Inayat Irfan: 260925580
Emile Riberdy: 260985547

Abstract

This report evaluates and analyzes various experiments conducted by implementing a multi-layer perceptron (MLP) trained on the sign language MNIST dataset. It details the implementation and the various steps taken to optimize the relevant hyperparameters of an MLP as well as strategies to best capture non-linear data through strong activation functions, hidden layers, node optimization and learning regularization/optimization. These performances of various MLP models were compared to the performance of the Keras built-in Convolutional Neural Network (CNN) implementation. These CNN models further delved into to show opportunities for strong optimization of accuracy and what levers CNN models have, including filters, filter sizes and strides to be best placed for image classification problems. This report demonstrates the importance of choosing the correct hyperparameters for a given MLP, and how to build the correct CNN architecture for the strongest image processing.

Introduction

The main task for this assignment was to implement a multi-layer perceptron (MLP) for image recognition from scratch using the Sign Language MNIST dataset, which consists of images representing letters made in sign language. Various hyperparameters for the MLP were to be tested, with the performance evaluated for each change in hyperparameters. Then, a Convolutional Neural Network (CNN) was to be trained on the same dataset, with its performance compared to that of the MLP. Different numbers of layers and hidden units were tested for the MLP, and it was found that a 1-layer MLP with 128/256 hidden units is the most consistent performer. However, a 2-layer MLP with 256 hidden units provided the optimal accuracy performance of 60%. Further, we discovered the benefit of Leaky-ReLU compared to ReLU and Sigmoid in capturing non-linear relationships without a dying gradient. L2 Regularization additionally showcased how to broadly boost the accuracy of the MLP model, with the need for it to be tuned.

The strongest CNN model, with a test accuracy of 91%, was achieved when training the model with 256 hidden units. Higher node counts showed better test accuracy, showcasing the breadth of data available for training and the lack of overfitting. A moderate filter number of approximately 32, with a larger filter size and a lower stride, was the strongest configuration to boost the accuracy of a CNN, with smaller sizes compromising spatial context and high strides skipping necessary data. Padding was not found to be a meaningful optimization, as the core features of images in the MNIST dataset are not at the border. It can be concluded that CNNs are better suited for image classification tasks.

Dataset

Only one dataset was required for this experiment. The sign language MNIST dataset is a dataset of images representing sign language letters. The images are 28x28, represented by a pixel value vector of 784 values between 0 and 255. This vectorized format could be used directly for the MLP, but had to be reshaped to a 28x28x1 shape for the CNN. The data was standardized so that it has a mean of 0, and a standard deviation of 1. The training set is composed of 27455 cases, which were all used for training, and the test set is composed of 7172 test cases. No other manipulation was done on the initial dataset. The data also seems to have a strong class distribution for the volume of data in the training set; this ensures the model does not have selective training bias when attempting to classify images in the test set.

Results

In this section, we will present the results obtained from the experiments and address the questions outlined in Task 3 of the assignment.

Comparison of performance for MLP of different layer numbers

As observed in Figure 1, an MLP with no hidden layers performed poorly at capturing the non-linear relationships present in images, with the highest test accuracy being 0.43. This outcome is anticipated since an MLP with no hidden layers functions as a single perceptron, effectively a linear classifier. MLPs with 1 hidden layer show consistent test accuracy across different hidden layer node counts. However, the optimal performance is achieved at a node count of 128, after which the addition of nodes does not significantly increase benefits, with a node count of 256 yielding the same level of accuracy. Models with higher node counts also achieved convergence in cross-entropy minimization at lower epochs. Models with 2 hidden layers exhibited the most variation in output configurations, indicating a susceptibility to overfitting and an underfitting problem with 32 nodes. However, a 256-node model with two hidden layers demonstrated the strongest confidence across all models at the lowest epoch.

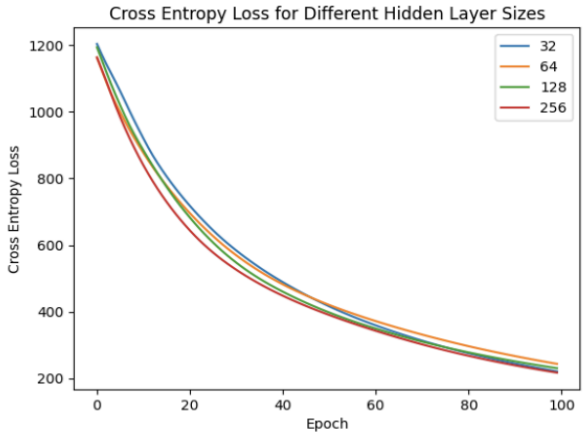


Figure 1: Cross entropy loss for 0-layer MLPs of different hidden unit size

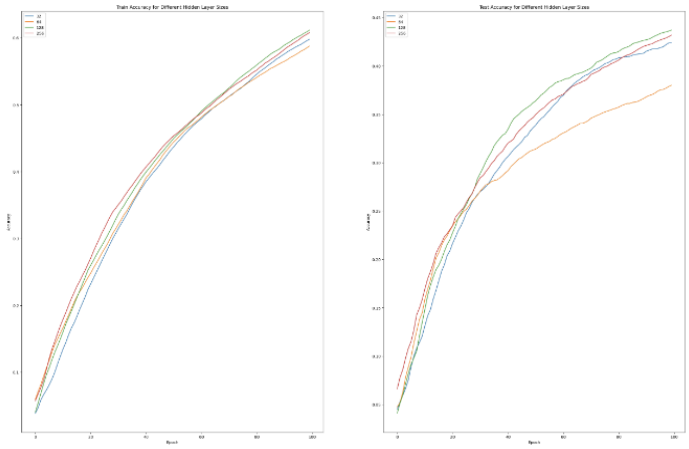


Figure 2: Training and testing accuracy for 0-layer MLPs of different hidden unit size

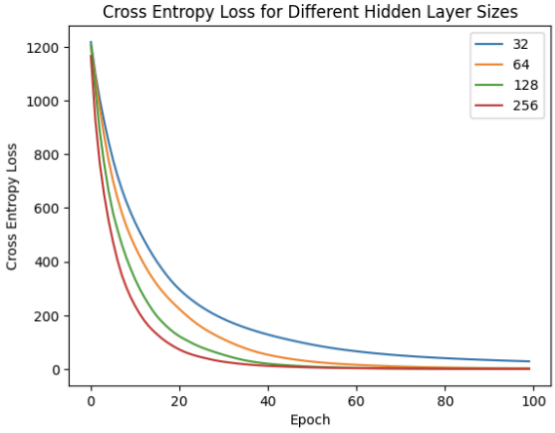


Figure 3: Cross entropy loss for 1-layer MLPs of different hidden unit size

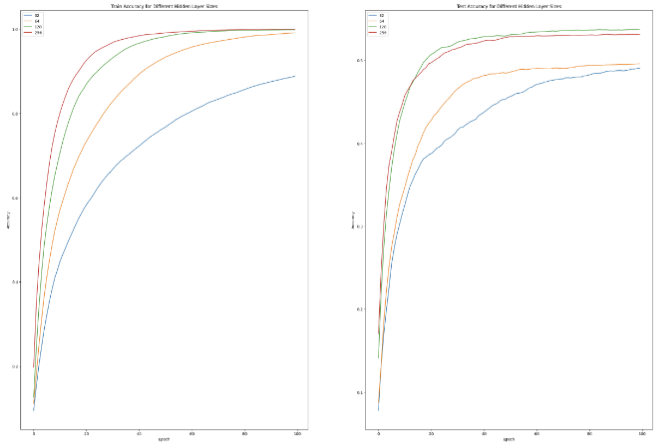


Figure 4: Training and testing accuracy for 1-layer MLPs of different hidden unit size

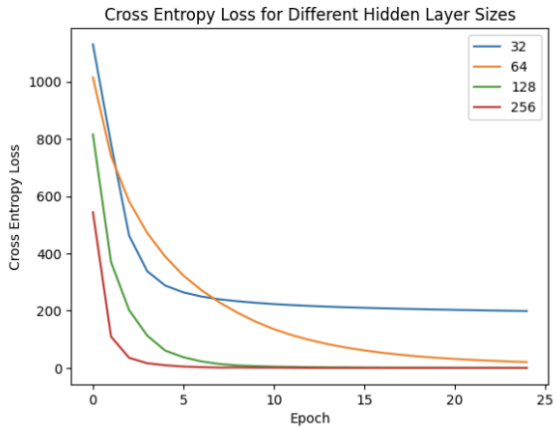


Figure 5: Cross entropy loss for 2-layer MLPs of different hidden unit size

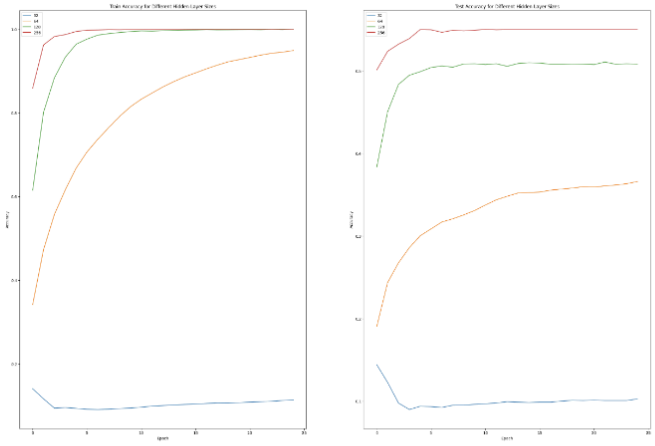


Figure 6: Training and testing accuracy for 2-layer MLPs of different hidden unit size

Different Activation Functions

The strength of an activation function in an MLP is to capture non-linear relationships that are present in datasets. Without the ability to capture these nuances, a network, regardless of its robustness in-depth, learning rate, or other hyperparameters, will be held as a single-layer perceptron capable of solving linearly separable problems. An activation function is applied to the output of a layer prior to its 'passing'. A ReLU is a function that effectively outputs the input if the value is positive; otherwise, it is a zero [negative value]. It is a widely used function as it is computationally efficient and provides aid in solving the vanishing gradient problem as it allows the derivative in training to be 0 or 1. A Leaky-ReLU is a variant that allows a small positive gradient when input values are lower than 0, this is powerful in allowing gradients to propagate in the model but increases the need for training/tuning. A sigmoid activation is a function that outputs a probability between 0 and 1. It's powerful for binary classifications but is prone to a vanishing gradient in neural networks.

As we can see in Figure 7, when using different activations in our 2-hidden layered MLP, we observe that test accuracies vary for the models. While ReLU and Leaky-ReLU are close in value, 0.57 and 0.6, respectively, the sigmoid-based model has a test accuracy of 0. Leaky-ReLU is evidently the strongest, and our interpretation is that it excels because it can capture the non-linear relationships in image classification problems [the core basis of the MNIST dataset] where sigmoid falters. Moreover, it is better than ReLU, which is a commentary on the dead neuron problem, as it would just give values of 0 for negative inputs, which could be relevant and contextual in image classification. As such, the results align with our expectations and reflect the model's use cases.

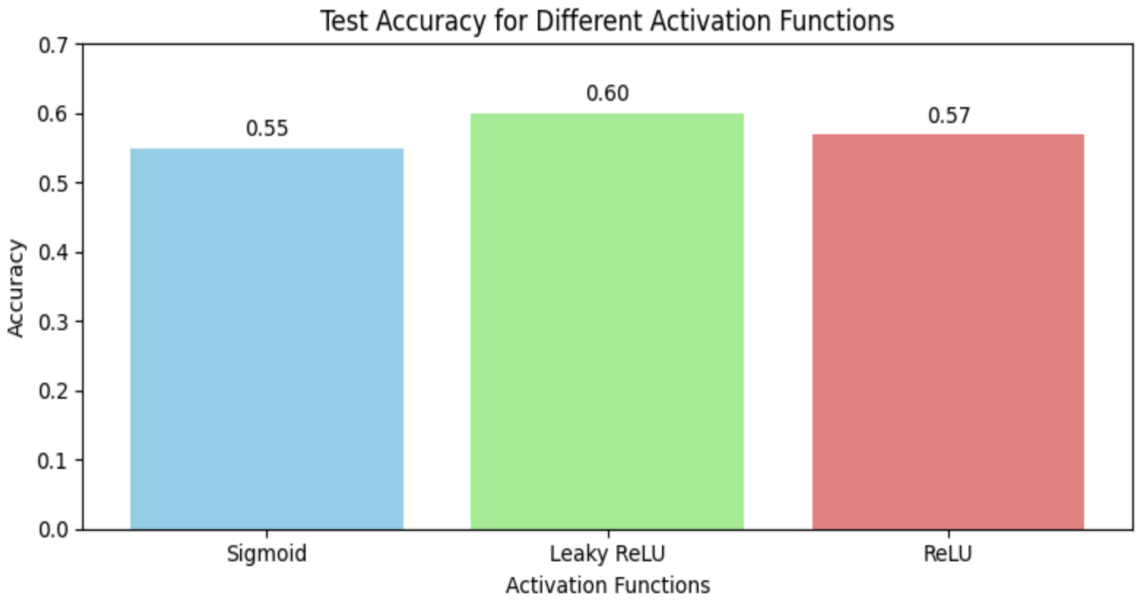


Figure 7: Test Accuracies of Different Activation Functions on MLP

L2 Regularization

L2 Regularization (Ridge) is a technique to prevent overfitting in MLPs (models broadly). L2 Reg effectively adds a penalty to the loss that shrinks the weights towards zero (close to); this allows for a less complex and simpler model. By penalizing larger weights, the model can train better on unseen data due to generalization. This reduces overfitting, building more robust models that can be deterred from noisy data. Additionally, it maintains the continuity of weights by not placing them as 0, which is beneficial for gradient-based optimizations. As we can see in Figure 8, L2 Reg does play a role in the model's test accuracy (the original model was ~55%), whereas now it is closer to ~60% with both the hyperparameters at 0.01 and 0.0001, whereas 0.001 has a nominal increase in the model's test accuracy. This indicates that L2 Reg has to be optimized; while generally, its regularization seems to indicate an increase in testing accuracy, fine-tuning the parameter is a crucial element in the benefit of the model. We can infer that L2 Reg is aiding the model in generalizing the data, boosting its ability to classify the images.

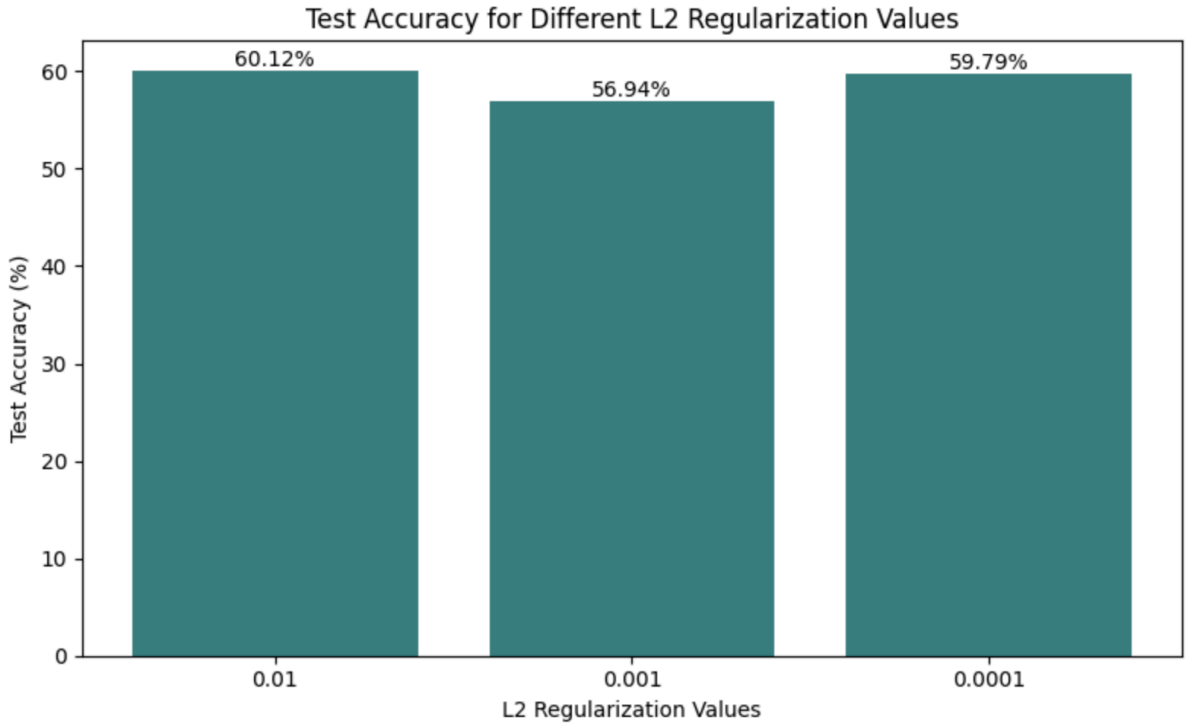


Figure 8: Test Accuracies of Different HyperParam Values for L2 Reg

CNN 3 Convolutional, 2-Fully Connected Layers, Experiment Hidden Units

CNNs are a class of deep neural networks actively leveraged in image visualization/classification problems. They are composed of convolutional layers, which apply filters to an input image, allowing for features (edges, shapes) to be extracted from images which might otherwise be overlooked. The filters have a constant shared weight, which reduces the trainable parameters and makes CNNs more efficient. To utilize a CNN model, we had to reshape our data beyond our standardization of the data thus the CNN could maintain the integrity of the spatial nuances of the images.

In our experiment, we permutated through the number of different nodes in the hidden layers. In a CNN, too few nodes don't capture complexity, and too many may lead to overfitting.

As seen in Figure 9, as the number of hidden units increases, the model's test accuracy rises as well, indicating it is able to capture complex features from the images. The highest accuracy of 91% was achieved with 256 units. The consistent increase in accuracy alongside node size indicates the model is not overfitting; moreover, there is enough data for the model to leverage to build its interpretation of complex nuances. We can generalize and observe that CNNs are magnitudes stronger than the designed MLPs with our MNIST dataset. This aligns with the CNN use case of tackling image-based problems.

```
Hidden units: 32 MODEL ACCURACY: 88.78973722457886%
Hidden units: 64 MODEL ACCURACY: 90.14222025871277%
Hidden units: 128 MODEL ACCURACY: 90.8951461315155%
Hidden units: 256 MODEL ACCURACY: 91.31343960762024%
```

Figure 9: Results of Test Accuracy CNN with Different Hidden Units

Optimal MLP Architecture

Our big takeaways from the experiments have been that two hidden layers, with a LeakyRelu layer activation, L2 regularized with a hyper pam value of 0.01, with 256 hidden nodes in their localized cyclos experiments, have had the strongest model test accuracy results. What we see from this test is the 'optimal model' architecture touches the 0.6 accuracy; across all our MLP, this seems to be an effective ceiling.

Experiments of CNN

Dropoff Rates

A dropout rate is a fractional representation of the nodes that are turned off during a training pass. The nodes have an equal probability of being temporarily removed from the layer of a CNN. The benefit of this randomized dropout is that it restricts the model from building a bias toward a specific node and forces it to generalize toward the broader domain of data. Allowing the model to be more robust and not fall prey to potential 'noise'. This is considered a hyperparameter of a CNN, which should be tuned based on the base model; a rate that is too high can result in underfitting, where due to the volume of 'off nodes', the model does not 'learn'.

As seen in Figure 10, there is a space of optimal dropout rate around 0.2%, as that is where the highest localized test accuracy of the model is observed. As the dropout rate increases, the model's test accuracy significantly drops, especially at 0.5. This aligns with our interpretation of dropout rates and additionally implies the model may not be prone to high overfitting and is rather learning from a regularized approach to the broader data, which seemingly isn't too complex.

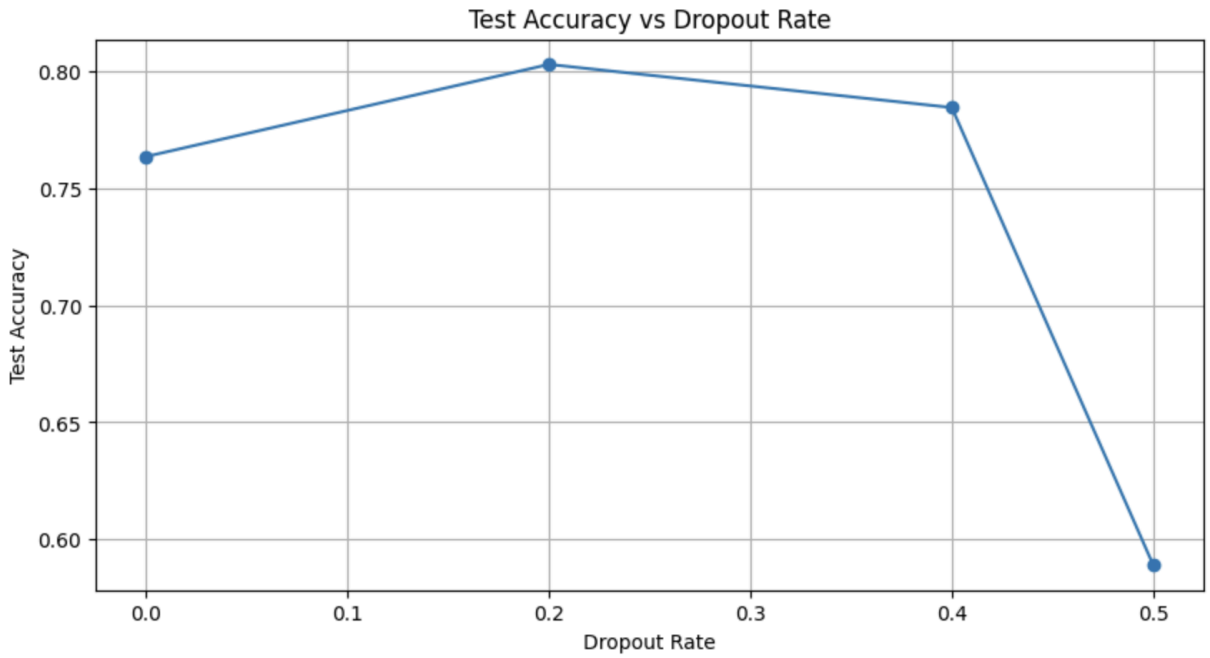


Figure 10: Test Accuracy As A Function of Dropoff Rate In CNN

Padding Types

Padding within CNNs is the practice of 'padding' an image with additional pixels before its processing in the CNN. This allows the model to maintain control of the output volume and build a robust architecture. 'Same' padding is designed to ensure the output has the same dimensions as the input, allowing for the preservation of spatial dimensions across convolution. However, it deviates from representing the true data. 'Valid' padding is when no padding is added, which can restrict the model to learn from tangible features of the images but may lead to information loss at the edges of an input image.

As seen in Figure 11, test accuracies for a given model with a constant filter amount seem to be within their respective accuracy space, regardless of the style of 'valid' or 'same' padding. We can infer from this that the model is indifferent to padding at its core, which suggests that the essential features of an image in our MNIST dataset are not at the broad edges but its core. The more significant parameter that is implicitly powerful in this context is the number of filters, where 32 seems to be the strongest, regardless of padding.

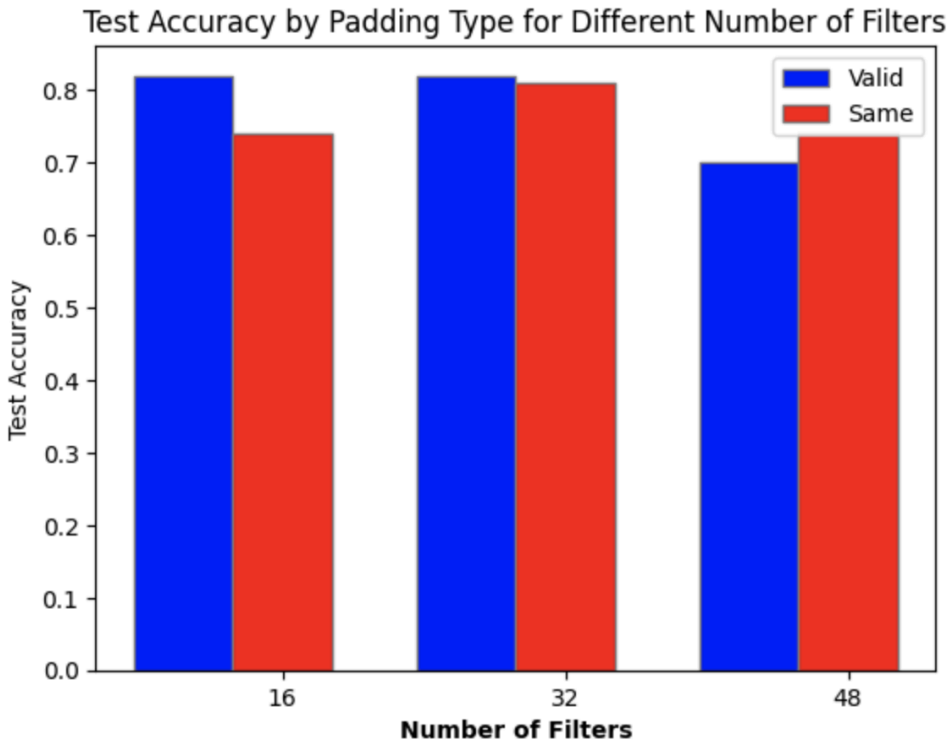


Figure 11: Test Accuracy As A Function of Paddying Type

Number of Filter, Filter Size, Strides Experimentation

In a CNN, filters are matrices of weights that are trained and used to extract features from a given input image. The values are learned through backpropagation, tuning towards valuable aspects of an image through gradient optimization. Filters facilitate a deep understanding of features in a CNN before being pushed down the layers, allowing them to evolve and grasp stronger insights into the features of the image. The number of filters is the amount used in any convolutional layer; more filters can capture a higher tier of information but are prone to overfitting and can be computationally costly. A filter size is the dimension of the matrix, which needs to be tuned in context to the dataset so it's abstracting the right features and not broadly global.

A stride is the magnitude of pixels in the image that a filter moves over, allowing for faster computation but reducing the spatial dimension of an image, which can have conflicting effects on the model.

As seen in Figure 12, with a CNN using 16 filters, the strongest accuracy (0.84) is achieved with a model using a 4x4 filter size and a 1x1 stride. When the stride increases, accuracy drops significantly; this trend is constant regardless of filter size. This suggests that a model with a larger stride is missing spatial information it relies on to classify images, effectively skipping over pixels that are needed to build understanding.

In Figure 13, with a CNN using 32 filters, the strongest model again shows an accuracy of 0.81 with a 4x4 filter and a 1x1 stride. Here, an increase in filter size is not a driving change in accuracy, whereas an increase in stride size still reduces the accuracy of a model given everything else is constant.

Finally, as seen in Figure 14, with a model using 48 filters, the peak accuracy is still achieved with a 4x4 filter size and 1x1 stride, but across configurations, accuracy change is nominal. This suggests that the exhaustive 48 filters can capture general nuances of the images and maintain certain performance, where hyper-parameter tweaking does not have a large effect.

The evident insight is that a larger filter size with a small stride is the strongest configuration across all models, with larger strides and small filter sizes generally reducing accuracy. The increase in the number of filters seems to stabilize accuracy; however, the strongest configuration is a model with a small number of filters (16). We can infer that filters can act as a buffer against accuracy drops from other hyperparameters, but a larger number of filters is not indicative of a better model. The insight to gain on the dataset is that spatial data and capturing features of the image (edges, shapes) are of high value to the CNN solution for the dataset's problem.

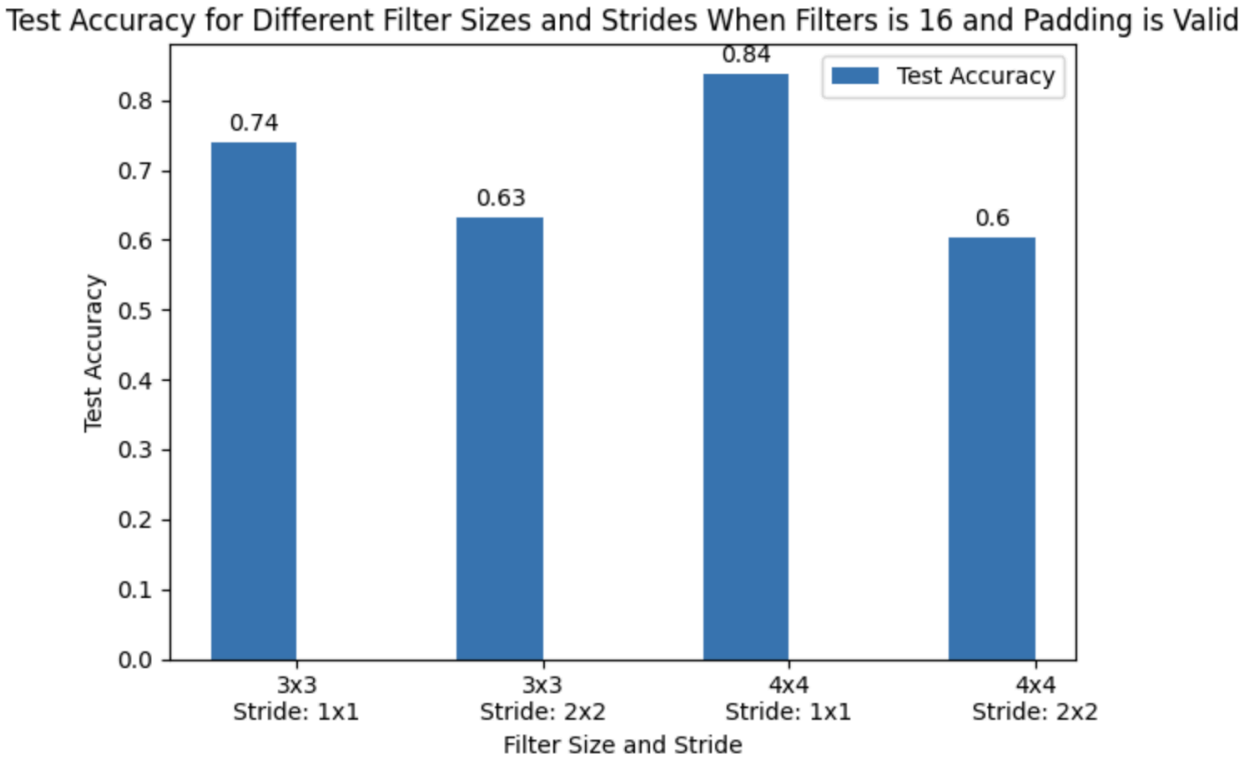


Figure 12: Test Accuracy as a Function of Stride and Filter Size In CNN with 16 Filters

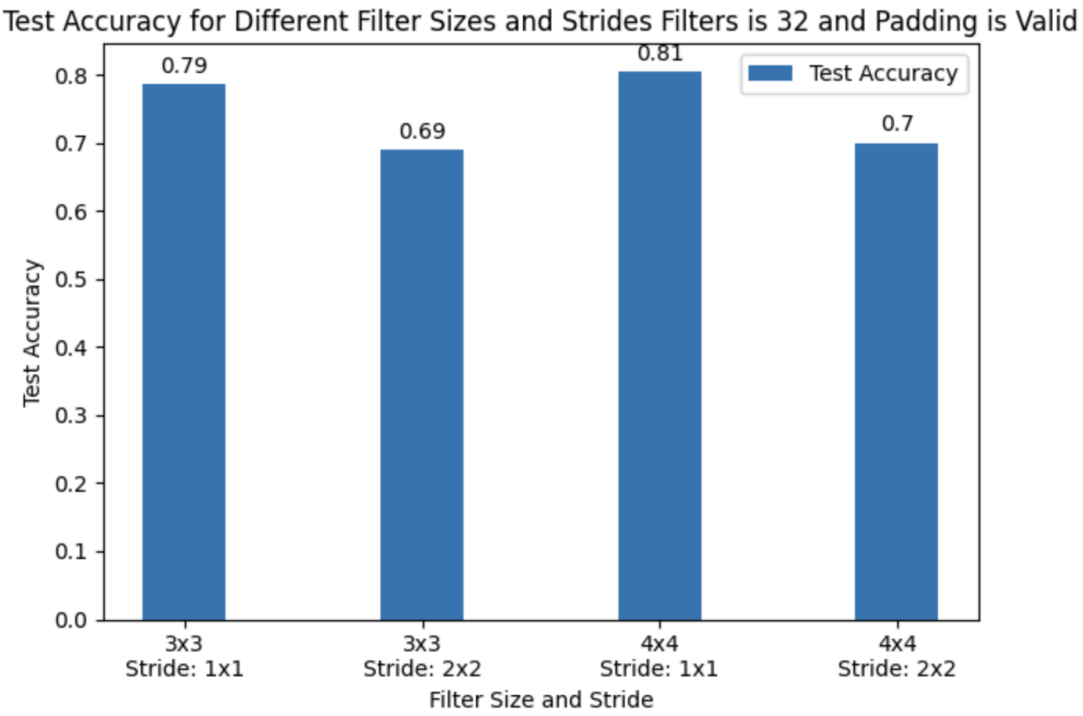


Figure 13: Test Accuracy as a Function of Stride and Filter Size In CNN with 32 Filters

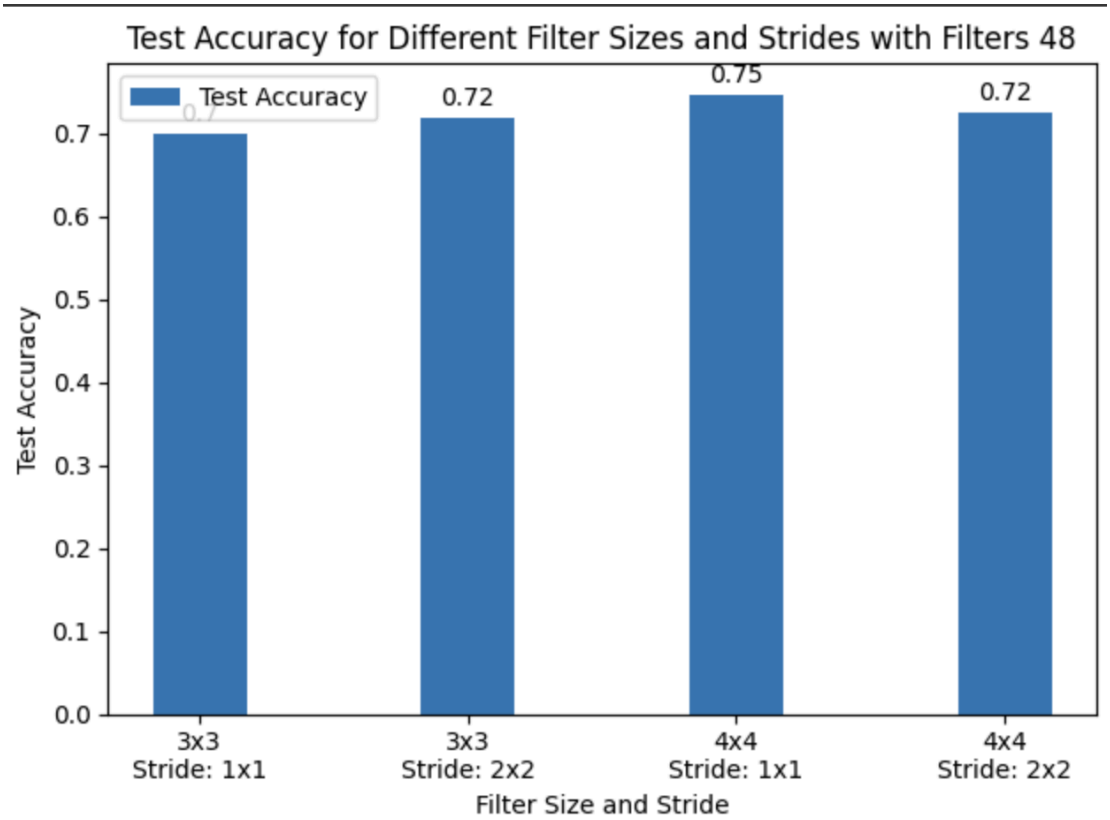


Figure 14: Test Accuracy as a Function of Stride and Filter Size In CNN with 32 Filters

Discussion & Conclusion

Our experiments have deepened our understanding of our data and also aligned with our general interpretation of MLPs, CNNs, and their respective hyperparameters.

The biggest takeaway is that CNNs are more effective at the classification of images than MLPs. The convolutional structure of CNNs, combined with an optimal dropout rate, has proven to create a general model capable of high accuracy on test images without falling prey to overfitting or losing critical information to underfitting. Additionally, we observed that the type of padding in CNNs did not significantly impact our model. This implicitly indicates that the key spatial information in our MNIST images is not at the border, which padding might help to elucidate further. Moreover, it is evident that larger filter sizes with low strides are the best configuration to capture the most relevant information from images in this dataset; high stride values are detrimental and cause the model to skip essential information about the image. While a higher number of filters tends to stabilize accuracy across configurations, it is not the optimal model for achieving the highest accuracy, which appears to be around 32 filters.

For MLPs, we clearly see the benefit of their ability to analyze non-linear relationships, as models with the sigmoid activation function and 0-hidden layer are relatively insufficient to classify the test images. In contrast, a higher set of hidden layers/nodes and a non-linear activation function played a prevalent role in boosting model accuracy in classifying images. Regularization further helps the model generalize data that might be unseen in initial runs, but it needs to be hyperparameter-tuned for optimal performance for the underlying model. Leaky-ReLU additionally mitigated the dying gradient problem in ReLU and played a role in bringing weight to negative input values, which are prevalent in image classification tasks. A 1-hidden layer MLP might be more stable across different numbers of hidden nodes, but at least for MNIST, a double hidden layer model with 256 nodes is best able to capture information about the dataset.

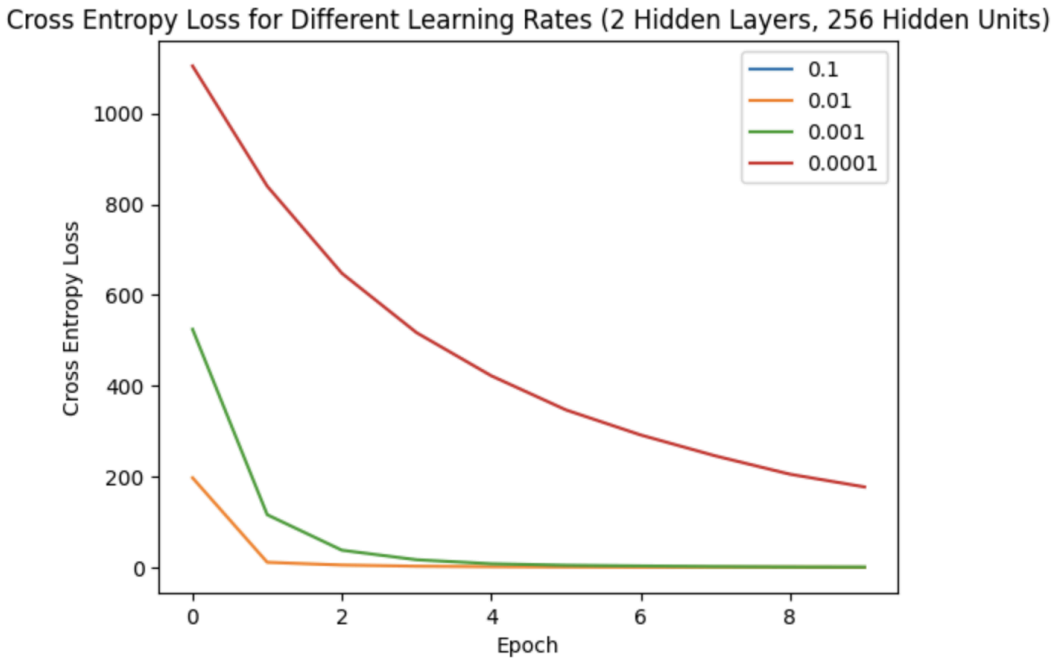
Next steps would include exploring methods to optimize the MLP to reach accuracy closer to that of a CNN; potentially by exploring other activation functions and an L1 regularization technique. There is an opportunity to build a CNN with our learned understanding of filter size and stride size, and approximately 32 filters, to see how high we can push accuracy. This would require exhaustive compute and many hyperparameter optimizations. We should additionally explore data augmentation techniques to allow for further experimentation on the data and potentially enable the application of more complex models without overfitting.

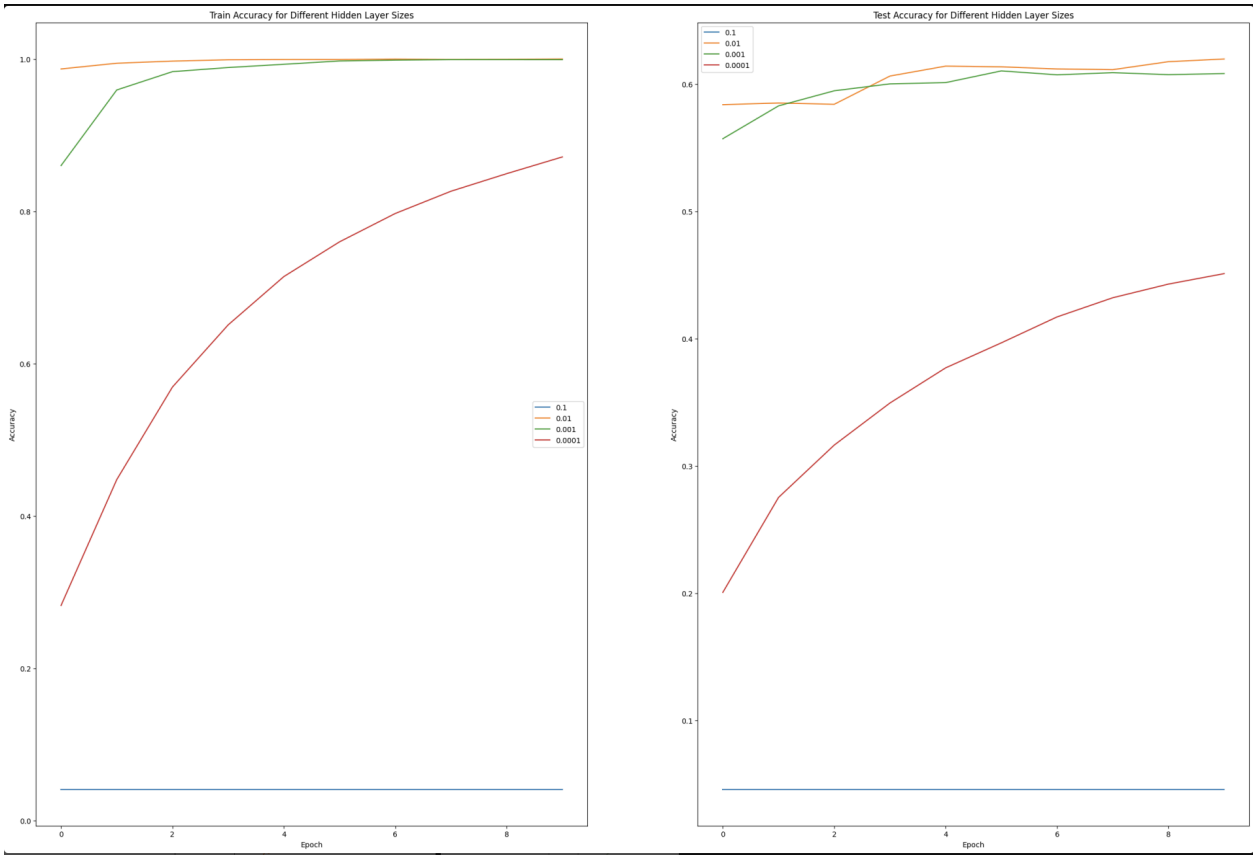
Statements of Contribution

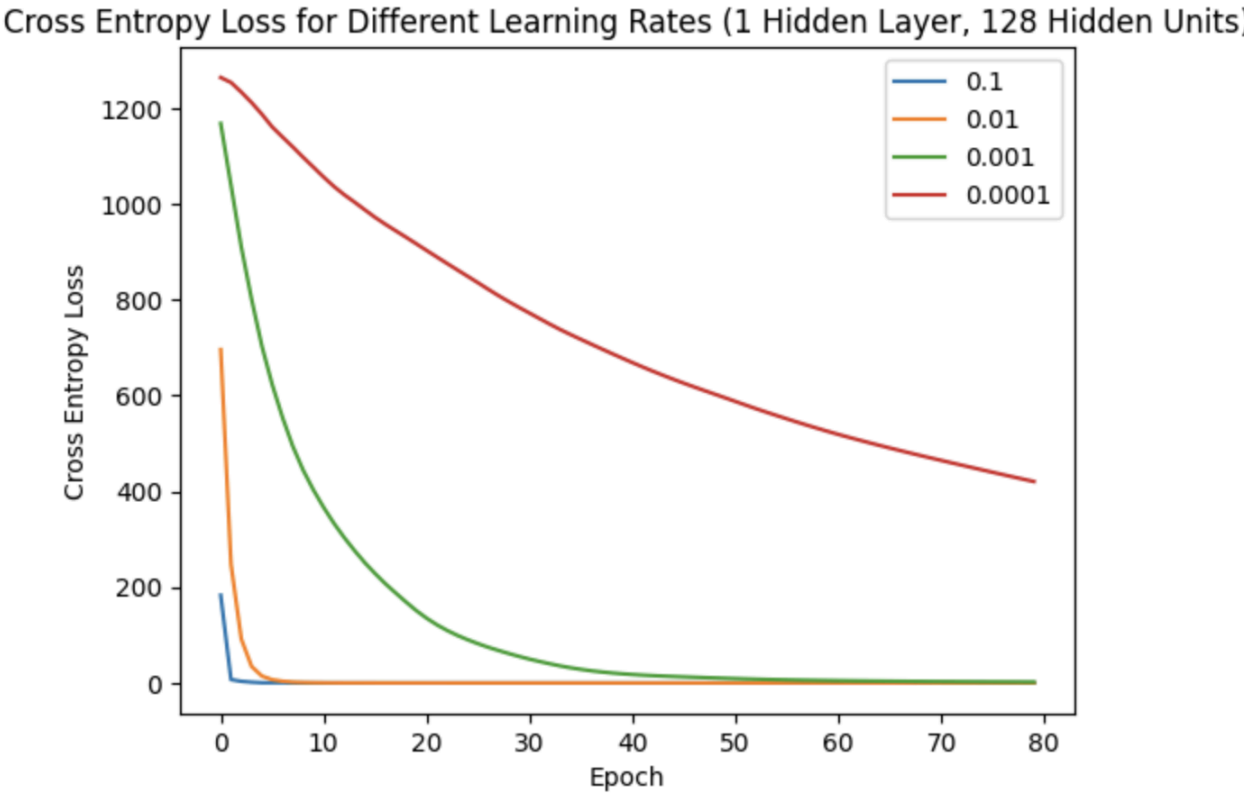
Inayat and Emile equally split and managed the the cleaning of the dataset, building the core MLP class accounted for batch-based SGD and the mandatory experiments of MLP as well as additional experiments on CNN. Moreover, they split the finalization of the documents. Alex was not able to contribute to the project as he had family emergencies and should be contact with the Prof regarding circumstances.

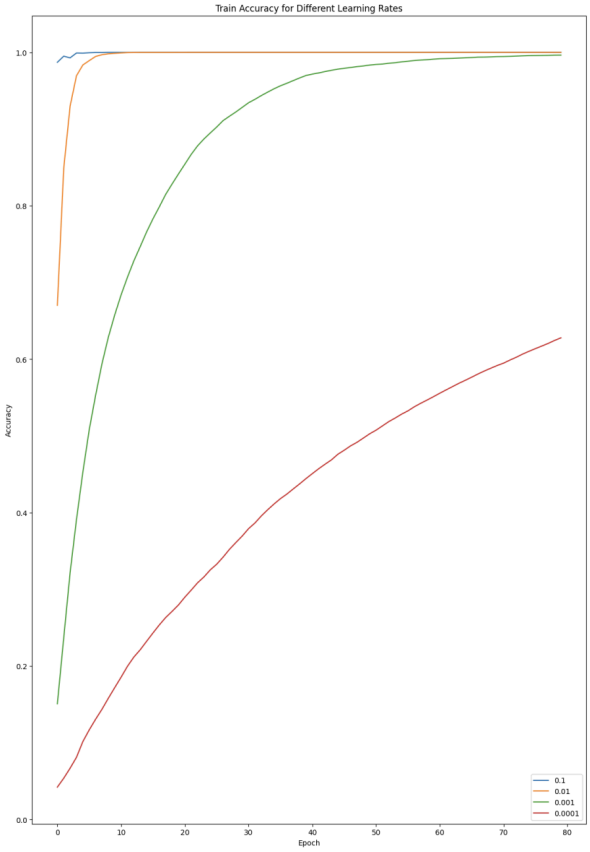
Appendix Images For Reference of Hyperparam Decisions

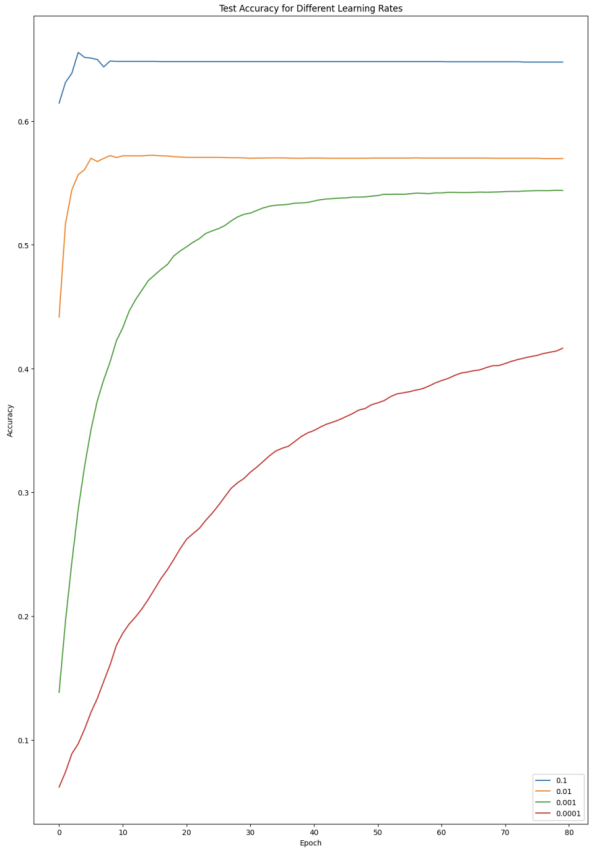
Learning Rates In MLP [Loss]



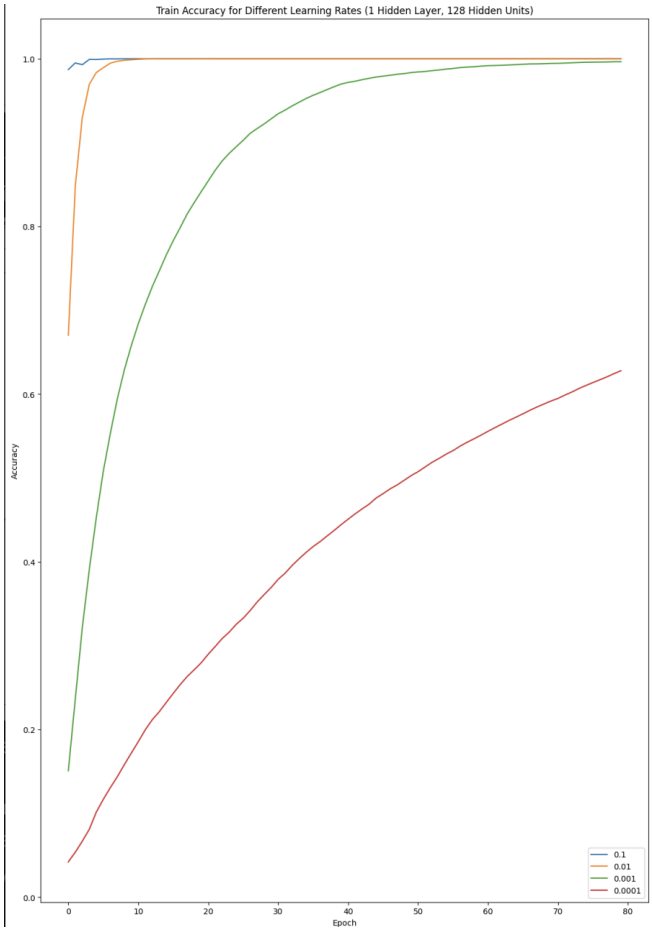


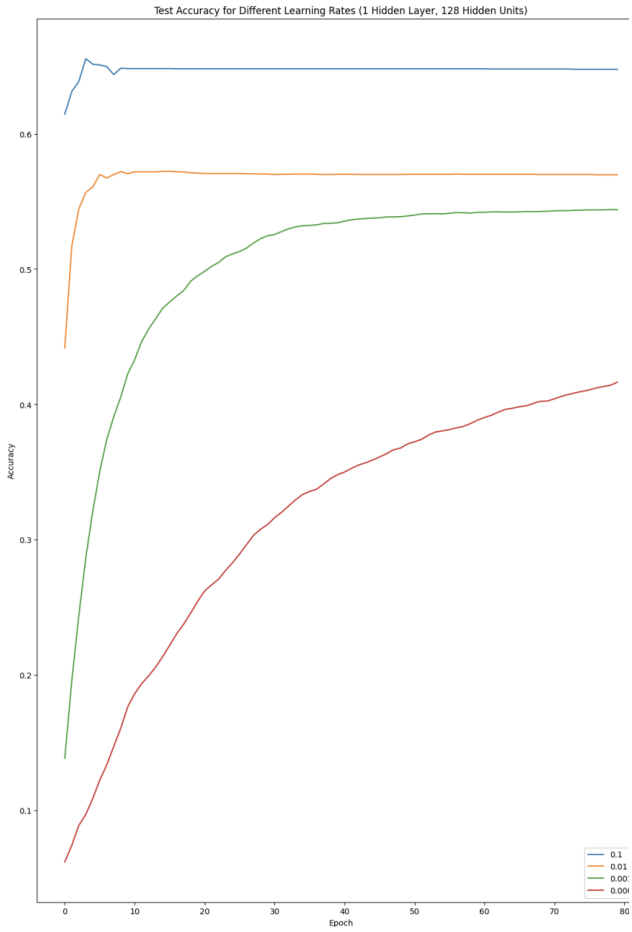

Cross Entropy Loss for Different Learning Rates (1 Hidden Layer, 128 Hidden Units)
Cross Entropy Loss
Epoch
0.1
0.01
0.001
0.0001
0
200
400
600
800
1000
1200
0
10
20
30
40
50
60
70
80

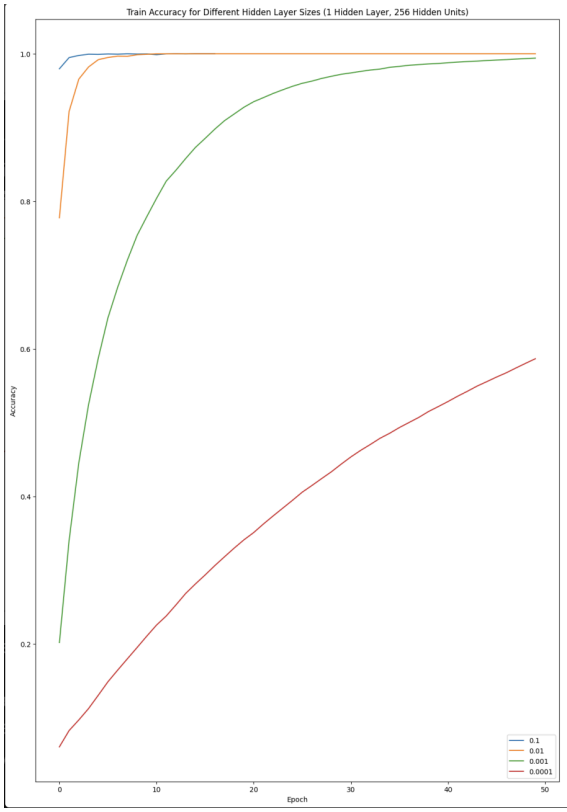

Train Accuracy for Different Learning Rates
Accuracy
Epoch
0.1
0.01
0.001
0.0001

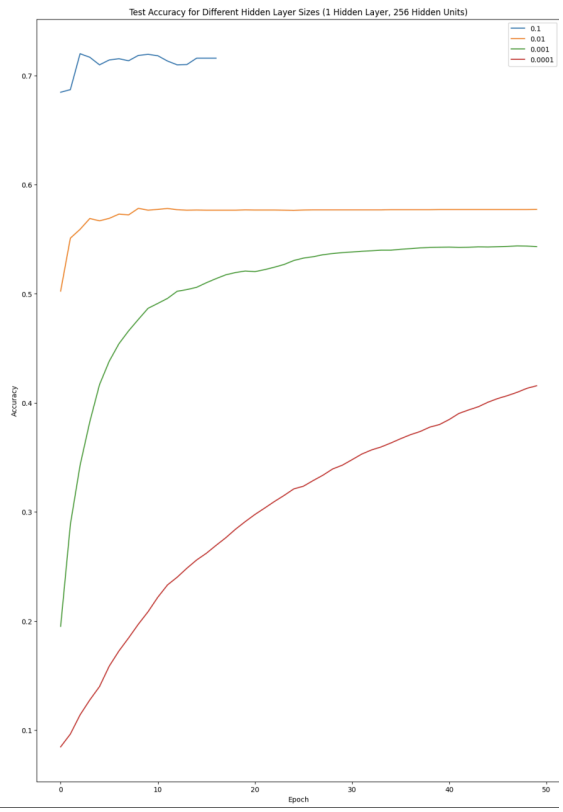

Test Accuracy for Different Learning Rates
Accuracy
Epoch
0.1
0.01
0.001
0.0001

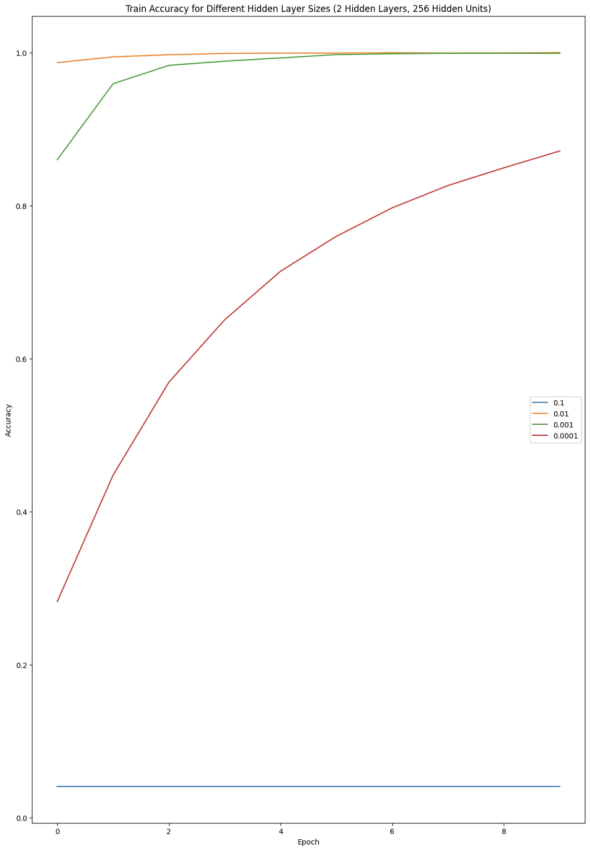
Learning Rates In MLP [Accuracy]

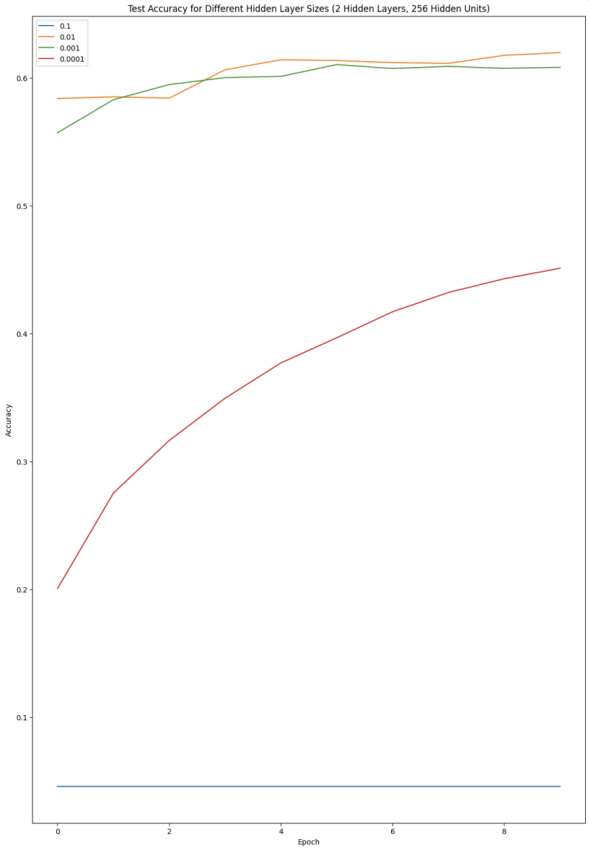


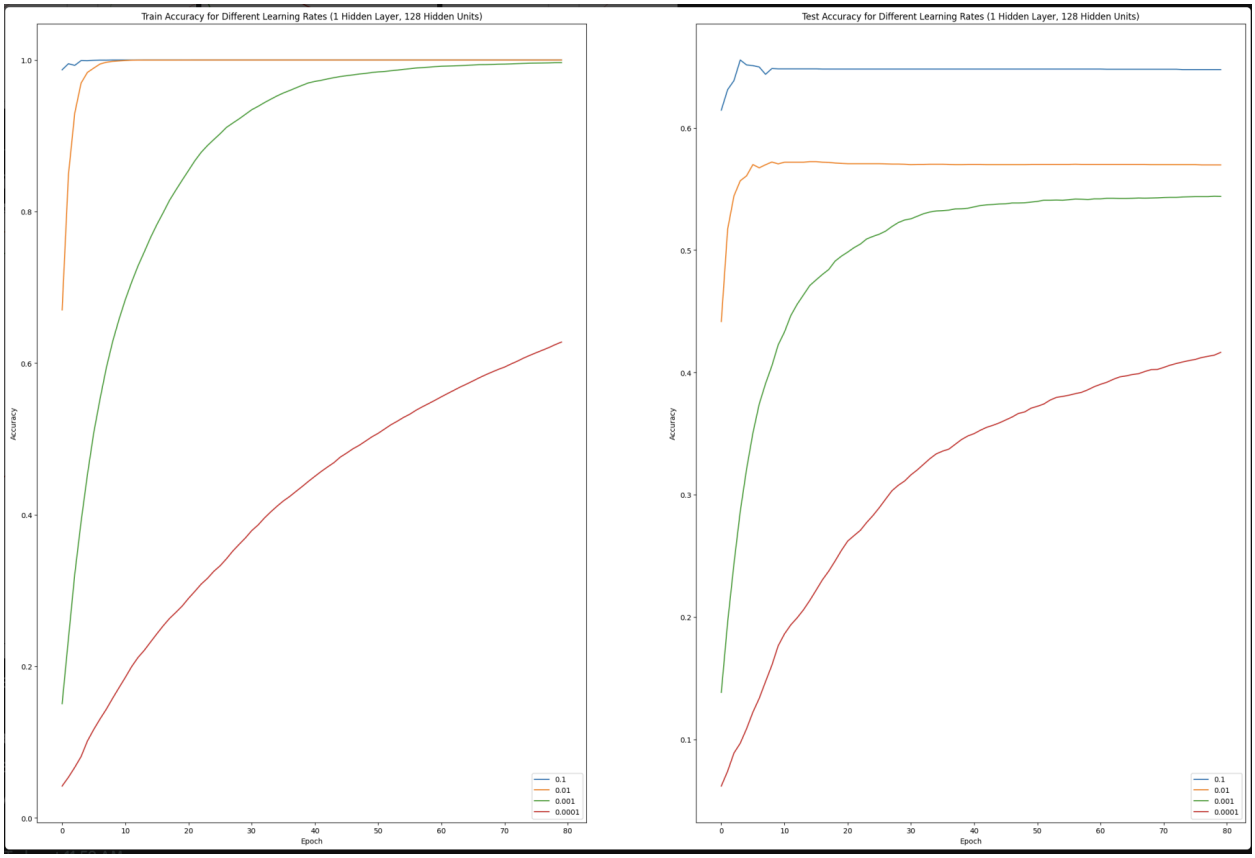



Train Accuracy for Different Hidden Layer Sizes (1 Hidden Layer, 256 Hidden Units)
Accuracy
Epoch
0.1
0.01
0.001
0.0001


Test Accuracy for Different Hidden Layer Sizes (1 Hidden Layer, 256 Hidden Units)
Accuracy
Epoch
0.1
0.01
0.001
0.0001


Train Accuracy for Different Hidden Layer Sizes (2 Hidden Layers, 256 Hidden Units)
Accuracy
Epoch
0.1
0.01
0.001
0.0001


Test Accuracy for Different Hidden Layer Sizes (2 Hidden Layers, 256 Hidden Units)
Accuracy
Epoch
0.1
0.01
0.001
0.0001


Train Accuracy for Different Learning Rates (1 Hidden Layer, 128 Hidden Units)
Accuracy
Epoch
0.1
0.01
0.001
0.0001
Test Accuracy for Different Learning Rates (1 Hidden Layer, 128 Hidden Units)
Accuracy
Epoch
0.1
0.01
0.001
0.0001