Assignment 3: Multilayer Perceptron

Anson Ng Howard Qin Aybuke Ekiz

April 8, 2024

Abstract

In machine learning, simple regression models such as linear and logistic regressors have may useful applications. However, they struggle to perform on non-linearly separable problems. Neural networks, such as MLP, can accomplish these tasks with greater accuracy. This project involves the implementation of a MLP from scratch, and the performing of experiments to optimize the choice of hyperparameters and architecture, as well as a comparative analysis between the performance of the MLP and that of the CNN implemented in machine learning libraries. Our findings show that using small learning rate, Leaky ReLU, and large number of hidden units all contribute to the improvement of MLP performance. Further, we found that the CNN implemented by the libraries outperform our own MLP in virtually every aspect, both training faster and achieving higher test accuracy.

Introduction

The main objective of this project was to implement a MLP from scratch and train it with the provided serialized sign language image data, and find how different hyperparameter settings affect its performance.

Our findings show that a higher number of hidden units and a higher number of hidden layers correlate to higher training and testing accuracy but also more training time. Additionally, our research delved into the effects of varying learning rates for different MLPs with various hidden layers and hidden units on their accuracy. While smaller learning rates generally improve the models' accuracy, sometimes the improvement is marginal yet the increase in convergence time is significant. We also trained a CNN with 3 fully connected and 2 convoluted layers on the same sign language data set, and found that more hidden units and smaller learning rates also improve model accuracy at the cost of training time. Further testing revealed that using max pooling and increasing the number of filters both increase the CNN's accuracy, while other hyperparameters such as kernel size, padding and stride have more complex influences on the accuracy.

Dataset

To train our MLP, the Sign Language MSINT dataset was used, which conveniently contained csv files detailing the greyscale values of each pixel in each image for both the testing and training sets. After loading, we centered, normalized and vectorized the data accordingly, in order to speed up gradient descent convergence and ensure all features have the same importance (although greyscale values have a strict range of 0-255 already, we normalized nonetheless just in case). We further took the liberty of taking half of the testing set as a validation set.

Results

Experiment 1

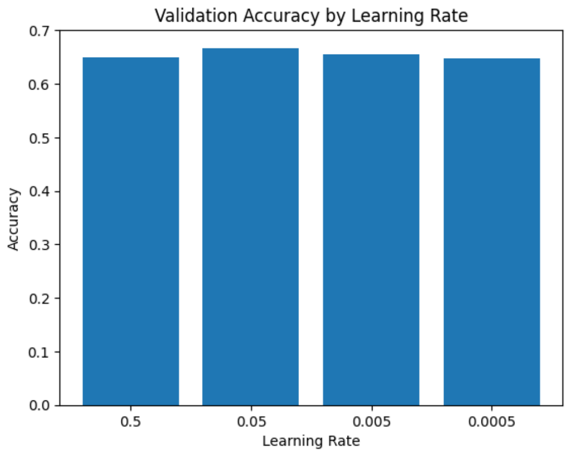

Figure 1: 0-hidden layer MLP accuracy with no hidden layer with various learning rates

Figure 1 displays the validation accuracy of the 0-hidden layer MLP at different learning rates in a bar chart. Although learning rate of 0.05 shows some improvement over 0.5, the accuracy is universally very low, at about 0.63 to 0.67. Since the number of iterations is fixed, it is possible that with a learning rate below 0.05, the model did not have enough time to converge properly.

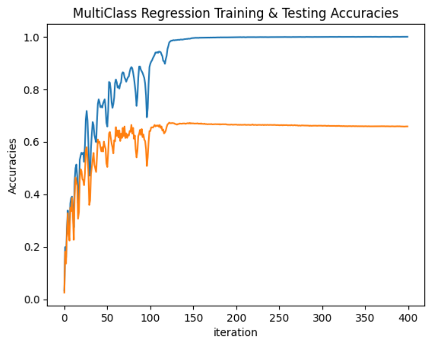

Figure 2: 0-hidden layer MLP training and testing accuracy plot against number of iterations

Figure 2 displays the 0-hidden layer MLP's training and testing accuracy plotted against the number of iterations, with the best learning rate of 0.05 as determined previously. After a period of sporadic growth, both plateau after about 130 iterations, with a near-perfect training accuracy and a subpar testing accuracy of 0.658.

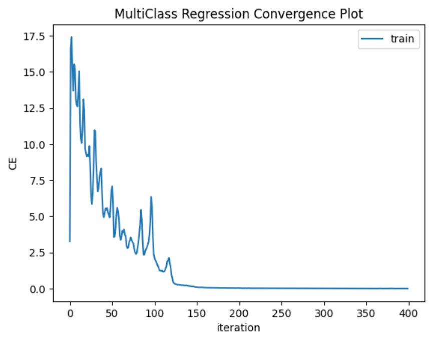

Figure 3: 0-hidden layer MLP's CE convergence plot

As expected from the plateauing of accuracy, after about 130 iterations, the cross entropy loss goes down to negligible values. The sporadic rise and fall in both the accuracy and cross entropy loss indicate that the 0-layer model is far too simple to capture the complexities within each image.

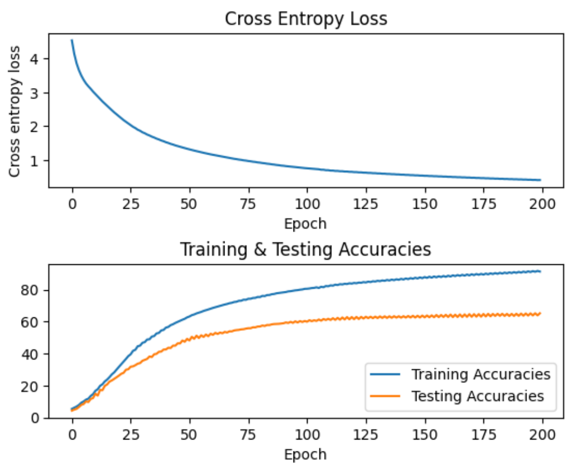

Figure 4: 1-hidden layer MLP's CE convergence & accuracy plot, 32 hidden units

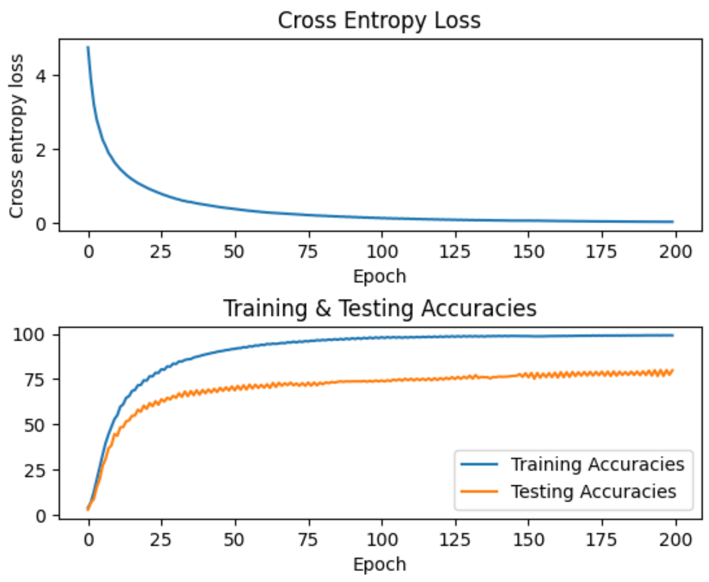

Figure 5: 1-hidden layer MLP's CE convergence & accuracy plot, 256 hidden units

From figures 4 and 5, it is observed that with 1 hidden layer, the cross entropy loss converges much more smoothly as is usually expected. Further, the cross entropy loss converges faster with more hidden units and achieve better accuracy faster.

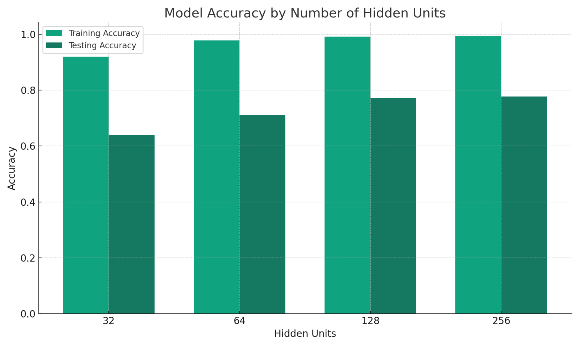

Figure 6: 1-hidden layer MLP's training and testing accuracy with various numbers of hidden units

In figure 6, we see the expected pattern where increasing the number of hidden units lead to an increase in accuracy, as the model becomes more capable of capturing more complexity. The testing accuracy kept improving with more hidden units. However, although increasing the hidden units from 128 to 256 resulted in a marginal improvement, the increase in training and prediction time is significant. In some practical applications, it may be more desirable to save computation resources at the cost of a trivial decrease in model quality.

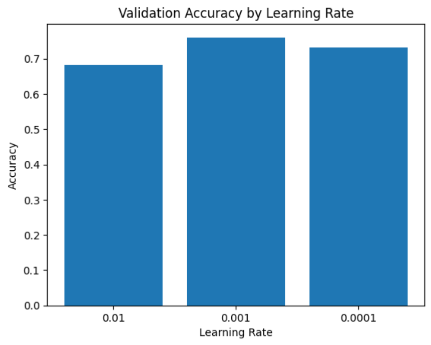

Figure 7: 1-hidden layer MLP validation accuracy at different learning rates

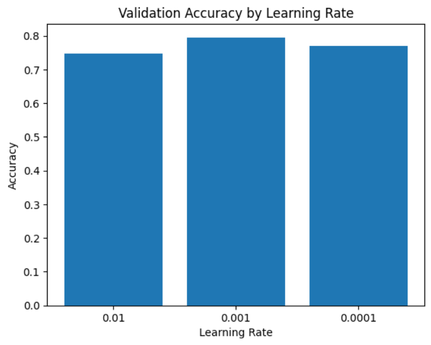

Figure 8: 2-hidden layer MLP validation accuracy at different learning rates

Figure 7 and 8 show the validation accuracy of the 1-layer and 2-layer variants of the MLP at different learning rates. 0.01 is far too coarse and results in the weights updating too aggressively, while 0.0001 is too low for the models to properly converge within the max iterations. For this reason, 0.001 is chosen as the learning rate.

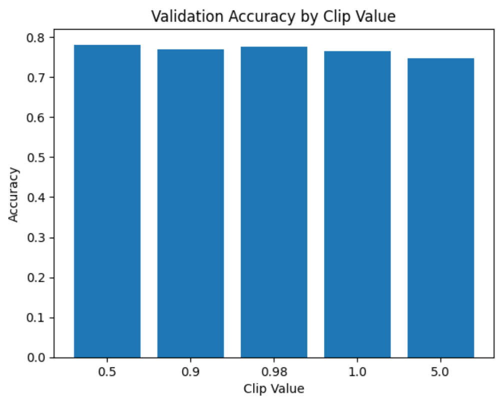

Figure 9: 2-hidden layer MLP validation accuracy at different clip values

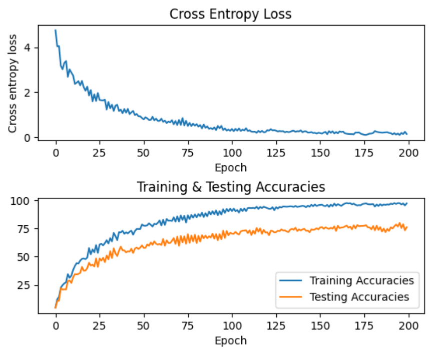

Figure 10: 2-hidden layer MLP convergence and accuracy plots at clip value 5.0

From figure 9, we see that a clip value of 0.5 provides the best accuracy, although not by much compared to the others within the 0.5-1.0 range. However, with a clip value of 5.0, the accuracy starts to noticeably drop. Figure 10 explains this phenomenon: with the gradient less clipped, the updates weights can become too aggressive, thus causing the oscillations shown in the figure which was not present before when the clip value was 1.0 as default.

Thus for experiment 1, we devised a best model consisting of 2 hidden layers, 256 hidden units, with learning rate of 0.001 and clip value of 0.5, achieving a testing accuracy of 0.786.

Experiment 2

For experiment 1, we were using MLPs with ReLU activations. However, there might be better choices of activation functions that further improve performance and training time. To find a potentially better solution, we made 2 MLP variants of the best MLP from experiment 1, one using sigmoid layers and another using leaky ReLU layers.

While testing the sigmoid MLP, we found that its CE loss convergence plots tend to take a shallower descent than that of ReLU or leaky ReLU MLPs, which could be due to the fact that the output of the sigmoid function is not 0-centered, and that the derivative of the sigmoid function is $\sigma'(x) = \sigma(x)(1 - \sigma(x))$, and when $\sigma(x)$ becomes too close to 0 or 1, the derivative becomes too small. Through backpropagation, vanishing gradient could occur.

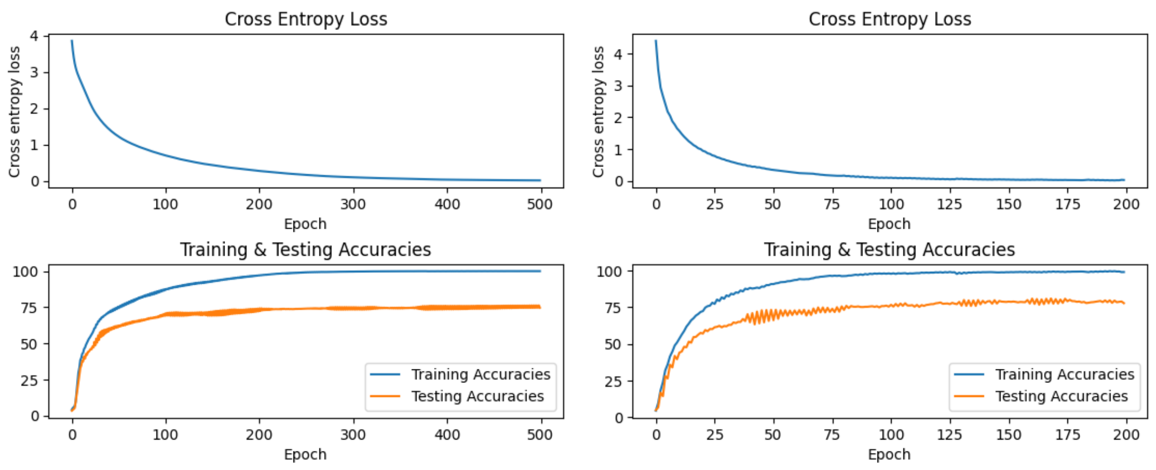

Figure 11: Sigmoid and leaky ReLU variants MLPs convergence and accuracy plots at learning rate 0.001

Figures 11 illustrates this point perfectly. With all other hyperparameters held to be the same, the LeakyReLU MLP took less than half the amount of iterations of what the sigmoid MLP did (notice the difference in X axis in figures 11), yet achieved a better validation accuracy of 0.783 compared to the sigmoid MLP's 0.762. Between ReLU and leaky ReLU, there is no significant difference in convergence time.

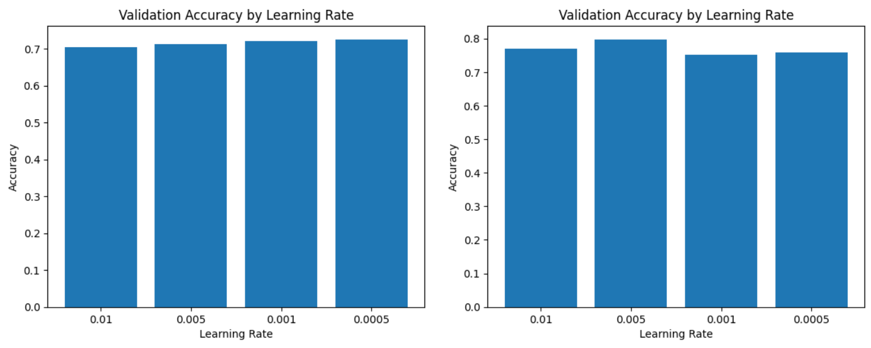

Figure 12: Validation accuracy by learning rate for sigmoid and leaky MLPs

In the end, out of the 3 model variants with different activations and optimal hyperparameters (all 256 hidden units and 2 hidden layers, 0.0005 learning rate for sigmoid, 0.001 for ReLU, 0.005 for leaky ReLU), the leaky ReLU variant had the best test accuracy of 0.807, ReLU at 0.786, and sigmoid at 0.754. This is as expected, since the 2-hidden layer architecture adds depth and thus increases the risk of dying ReLU, which leaky ReLU would address. Sigmoid performed the worst, not only taking a larger epoch to train but also achieving a lower accuracy.

Experiment 3

Having a model with depth and many hidden units usually helps to capture the complexities within the input features, however doing this too much might lead to overfitting, where the model effectively learns the noises within the training data, thus making its weights less applicable to general predictions. To solve this problem, L2 regularization can be used to penalize large weights and prevent overfitting.

A λ value that is too large would penalize the weights to essentially nothing, destroying the complexities we aimed to capture, while a value that is too small would result in vanishingly small correction effects. To find out which λ value is best, we tried different values within 1, 0.5, 0.1, 0.05, 0.001 on a 2-hidden layer ReLU MLP with only 128 hidden units to save computation time. We assume that the best λ value for this model would be applicable to the full 256 hidden units model as well.

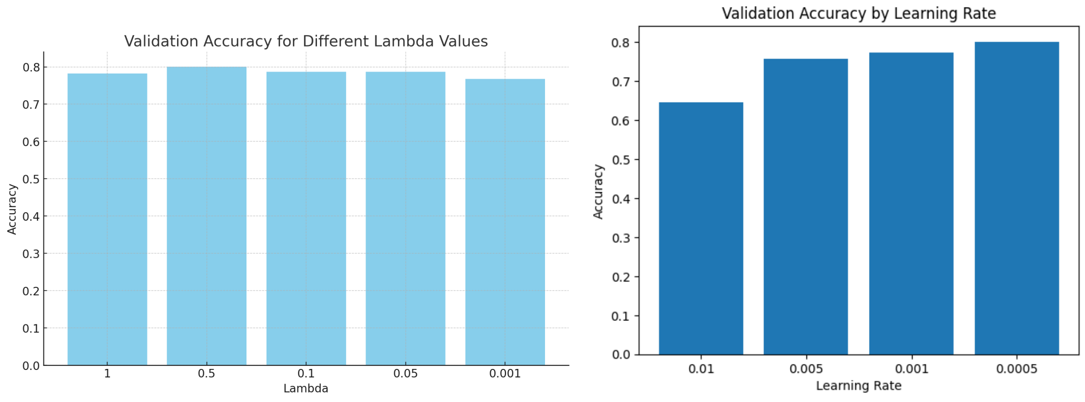

Figure 13: Validation accuracy for different λ values + validation accuracy by learning rate for model with λ of 0.5

From figure 13, since accuracy consistently dropped when λ is less than 0.5, we can deduce that some amount of overfitting did occur within the model, and L2 regularization was able to correct this to some degree. However, as λ becomes smaller, the correction effect became less noticeable, hence the drop in accuracy. Further, a λ of 1 is presumably too large and thus erasing some of the complexities that should have been captured otherwise.

Experiment 4

While MLP can be used for image classification tasks, CNN should in general be more efficient and more accurate at this. To test this claim, we trained a CNN model implemented by the tensorflow library to compare its accuracy and training time to that of our own MLP.

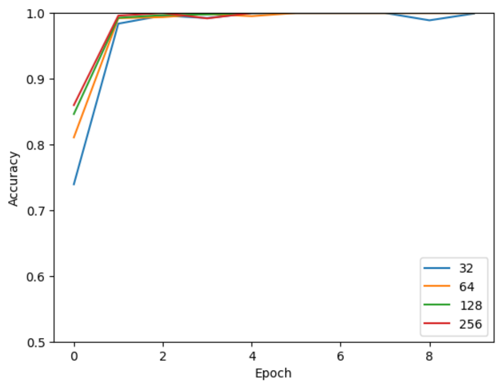

Figure 14: Validation accuracy by epoch of CNNs with different number of hidden units

In CNNs, we observed the same pattern as in MLPs: more hidden units lead to better accuracy and more consistent convergence. Comparing these validation accuracy measurements with that of our best MLP's clearly demonstrates CNN's superiority in image classification: Not only did even the worst CNN on the plot exceed our best MLP in validation accuracy, it did so with only 4% the epoch size. Although we did expect CNN to achieve better accuracy with smaller epoch, it is still surprising that CNN can outperform MLP so much in every aspect, particularly in epoch size.

Experiment 5

Summarizing the best hyperparameters and architecture we have found for our MLPs from experiments 1-3, we decided to try the following: 2 layers, hidden unit size of 256, leaky ReLU activation layer, L2 regularization with lambda value 0.5, clip value of 0.9, learning rate of 0.0005.

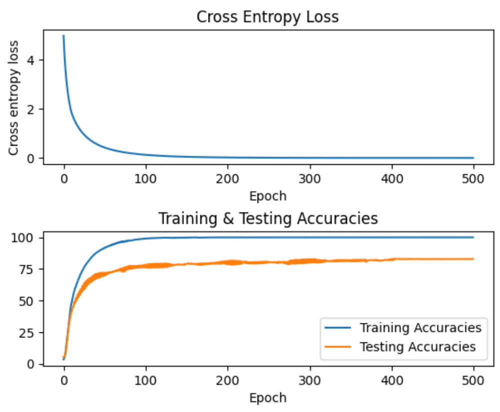

Figure 15: Convergence and accuracy plots for expected best MLP

However, the results are not as expected. This MLP achieved an accuracy of 0.788 and outperformed by a model trained as a part of experiment 3, which achieved an accuracy of 0.834, which strangely only comprised of 128 hidden units. Since all evidence points towards having more hidden units lead to better accuracy, it might have been a coincidence. To investigate, we took a generic 2-layer 256 hidden unit model with L2 and gradually altered its layer composition to observe any changes in accuracy.

After switching back to ReLU, we first attempted increasing the hidden unit count to 512, which only increased the accuracy from 0.807 to 0.809. We then increased the number of layers to 3 and achieved an accuracy of 0.820. Afterwards, we tried MLPs with hybrid layers, where one layer uses ReLU while another uses leaky ReLU, however these attempts still failed to exceed an accuracy of 0.820.

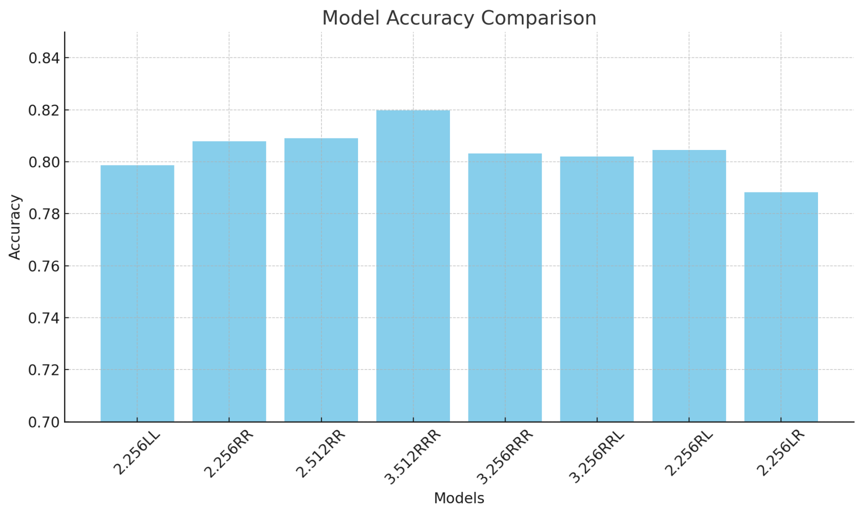

Figure 16: Accuracy of different models by architecture

In the graph above, “L” stands for Leaky ReLU, “R” stands for ReLU. This graph shows the accuracy of models with different architectures. Based on the results above, we chose 3 hidden ReLU layers, 512 hidden units, learning rate 0.0005, $\lambda = 0.5$, clip value of 1.0 as the best settings for our MLP implementation.

Experiment extra

On top of the required experiments on MLP, we further explored how different hyperparameters and architecture choices, such as max pooling, number of filters, kernel size, stride, padding, and activation function affect the performance of CNNs.

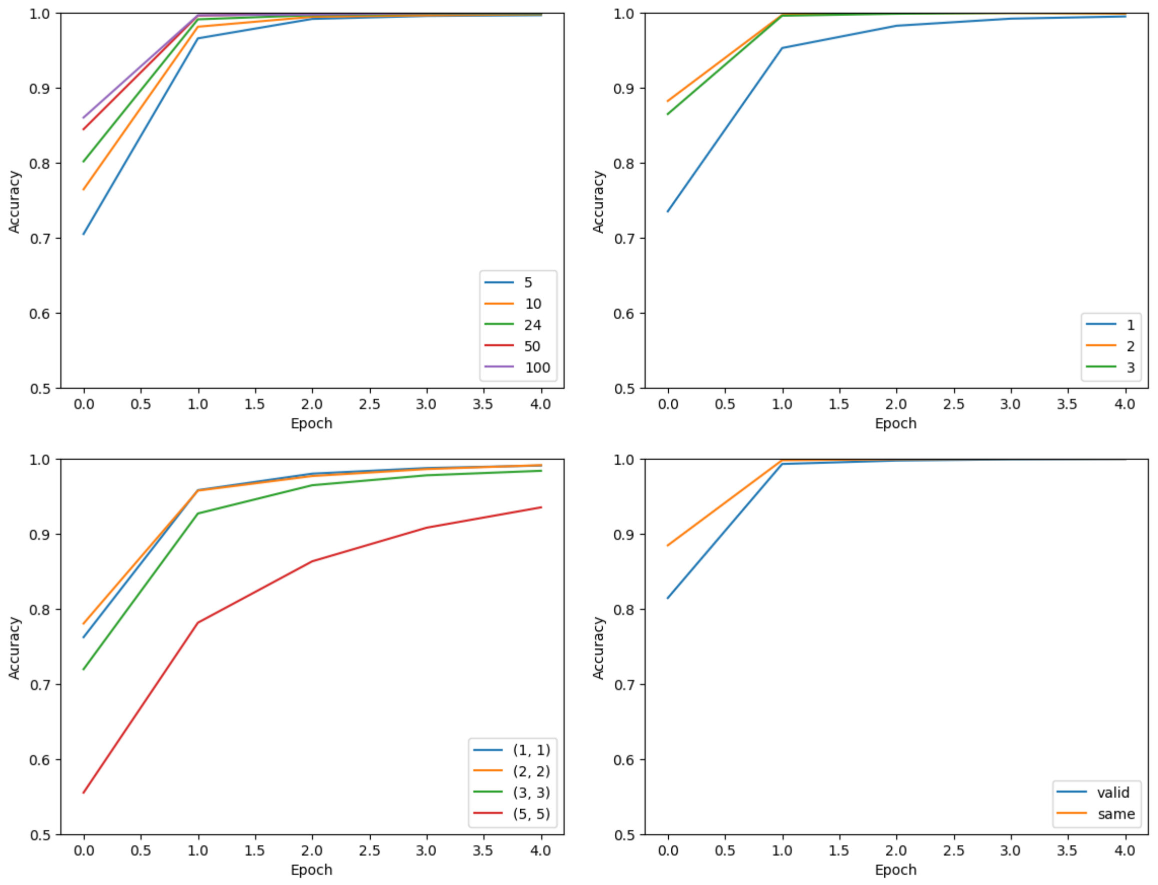

Figure 17: CNN accuracy plot by epoch with different numbers of filters, kernel sizes, strides and padding

We have found that increasing the number of filters tend to increase accuracy and decrease convergence time. A kernel size of 2 seems to be optimal, while increasing it to 3 actually slightly degrades performance. Larger strides, in this case, actually decreases accuracy, which could be due to the fact that the input images are only 28*28 in size, and large strides remove too much of the details. For padding, we found that same padding outperforms valid padding. This might, again, be due to the small size of input images, and valid padding reducing the spatial dimensions of the feature maps to a point where further convolutional operations are not possible without making the feature maps disappear.

Discussion and Conclusion

In conclusion, we found that for both MLP and CNN, an increase in depth and number of hidden units generally result in better accuracy, at the cost of training time. We further found that L2 regularization helps to mitigate overfitting when the λ value is just right. When compared, CNNs outperform MLPs in not only accuracy but also required training epoch due to their specialized architecture which are better tuned for image classification, where as MLPs have a more generalist architecture. In attempting to construct an MLP that performs as good as possible, we found that various hyperparameters, when their best values are determined separately, might not necessarily work together to make the model better when combined, and that accuracy itself may not be the best metric for model performance.

For future investigation, it would be worthwhile to explore other activation functions. For example, since we concluded that sigmoid could be subpar due to it not being 0-centered, it would be interesting to see how hyperbolic tangent, which has a similar shape to sigmoid but is 0-centered, would perform. It would also be interesting to attempt dynamic adjustment to lambda and learning rate during training to get best results in the end. We can also further explore how MLPs with hybrid activation layers might behave, and how using different hybrid permutations affect their ideal hyperparameters.

Statement of Contribution

Anson: MLP implementation
Howard: Writing report
Aybuke: CNN implementation with tensorflow, running experiments