# Classification of Image Data with Multi-Layer Perceptrons and Convolutional Neural Networks

1st Sasha Denouvilliez-Pech
*McGill University*
sasha.denouvilliez-pech@mail.mcgill.ca

2nd Shidan Javaheri
*McGill University*
shidan.javaheri@mail.mcgill.ca

3rd Taz Scott-Talib
*McGill University*
taz.scott-talib@mail.mcgill.ca

***Abstract*—Neural networks, mimicking the neural pathways in the brain on a far smaller scale, have an extensive range of uses, including image classification. In this project, multiple neural networks are trained to perform image classification and tested against the CIFAR-10 dataset, which consists of 60,000 color images categorized into 10 different classes. The first neural network explored is a Multi-Layer Perceptron, implemented from the ground up to allow for any number of hidden layers and hidden units, as well as ReLU, Tanh or LeakyReLU activation functions, ending with a softmax output layer. Our implementation primarily uses L2 regularization to prevent overfitting with both Mini-batch and Adam gradient descent. Our exploration of model depth, activation functions, regularization, data normalization, hyper-parameter values and gradient descent optimizers discovers the best model to be 2 layer MLP with 256 hidden units at each layer, ReLU activation functions, and L2 regularization. This model has a test accuracy of 54% after being trained on 20 epochs of data with a batch size of 32 and a learning rate of 0.013 or 0.0004 for Mini-batch and Adam gradient descent respectively. The second model investigated is a Convolutional Neural Network (CNN), implemented using TensorFlow libraries. When hyper-parameter tuning is performed and max pooling added, the CNN has 72% test accuracy. Finally, a transfer learning exercise is performed on the dataset using the ResNet50V2 architecture. Using the pre-trained weights on the ImageNet dataset and training fully connected layers on top of the frozen model, a test accuracy of a 86% is obtained. These major results are summarized in the table below:**

**TABLE I:** Model Performance Summary

| Model | Test accuracy (%) |
|---|---|
| 2 Hidden Layers MLP (ReLU) | 54.1 |
| Benchmark CNN | 72.5 |
| 1 Layer Custom ResNet-50V2 | 86.2 |

***Index Terms*—Convolutional Neural Networks, Gradient Descent, Image Classification, Multi Layer Perceptions, Transfer Learning**

## I. Introduction

The ability of Neural Networks to imitate the learning process of the human brain on a smaller scale makes them a very popular type of Machine Learning algorithm with a wide range of applications, including image classification [1]. This project explores implementations of several neural networks to classify image data from the benchmark CIFAR-10 image dataset [2].

The first neural network the project explores is a Multi-Layer Perceptron (MLP), implemented from scratch in Python primarily with the cupy library. Our implementation is modular, where every layer in the network is a linear layer with weights and biases followed by an activation function layer. All layers, whether linear or non linear, contain a gradient as well as forward and backward methods for data prediction and backpropagation. A softmax output layer is the final layer in the model, so that it can perform image classification. An MLP is instantiated as a connection between several of these layers, and optimized by an optimizer object that calls the forward method on all layers, and then performs backpropagation with the backward method to update all parameters. Our implementation contains Mini-batch and Adam gradient descent optimizers, both of which have the option to add L2 regularization. Our exploration of model depth, activation functions, regularization, data normalization, and hyper-parameter values demonstrates that the best model is a two-layer MLP with 256 hidden units at each layer, ReLU activation functions, and L2 regularization, with either Mini-batch or Adam gradient descent. This model has a test accuracy of 54% after being trained on 20 epochs with a batch size of 32 and a learning rate of 0.013 or 0.004 for Mini-batch and Adam respectively. This is very similar to the 56% accuracy achieved at the university of North Florida [3] with a more complex MLP.

To explore other models, we also implement a Convolutional Neural Network (CNN) using TensorFlow libraries. The CNN architecture consists of 2 convolutional layers and 2 fully connected layers of 256 units, as well as $(2, 2)$ max pooling. When trained with a learning rate of 0.001 and batch size of 32, the CNN has a test accuracy of 72%. Our exploration ends with a transfer learning exercise, using the ResNet50V2 architecture as the base model [4]. The pre-trained weights on the ImageNet dataset are frozen and two fully connected layers are stacked before the classification layer. After training the fully connected layers, the test accuracy achieved is 86%.

## II. Dataset and Data Pre-processing

The CIFAR-10 dataset for image classification is "one of the most popular image datasets in computer vision" [5], and contains 60,000 images of ten different classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each image has a size of $32 \times 32$ pixels with three pixel channels for the red, green, and blue parts of

the image. By default, 50,000 instances are used for training and 10,000 instances are used for testing.

After the dataset is imported, the images must be preprocessed before being being passed into the neural networks. The first preprocessing method performed on the data was the min-max, or linear, normalization. Since pixel values range from 0 to 255, the normalization consists of dividing every pixel value by 255, so every data point ranges from 0 to 1.

The second preprocessing method performed on the data was the z-score normalization, where the average and standard deviation were computed for the whole training dataset. The datapoints are normally distributed with a mean of 0 and a standard deviation of 1. One thing to note is that the statistics are only computed on the training data, but applied to both the training and testing dataset [6]. Section III-D explores this process and demonstrates that z-score normalization is far more effective for the MLP. However, Section III-F reveals that much higher accuracies are obtained with the ResNet50V2 architecture when min-max normalized data is used.

The final step of the data preprocessing is to transform the datasets to fit the models. For the inputs, images are vectorized from a $32 \times 32 \times 3$ tensor to a vector of size 3072. The reshaping is performed for all instances in training and testing. The output labels are one-hot encoded for the classification.

## III. Results

To obtain the highest test accuracy possible, we explore the effect of network depth and non-linearity, as well the effect of different activation functions, regularization, and data normalization on model performance. To do so, we primarily use mini-batch gradient descent with a batch size of 32 [7], and a learning rate of 0.013, through hyperparameter tuning shown in figure 1 below, as well as in figure 8 (see appendices). The training set was divided with a 90-10 split into training and validation sets, to calculate loss on the validation set during training to see if performing early stopping was necessary. All models are trained on 24 epochs of data (33750 iterations), and loss was calculated as categorical cross entropy loss. All weights were initialized to a random number with a variance of $2.0/n$ [6], which produced the best performance compared to other options. All biases were initialized to 0. All results were obtained using z-score normalized data.

For simplicity of reference, the number of "layers" in an MLP referred to in this section will be its number of hidden layers. We will always ignore the input layer and output softmax layer in the count, since they are always necessary.

### A. Network Depth and Non Linearity

To determine the effect of network depth and non linearity on our model, we tested an MLP that had between 0 and 3 hidden layers, each with 256 hidden units and ReLU activation functions when applicable, all with a softmax output layer on the end. The results demonstrate that adding non-linearity greatly increases model performance, as it makes the model much more expressive [1]. The 0-hidden-layer MLP, with no non-linearity by definition, had a training accuracy of 36% and a test accuracy of 31%. This is a stark contrast to all models with hidden layers that add non-linearity, each with a training accuracy of 79%, 81% and 87% respectively, and test accuracies from 45 - 47 %. The higher training accuracy for the models with more layers indicates that overfitting may be starting to occur. Regularization is used to improve the performance of these models as a result, in Section III-C. The results for this section illustrated in Figure 2 below.

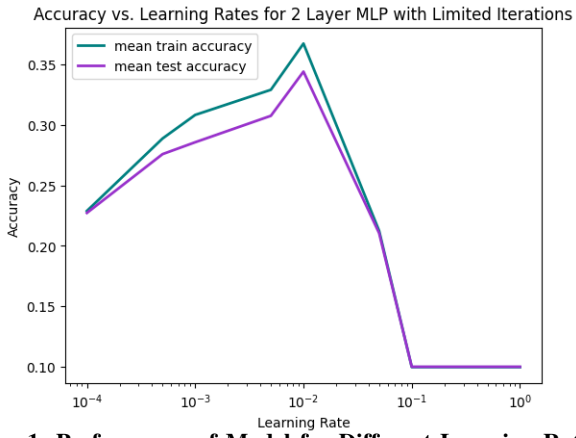


**Fig. 1: Performance of Model for Different Learning Rates**

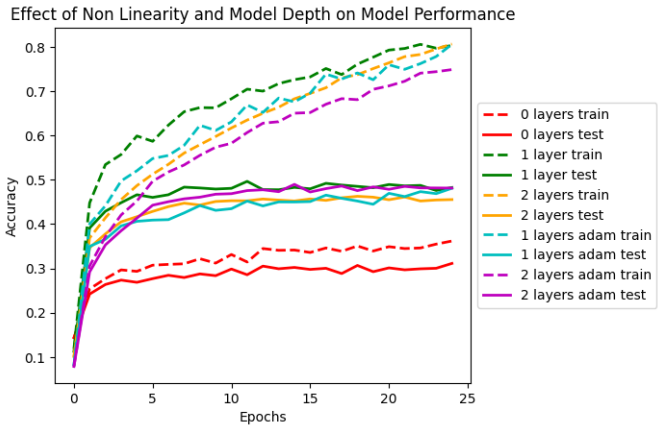


**Fig. 2: Performance of Model for Different Learning Rates**

The results also indicate that model depth does not have a major impact on performance, perhaps because the dataset may be too complex for a simple MLP to achieve very high test accuracy. The model with the highest test accuracy of 47% was the 1 layer MLP, which may support the Occam's razor hypothesis that the simplest model generalizes best [1]. Experiments on MLPs with even more layers are omitted from the report, as even with 7 layers, similar or lower accuracy was achieved. However, using Adam gradient descent, the performance of the two-layer MLP could be increased to 49%, indicating that the addition of momentum and an adaptive learning rate to gradient

descent slightly increases performance and the accuracy of the deeper MLP.

While training each model for 24 epochs drastically increased training accuracy compared to test accuracy, overfitting was barely observed, as validation loss stayed almost completely inline with training loss for all models. This shows that early stopping alone would not have improved model performance significantly in this case. An example of this for the highest performing 2 layer MLP with Adam gradient descent is shown in Figure 3 below, with the figures for other models attached in the appendices (Figures 9 - 13).

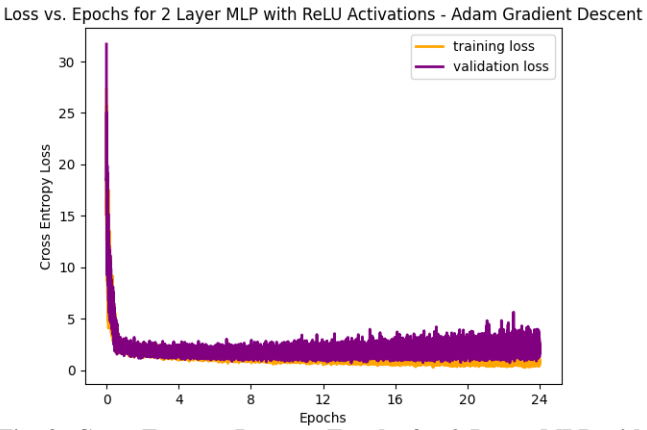


**Fig. 3: Cross Entropy Loss vs. Epochs for 2 Layer MLP with ReLU Activations - Adam Gradient Descent**

### *B. Different Activation Functions*

The first modification made to the 2 layer MLP was the replacing of its ReLU activations with a hyperbolic tangent function. This updated model had a noticeably lower test accuracy than its ReLU-activated predecessor (41%), despite an increase in training accuracy to 90% (Figure 4). The overfitting by the $tanh$ activation could be a result of a vanishing gradient [1]. The function has little change as it strays further from 0, resulting in very small derivatives. Since the neural network has multiple hidden layers, the gradient decreases exponentially during back propagation, and model weights are not adequately updated [8]. The overfitting is also visible by high variance and increasing validation loss shown in Figure 14 (see appendices).

Another activation function used was LeakyReLU, with a negative slope value of 0.1. The goal of this function is to account for the zero-gradient fallback of the original Rectified Linear Unit: when model weights are below zero, or "inactive", the derivative of the activation function is zero and gradients are not updated [9]. Since the original ReLU model was initialized with a random weight matrix as opposed to one composed of zeroes, this problem should not have occurred. Thus, the very slight performance increase of under a 1% change in test accuracy in the LeakyReLU MLP model is close to negligible (Figure 4). To further explore the difference between the two models, we could have compared them for different values of the negative slope value to see what impact that would have. However, the tendency of slightly more weights being "active" also seems to make the model slightly more prone to overfitting.

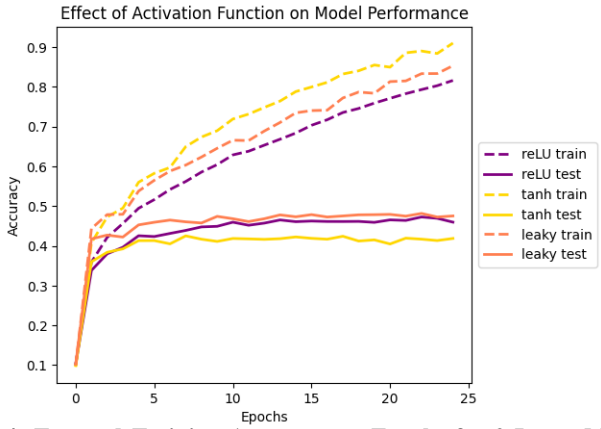


**Fig. 4: Test and Training Accuracy vs. Epochs for 2 Layer MLPs with Different Activation Functions**

According to these observations, ReLU activation functions (LeakyReLU included) are a better choice for multilayer neural networks than the hyperbolic tangent. Due to the vanishing gradient problem, weight and bias terms are updated more effectively when the Rectified Linear Unit is used, making predictions on the test set more accurate.

### *C. Regularization*

Regularization had a significant impact on performance of 2 hidden layer MLPs with 256 hidden units with both Mini-batch and Adam gradient descent, improving test accuracy in all cases as shown in Figure 5. The model trained with L1 regularization [10] negated the overfitting behaviour of the base model, resulting in drastically reduced training accuracy to 48% and very similar test accuracy of 46%. Test and training curves that are extremely close together and mimic each other's behaviour. The benefits of regularization were more noticeable in the L2-regularized model, however, as test accuracy improved to values consistently exceeding 50%, reaching the highest accuracy of any MLP model at 54% for both Adam and Mini-batch gradient descent. Training accuracy also suffered less. Test and training values approaching each other, visible in the figure below, indicating a clear improvement over the original MLP and demonstrating a viable remedy for potential overfitting.

For both L1 and L2 regularization, tuning the $\lambda$ hyperparameter caused similar behaviour. As $\lambda$ approaches 0, the model accuracy reflects that of the non-regularized model since the penalties applied to the MLP weight matrix become less and less significant. As such, with low $\lambda$ values, training accuracies above 70% and much larger than test accuracies are obtained, indicating the danger of overfitting. However, when $\lambda$ is increased above optimal values, though the model has test and training accuracies that are almost equal, overall loss is higher and performance suffers. This

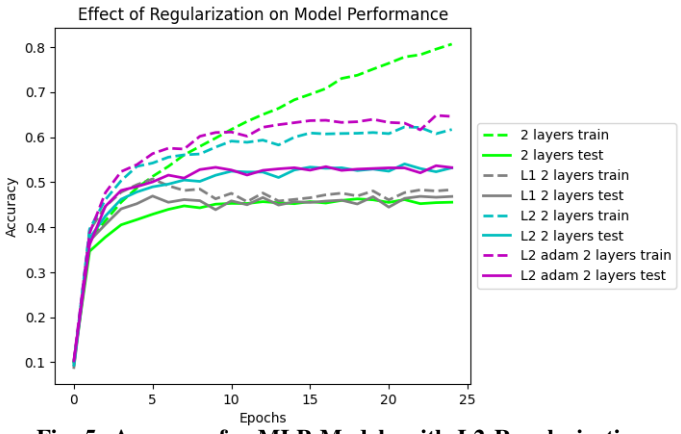


**Fig. 5: Accuracy for MLP Models with L2 Regularization**

can be explained by the fact that an overly large weight penalty has too much influence on weight updates, resulting in a model that underfits, never reaching ideal prediction accuracy [11]. The hyper-parameter effects can be observed in Table II.

**TABLE II:** Effect of $\lambda$ on L2 Regularization

| **Hyper-parameter Value** ($\lambda$) | **Model Accuracy (%)** | |
|---|---|---|
| | ***Training*** | ***Test*** |
| 0.0001 | 80.6 | 45.8 |
| 0.001 | 82.7 | 48.4 |
| 0.005 | 73.9 | 52.4 |
| 0.01 (optimal) | 65.1 | 54.2 |
| 0.05 | 45.9 | 45.5 |
| 0.1 | 36.2 | 36.5 |

### *D. Normalization*

As expected, training an identical model on un-normalized data decreased prediction accuracy significantly. When the MLP with two hidden layers, 256 units and ReLU activation functions was given raw input data (no linear or z-score normalization), its weights were far too large to be computed, resulting in numeric overflow during matrix multiplications. This meant that loss could not be plotted on a graph, and accuracy was minimal since the model was not really performing any mathematical predictions.

We also explored which normalization technique would lead to the best results, comparing normalizing with the min-max technique to z-score normalization, as well as using grayscale data. Results clearly indicate that z-score normalized data leads to the best results with the MLP, with the results for other techniques incomparably lower, having test accuracies below 11%. This justifies the fact that all models were trained in preceding sections using z-normalized data.

### *E. Convolutional Neural Networks*

To investigate the behaviour of Convolutional Neural Networks (CNNs), TensorFlow librairies are used to create a simple CNN with two convolutional layers and 2 fully connected layers of 256 units. Training this model yields a maximum test accuracy of 66% - a significant improvement over the MLP model, demonstrating the superiority of the use of CNNs in image classification [1]. To increase accuracy, we explore using max pooling layers with a pool size of $(2, 2)$ after each convolutional layer [12]. Max pooling layers "aggregate the feature maps", reducing dimensionality and increasing model generalization as well as computational efficiency [13]. Adding a max pooling layer after each convolutional layer yields a maximum of 72% test accuracy, which justifies the use of this architecture for further analysis. This is demonstrated in Figure 6 below.

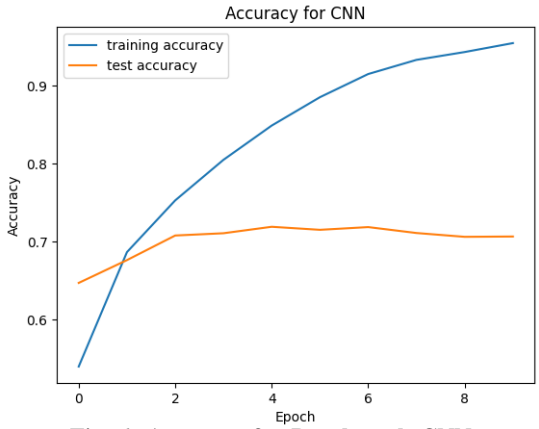


**Fig. 6: Accuracy for Benchmark CNN**

Analysing the performances of the benchmark CNN architecture for multiple learning rates associated with the Adam optimization algorithm ranging from $6.25e{-5}$ to 0.64 shown in Figure 15 (see appendices) further supports the default value of 0.001 suggested by the authors of the method [14]. Further hyper-parameter tuning on the kernel size and stride is also investigated (see appendicies Figure 16). For kernel size, values are tested in a range from 1 to 6. Results from our experiments align with literature [15], suggesting that the optimal kernel size given the benchmark CNN architecture and the image data size is $(3, 3)$. Due to architectural constraints, only 2 values of strides could be tested, with preliminary results indicating that unitary strides are the optimal choice for the benchmark CNN architecture.

### *F. Pretrained Model*

The ResNet-50V2 was used to perform a transfer learning exercise on the CIFAR-10 dataset [4]. For this transfer learning exercice, the min-max normalized data was used since it led to increased test accuracy. Also, since the ResNet-50V2 was trained on $224 \times 224 \times 3$ images, the instances from the CIFAR-10 dataset were upscaled to the expected size [16]. The classification layer on the top was removed and the 23,564,800 weights based on the ImageNet dataset were frozen.

The first step was to replicate the initial ResNet-50V2 architecture to suit 10 classes of CIFAR-10 instead of the 1000 classes of ImageNet. The classification layer, a dense layer with 10 units and a softmax activation function,

was stacked on top of a global average pooling layer. Only 20,490 out of the 23,585,290 total parameters of the model are trainable. Training the custom implementation of the ResNet-50V2 for 10 epochs gave a maximum testing accuracy of 85%.

The second step was to add a fully connected layer on top of the global average pooling layer. Given the output of the latter is a vector of size 2048, intuition leads to choose a size of 2048 for the dense layer. The reasoning behind this is to imitate the AlexNet stacking of constant size dense layers before the classification layer. Adding a dense layer allows an increase in the number of trainable parameters to 4,216,842 out of 27,781,642 [1]. Training the implementation of the ResNet-50V2 with an additional dense layer gave a maximum testing accuracy of 86%, which is the highest accuracy reached for this project, and is demonstrated in Figure 7 below.

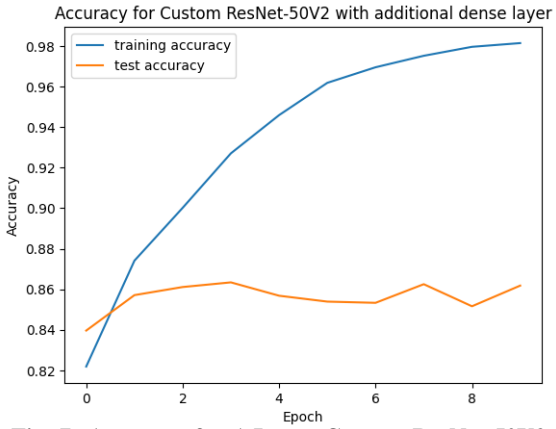


**Fig. 7: Accuracy for 1 Layer Custom ResNet-50V2**

The last step was to add a second fully connected layer right before the classification. The architecture is the ResNet-50V2 with frozen weights and no top layer, a global average pooling layer, two dense layers of 2048 units with ReLU activation, and a classification layer. This architecture has 8,413,194 trainable parameters out of 31,977,994. Training the ResNet-50V2 with two additional dense layers gave a maximum testing accuracy of 86%, which indicates the additional dense layer does not provide improvements. Still, it is important to keep in mind that only ten epochs were completed, and more extensive training on the model is required to determine the use of that layer.

## IV. Discussion & Conclusion

This project explored classification on the CIFAR-10 dataset using multiple models. The first model was a Multi-Layer Perceptron (MLP) implemented from the ground up. A modular approach was favored to allow for extensive analysis. The MLP was trained and tested with 0 - 3 hidden layers all with 256 activations, and different activation functions including ReLU, LeakyReLU, and tanh. In addition, extra computations were performed to improve the performances of MLPs. In particular, the effects of L1 and L2 regularization were observed as well as the effects of normalization and the use of the Adam gradient descent algorithm. Our exploration led us to conclude that adding nonlinearity to an MLP drastically increases its performance and expressiveness, but adding more layers did not have much effect on this dataset. LeakyReLU and ReLU were both shown to be efficient activation functions with similar performance, tanh being very ineffective and prone to overfitting. Both L1 and L2 regularization are effective at preventing overfitting, but L2 regularization leads to the maximum test accuracy. Finally, z-score normalized data leads to by far the best performance. The maximum test accuracy reached was 54.2% on a two-hidden-layer MLP with 256 ReLU activations at each layer, L2 regularization, and either Adam or Mini-batch gradient descent.

The second model explored was a Convolutional Neural Network implemented with TensorFlow libraries. A simple CNN architecture was built using two convolutional layers, each followed by max pooling layers, and two fully connected layers. Experimenting with the learning rates for the Adam optimization algorithm as well as with the kernel size and stride size for the convolutional layers reinforced the use of the default values. The maximum test accuracy reached was 72.5% using max pooling.

The third and final model explored was a custom implementation of the ResNet-50V2. By freezing the 23,564,800 weights in its base layers trained on the ImageNet, adding a global average pooling layer, and two fully connected layers of 2048 units yielded the maximum test accuracy measured in the report, of 86.2%. All results are summarized in the table below.

**TABLE III:** Model Performance Summary

| Model | Test accuracy (%) |
|---|---|
| 2 Layer MLP | 54.2 |
| Basic CNN | 65.5 |
| Benchmark CNN | 72.1 |
| Custom ResNet-50V2 | 85.2 |
| 1 Layer Custom ResNet-50V2 | 86.2 |
| 2 Layer Custom ResNet-50V2 | 86.2 |

Moving forward, our team would like to further investigate the different types of computations or practices that can be done to improve test accuracy. In particular, looking into what type of normalization better fits image data since z-score normalized data performed generally better on models trained from the ground up and min-max normalized data performed better with transfer learning. Also, it would be interesting to look into proper cross-validation to determine the best hyper-parameters for the models, increasing our confidence in our final choices.

### A. *Statement of Contributions*

All group members contributed equally to the project and to the report, and were familiar with all parts of code.

## References


[1] R. Rabbany, "Applied Machine Learning," in McGill Computer Science: COMP 551, 09-Mar-2023.

[2] "The CIFAR-10 dataset," CIFAR-10 and CIFAR-100 datasets. [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html. [Accessed: 09-Mar-2023].

[3] A. RUDOLPH and N. KULKARNI, "CIFAR-10: RESEARCH USING CNN AND MLP MACHINE LEARNING MODELS," 2022.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," arXiv.org, 25-Jul-2016. [Online]. Available: https://arxiv.org/abs/1603.05027. [Accessed: 09-Mar-2023].

[5] "Once upon a time in CIFAR-10," Medium, 02-Sep-2022. [Online]. Available: https://franky07724-57962.medium.com/once-upon-a-time-in-cifar-10-c26bb056b4ce. [Accessed: 09-Mar-2023].

[6] R. Gao, F.-F. Li, and J. Wu, "CS231n Convolutional Neural Networks for visual recognition," Stanford University. [Online]. Available: https://cs231n.github.io/neural-networks-2/. [Accessed: 09-Mar-2023].

[7] D. Masters and C. Luschi, "Revisiting small batch training for Deep Neural Networks," arXiv Vanity. [Online]. Available: https://www.arxiv-vanity.com/papers/1804.07612/. [Accessed: 09-Mar-2023].

[8] C.-F. Wang, "The vanishing gradient problem," Medium, 08-Jan-2019. [Online]. Available: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484. [Accessed: 09-Mar-2023].

[9] S. Singh, "Leaky ReLU as an activation function in neural networks," Deep Learning University, 29-Aug-2021. [Online]. Available: https://deeplearninguniversity.com/leaky-ReLU-as-an-activation-function-in-neural-networks/. [Accessed: 09-Mar-2023].

[10] J. McCaffrey, "Neural network L1 regularization using Python," Visual Studio Magazine, 05-Dec-2017. [Online]. Available: https://visualstudiomagazine.com/articles/2017/12/05/neural-network-regularization.aspx. [Accessed: 09-Mar-2023].

[11] U. Tewari, "Regularization-understanding L1 and L2 regularization for Deep Learning," Medium, 10-Nov-2021. [Online]. Available: https://medium.com/analytics-vidhya/regularization-understanding-l1-and-l2-regularization-for-deep-learning-a7b9e4a409bf. [Accessed: 09-Mar-2023].

[12] "Convolutional Neural Network (CNN): Tensorflow Core," TensorFlow. [Online]. Available: https://www.tensorflow.org/tutorials/images/cnn. [Accessed: 09-Mar-2023].

[13] V. Christlein, L. Spranger, M. Seuret, A. Nicolaou, P. Král, and A. Maier, "Deep generalized Max pooling," arXiv.org, 14-Aug-2019. [Online]. Available: https://arxiv.org/abs/1908.05040. [Accessed: 09-Mar-2023].

[14] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," arXiv.org, 30-Jan-2017. [Online]. Available: https://arxiv.org/abs/1412.6980. [Accessed: 09-Mar-2023].

[15] X. Ding, X. Zhang, Y. Zhou, J. Han, G. Ding, and J. Sun, "Scaling up your kernels to 31x31: Revisiting large kernel design in CNNs," arXiv.org, 02-Apr-2022. [Online]. Available: https://arxiv.org/abs/2203.06717. [Accessed: 09-Mar-2023].

[16] M. Esmail Karar and M. A. Shouman, "Cascaded deep learning classifiers for computer-aided diagnosis of COVID-19 and pneumonia diseases in X-ray scans," ResearchGate, 09-Sep-2020. [Online]. Available: https://www.researchgate.net/publication/344341113_Cascaded_deep_learning_classifiers_for_computer-aided_diagnosis_of_COVID-19_and_pneumonia_diseases_in_X-ray_scans. [Accessed: 10-Mar-2023].


## Appendix

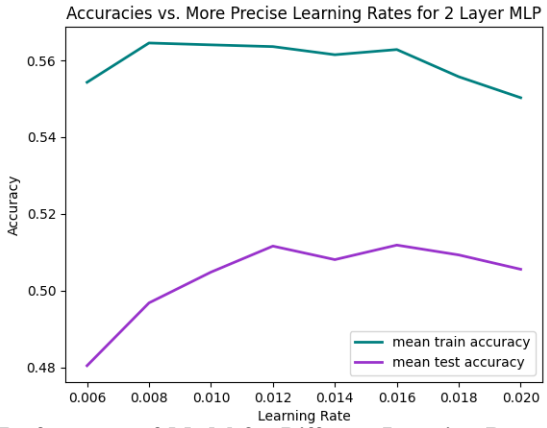


**Fig. 8: Performance of Model for Different Learning Rates - Small Range**

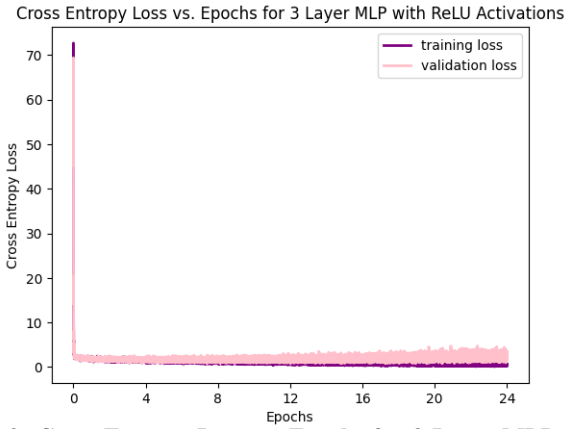


**Fig. 9: Cross Entropy Loss vs. Epochs for 3 Layer MLP with ReLU Activations**

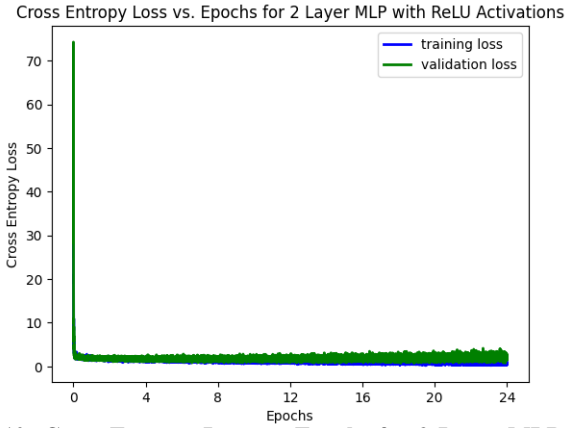


**Fig. 10: Cross Entropy Loss vs. Epochs for 2 Layer MLP with ReLU Activations**

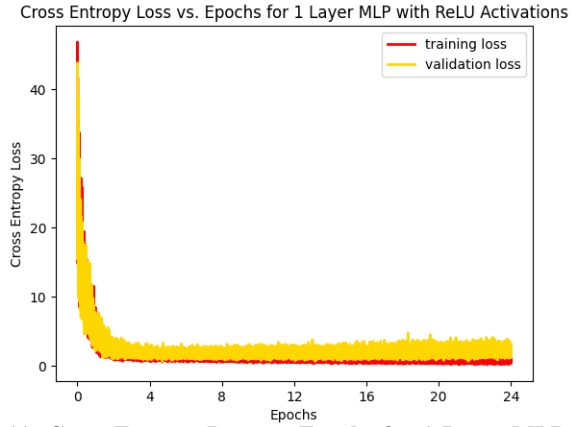


**Fig. 11: Cross Entropy Loss vs. Epochs for 1 Layer MLP with ReLU Activations**

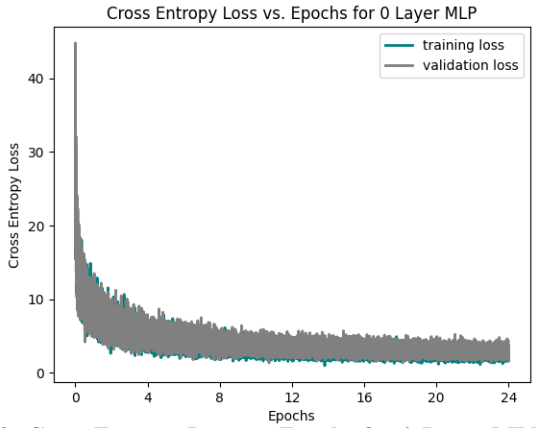


**Fig. 12: Cross Entropy Loss vs. Epochs for 0 Layer MLP with ReLU Activations**

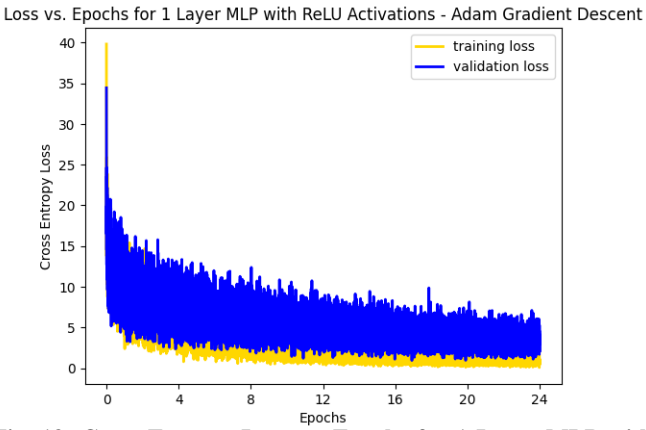


**Fig. 13: Cross Entropy Loss vs. Epochs for 1 Layer MLP with ReLU Activations - Adam Gradient Descent**

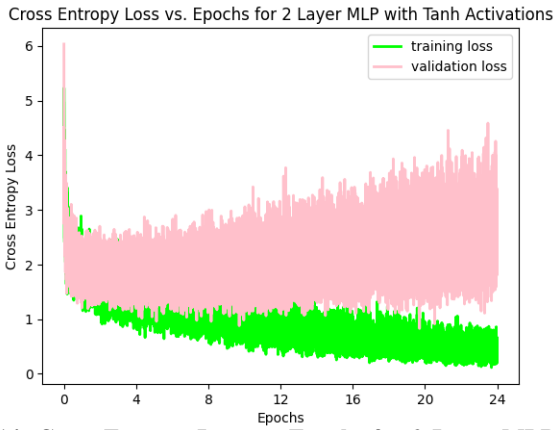


**Fig. 14: Cross Entropy Loss vs. Epochs for 2 Layer MLP with Tanh Activations**

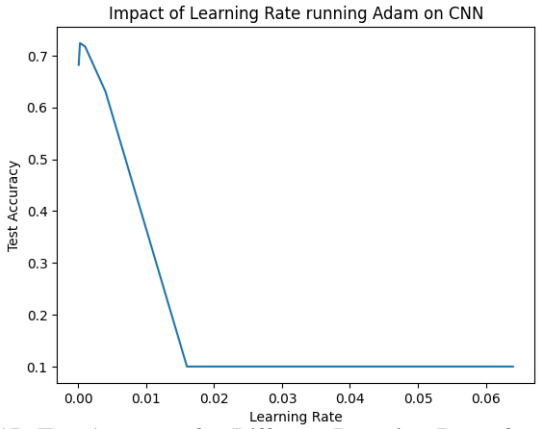


**Fig. 15: Test Accuracy for Different Learning Rates for CNN**

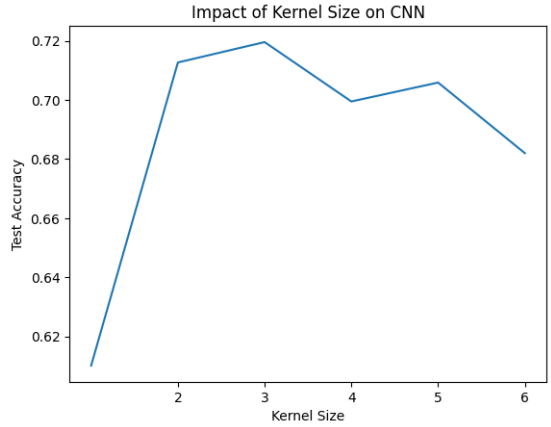


**Fig. 16: Test Accuracy for Different Kernel Sizes on Benchmark CNN**