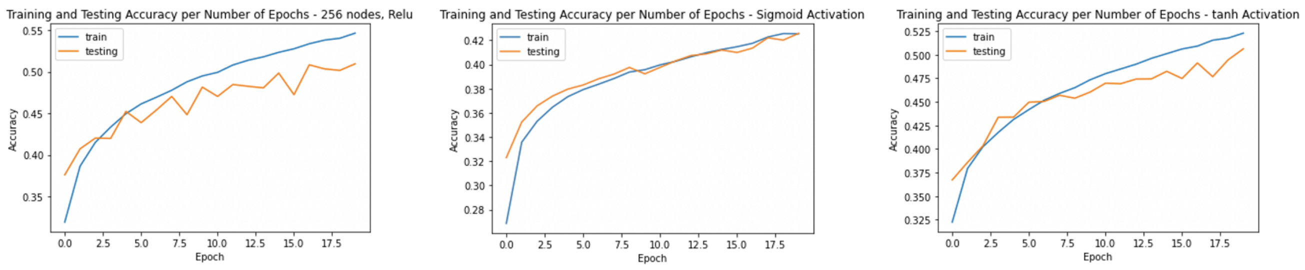


Figure 3: Training and testing accuracy over epochs using different activation functions.

Network Structure				
Layer	Number of Filters	Padding	Shape	Size
Input Image	-	-	(32x32x3)	3,072
Conv2d(f=5,s=1)	32	None	(28x28x32)	25,088
MaxPool(f=2,s=2)	-	None	(14x14x32)	6,272
Conv2d(f=5,s=1)	64	None	(10x10x64)	6,400
MaxPool(f=2,s=2)	-	None	(5x5x64)	1,600
Fully Connected Linear	-	-	(128,1)	128
Fully Connected Linear	-	-	(84,1)	84
Softmax	-	-	(10,1)	10

Table 2: Final Convolutional Neural Network Configuration

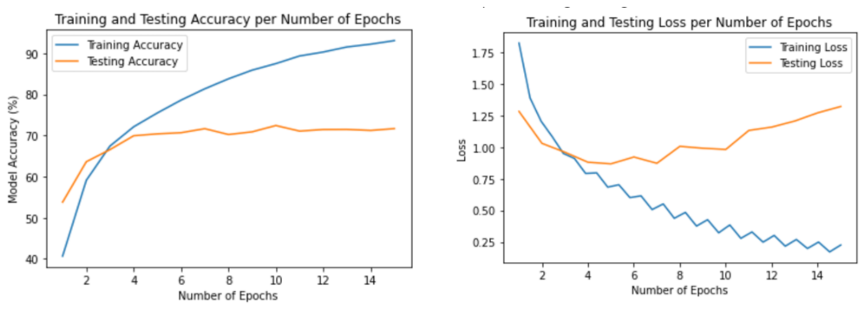


Figure 4: Training and Testing Accuracy/Loss per Number of Epochs with No Data Augmentation

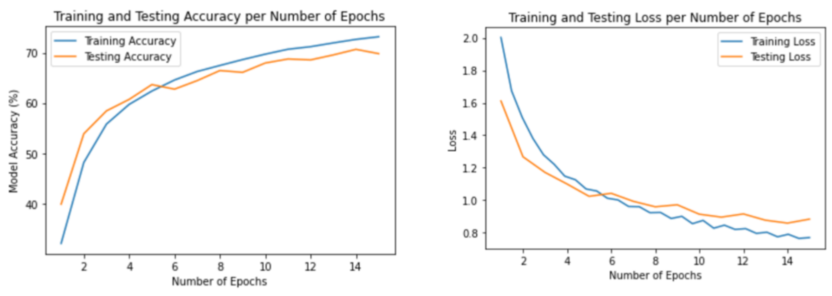


Figure 5: Training and Testing Accuracy/Loss per Number of Epochs with Data Augmentation