MLP KMNIST Image Classification Report

McGill Computer Science

Luca Louis, Genevieve Quinn

1 Abstract

This project examines the performance of various neural network architectures on the Kuzushiji-MNIST dataset. We implemented multilayer perceptrons (MLPs) from scratch and verified the correctness of our gradient computations using finite differences, and we also developed convolutional neural networks (CNNs) using PyTorch. By varying network depth, hidden unit counts, and tuning key hyperparameters including learning rate, L2 regularization, and dropout, we observed that increased complexity generally improves classification accuracy, though performance gains tend to plateau beyond optimal model size. Notably, our CNN model had an improved performance compared to the MLP variants. These findings highlight the critical role of architectural choices and hyperparameter optimization for challenging image classification tasks.

2 Introduction

Image classification is a fundamental problem in computer vision, and neural networks have proven to be highly effective in addressing this challenge. In this project, we implemented and analyzed two distinct approaches to classify the Kuzushiji-MNIST dataset: a multilayer perceptron (MLP) constructed from scratch and a convolutional neural network (ConvNet) built using modern deep learning frameworks. Kuzushiji-MNIST is a dataset containing images of handwritten Japanese characters, formatted in a similar way to the original MNIST dataset. It has been used in many image classification studies including the evaluation of a novel all-analog computing chip [1] and evaluation of a data augmentation technique called Mixup, which has shown to improve the generalization and robustness of deep neural networks [2]. The motivation behind this study is to explore how different architectural choices, such as the number of hidden layers, the selection of activation functions, and the application of regularization techniques, impact the overall performance of the models. We found that increasing both the network depth and the number of hidden units improved classification accuracy, with deeper architectures out- performing simpler ones up to a certain point. Moreover, employing an iterative tuning procedure to optimize hyperparameters was crucial in improving performance over all models. Notably, our CNN model achieved higher test accuracies than its MLP counterparts. These findings emphasize the importance of model complexity and hyperparameter optimization in building image classifiers.

3 Datasets

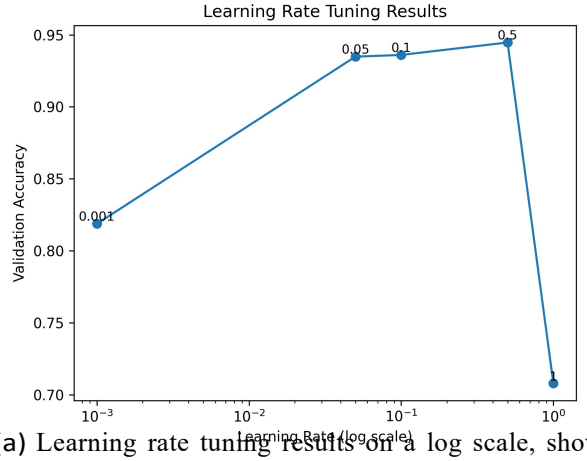
Kuzushiji-MNIST is a benchmark dataset comprising of grayscale images of handwritten Japanese characters, derived from historical documents. For this project, we are using a subset of the large dataset for a total of 70,000 28x28 KMNIST images and 10 class labels. Kuzushiji is a cursive script that was used for over a thousand years but is no longer taught in modern Japanese schools, making it so most Japanese natives cannot read texts written over 150 years ago. The Kuzushiji-MNIST dataset reformats these characters in a manner analogous to the original MNIST dataset, providing a challenging modern dataset for image classification and also bridging the fields of Japanese literature and machine learning. We began by loading the KMNIST dataset from its source files and verifying its structure to confirm that it consists of 70,000 images (60,000 for training and 10,000 for testing) of 28x28 pixels, which we then flattened into 784-dimensional vectors to act as our 784 input features for the MLP. We then computed the pixel-wise mean and standard deviation on the training set to standardize the data to ensure that all input features have a similar scale. Finally, we employed stratified sampling to create a validation set that preserves the original class distribution, ensuring balanced and normalized inputs for training the model.

4 Results

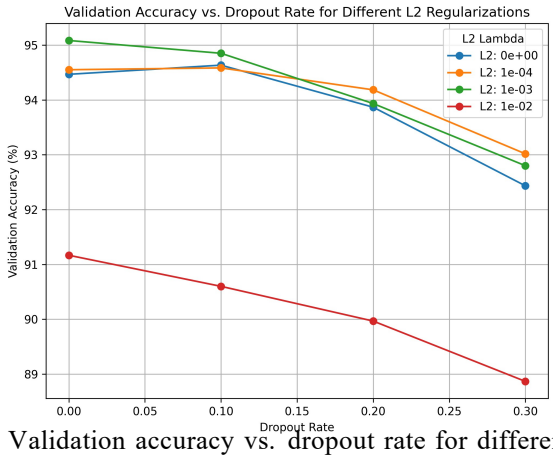
We implemented our MLP using backpropagation and the mini-batch gradient descent algorithm. In order to fine-tune our model's hyperparameters, we first performed a learning rate tuning procedure. We trained separate basic MLP models using candidate learning rates of 0.001, 0.05, 0.1, and 1 over 25 epochs, and recorded the final validation accuracies. The model trained with a learning rate of 0.5 achieved the highest validation accuracy

of 94.47% (Fig. 1a). Retraining the basic MLP using a learning rate of 0.5 resulted in a final test accuracy of 88.06%. Similarly, we further explored overfitting reduction by tuning regularization hyperparameters in our basic MLP model. We experimented with various combinations of L2 regularization and dropout rates. The tuning results revealed that setting the L2 penalty (λ) to 0.001 with a dropout rate of 0.0 provided the best validation performance of 95.08% (Fig. 1b). Retraining the model with these best hyperparameters led to a final test accuracy of 89.28%. Lastly, a test of training the model with and without learning rate decay showed that the model performs better with decay, with a validation and test accuracy of 94.47% and 88.06% compared to 92.75% and 85.86% respectively. These results demonstrate the importance of hyperparameter tuning, with both the learning rate and regularization parameters being critical to achieving model generalization.

Before beginning our experimental analysis, we verified the correctness of our gradient computations by comparing the analytical gradients from backpropagation with numerical gradients from the finite difference method. We obtained relative differences for all parameters (e.g. $W_0 : 1.95x10^{-6}$, $b_0 : 1.76x10^{-6}$, etc.) well below the $1x10^{-5}$ threshold, confirming that our backpropagation implementation was correct (Fig. 2).



(a) Learning rate tuning results on a log scale, showing how validation accuracy varies for each candidate learning rate.



(b) Validation accuracy vs. dropout rate for different L2 regularizations, illustrating how both hyperparameters jointly influence model performance.

Figure 1: (a) Learning rate tuning results; (b) L2 regularization and dropout rate tuning results.

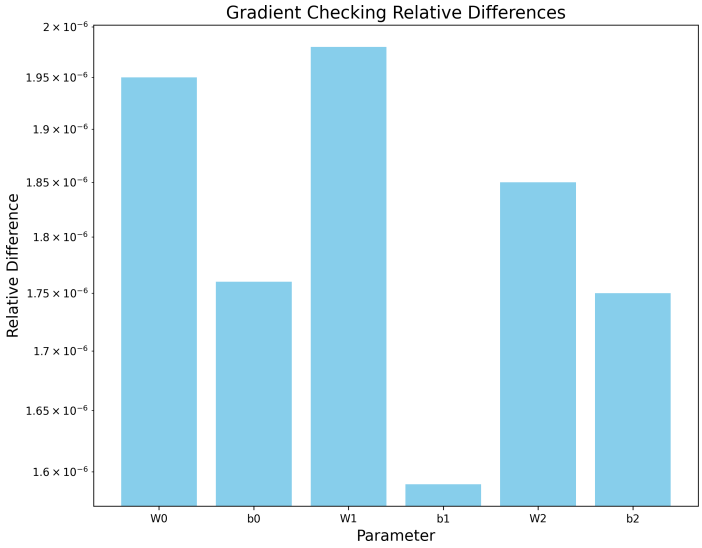


Figure 2: Bar chart displaying the relative differences between the analytical gradients and the numerical gradients for all parameters.

MLP with no hidden layers (Logistic Regression)

Our first model was an MLP with no hidden layers, which is implemented as logistic regression, comprising of 7,850 parameters. After training for 25 epochs using our best performing parameters determined from our experiments above, this model achieved a training accuracy of 83.16%, a validation accuracy of 79.68%, and a test accuracy of 67.29% (Fig. 5). This baseline provided a useful reference for evaluating the impact of added complexity.

MLP with one hidden layer

Next, we trained a single hidden layer MLP using ReLU activations and experimented with different hidden unit counts. We first had 32 hidden units for a total of 25,450 parameters. This model was improved compared to our no hidden unit model, with a training accuracy of 99.40%, a validation accuracy of 90.62%, and a test accuracy of 81.68%. However, we found that increasing the number of hidden units improved the model's performance. With 64 units (50,890 parameters), it achieved a test accuracy of 86.84%, which was further increased to 88.96% and 90.02% for 128 and 256 hidden units, respectively (Fig. 5). These results indicate that increasing the number of hidden units in a single hidden layer improves performance, with 256 units proving optimal among the tested options (Fig. 3).

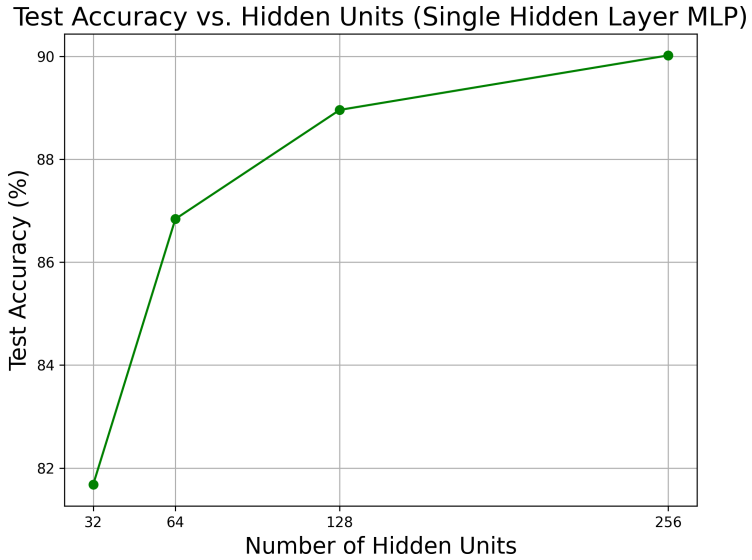


Figure 3: Plot of test accuracy as a function of the number of hidden units (32, 64, 128, and 256) in a single hidden layer MLP.

MLP with two hidden layers

We also experimented with different numbers of hidden units per layer in a MLP with two hidden layers. This model also used ReLU activation. Having 32 units per layer yielded a training accuracy of 99.57%, a validation accuracy of 91.78% and a test accuracy of 83.82%. Increasing to 64 units per layer resulted in similar training and validation accuracies, but a higher test accuracy of 88.31%. Increasing the number of hidden units to 128 per layer further increased the test accuracy to 89.94%, with 256 per layer having the highest test accuracy of 90.84% (Fig. 4 and 5). Thus, deeper models with two hidden layers and larger hidden dimensions provide a clear advantage over both the logistic regression baseline and the single hidden layer models.

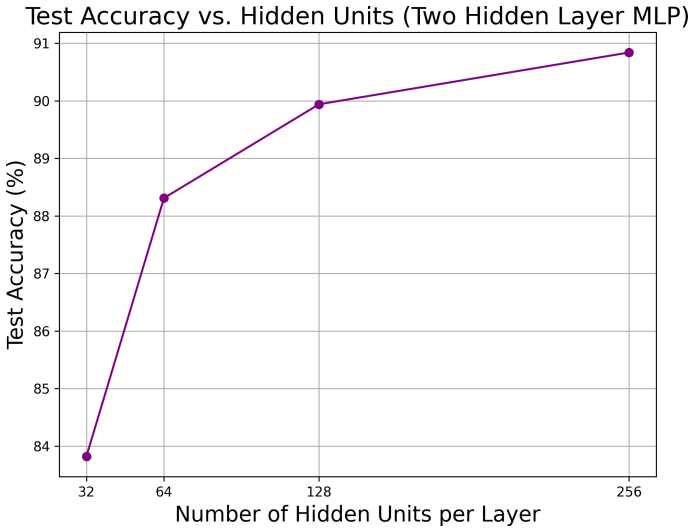


Figure 4: Plot of test accuracy as a function of the number of hidden units (32, 64, 128, and 256) in a two hidden layer MLP.

Deep MLP Architecture

Building on these findings, we wanted to try and improve our test accuracy further and experiment with a more complex architecture, so we designed a deeper MLP with four hidden layers having dimensions [1024, 512, 256, 128] and using ReLU activation. Since this model has such a large architecture, we had to use a learning rate of 0.1 rather than 0.5. This model, with nearly 1.5 million parameters, had a similar performance to our two-layer MLP with a training accuracy of 99.99%, validation accuracy of 96.02%, and a test accuracy of 90.73% (Fig. 5). This result demonstrates that, although increasing the depth and complexity of an MLP can initially improve performance, beyond a certain point further complexity does not substantially change accuracy.

These experiments conclude that increasing network depth—from no hidden layers (logistic regression) to one hidden layer, then two hidden layers, and finally a deep MLP with four hidden layers—leads to significant improvements in test accuracy, although beyond a certain level additional depth yields only marginal differences (Fig. 5 and 6). Likewise, increasing the number of hidden units per layer also improves performance. ReLU, the activation function used in these models, was a good choice because it helps mitigate the vanishing gradient problem by zeroing out negative values while retaining linearity for positive inputs. This piecewise linear behavior still introduces the necessary non-linearity, enabling the network to capture complex, non-linear relationships in the data that a purely linear model would miss. Overall, the benefits of ReLU combined with increased depth and capacity contribute to the observed improvements in model accuracy.

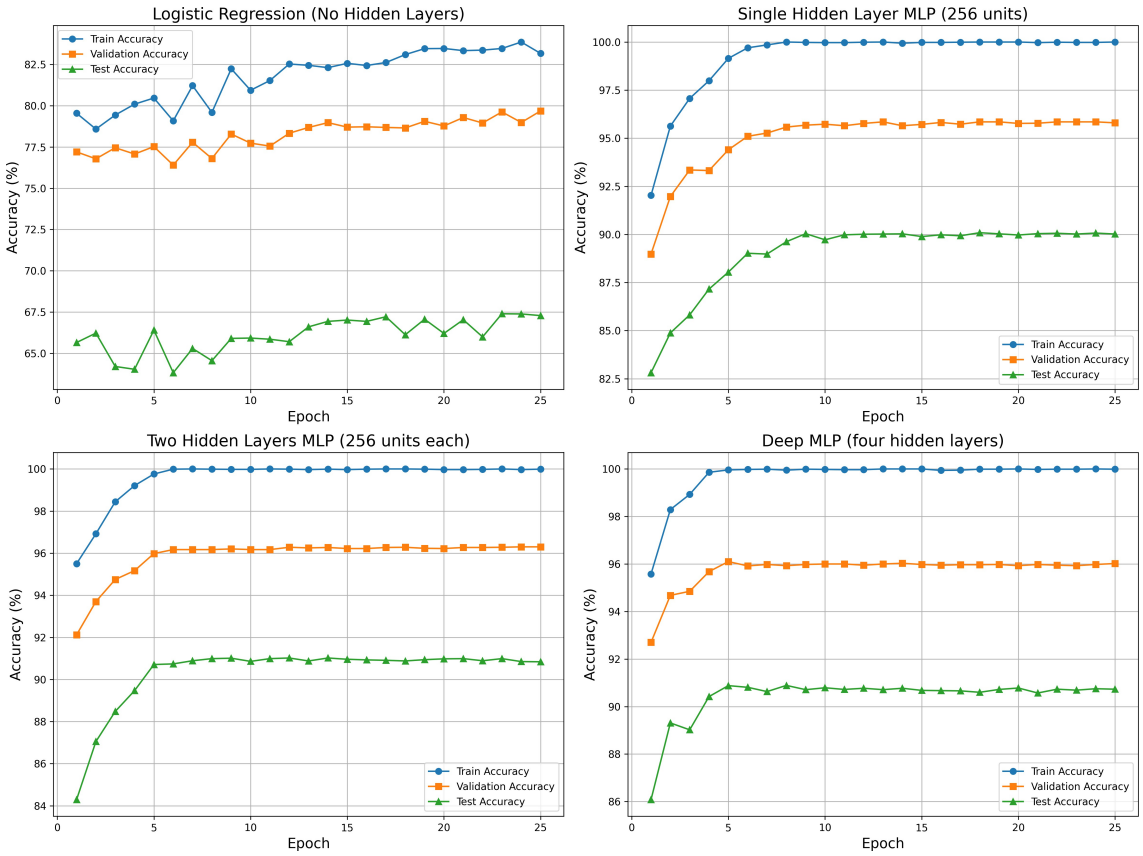


Figure 5: Learning curves depicting the training, validation, and test accuracies over 25 epochs for all MLP models (using best number of hidden units).

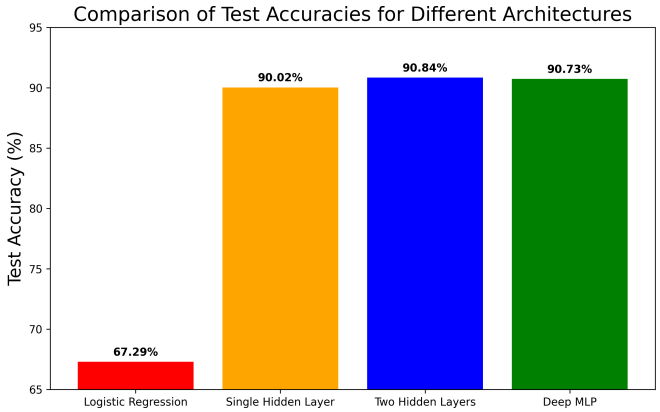


Figure 6: Bar chart summarizing the final test accuracies for all implemented architectures.

Activation Function Comparison

In order to try and improve our test accuracy for our best two hidden layer MLP (256 hidden units per layer), we experimented with different activation functions. We first implemented the model with Leaky ReLU, which differs from ReLU as it allows a small, non-zero output for negative inputs, thereby preventing hidden units from becoming inactive during training. With this implementation, the model achieved a validation accuracy of 96.23% and test accuracy of 90.80%, which is only a 0.04% decrease from normal ReLU activation (Fig. 7).

We also compared Sigmoid activation on our two layer MLP with 256 units per layer. However, this activation function performed slightly worse, with a validation accuracy of 94.45% and test accuracy of 88.53%. Further, the Sigmoid model also converges slower than the Leaky ReLU model (Fig. 7). This clearly indicates that ReLU and

Leaky ReLU are better suited for our architecture and dataset than Sigmoid. This is likely because Leaky ReLU's non-saturating behavior ensures that gradients remain informative even for negative inputs, whereas Sigmoid tends to saturate and suffer from vanishing gradients.

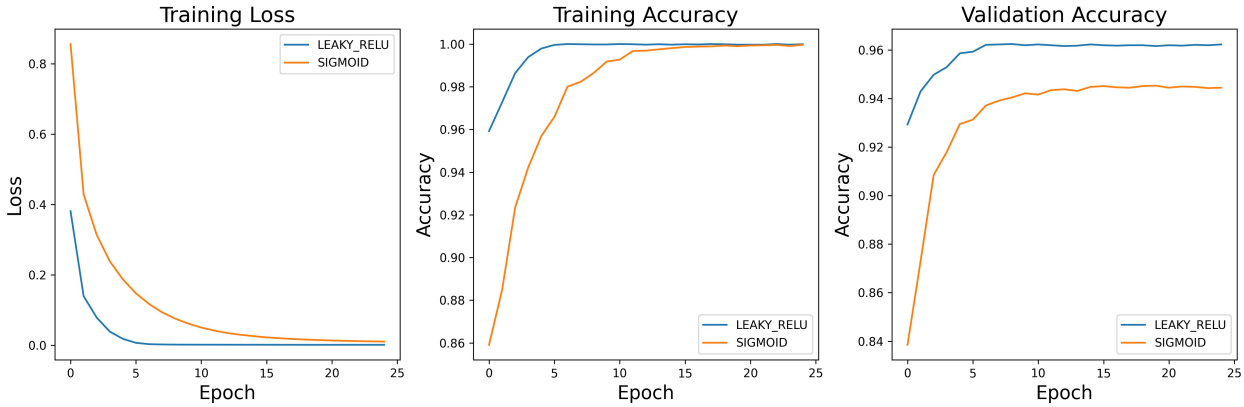


Figure 7: Training loss (left), training accuracy (center), and validation accuracy (right) curves comparing Leaky ReLU and Sigmoid activations over 25 epochs.

Regularization Experiments

To further improve our model's performance and combat overfitting, we incorporated L2 regularization into our two-hidden-layer MLP (with 256 units per layer). L2 regularization adds a penalty term to the loss function, proportional to the square of the weights, which encourages the model to keep weight values small. This constraint helps prevent the network from relying too heavily on any single feature or noise in the training data.

We experimented with various regularization strengths by setting the L2 lambda (λ) parameter to 0.0 (no regularization), 0.0001, 0.001, 0.01, and 0.1. Our results showed that with no regularization ($\lambda = 0.0$), the model achieved a test accuracy of 90.84%. When very small regularization values ($\lambda = 0.0001$ and $\lambda = 0.001$) were applied, performance remained nearly identical, around 90.86-91.15% test accuracy. However, increasing λ to 0.01 slightly reduced the test accuracy to 87.20%, and a high λ of 0.1 led to a significant drop in performance, with test accuracy falling to 61.98% (Fig. 8).

These results suggest that mild L2 regularization does no harm and can potentially stabilize model performance by reducing overfitting. In contrast, excessive regularization forces the weights to remain small, resulting in underfitting where the model is unable to capture important patterns in the data. Therefore, careful tuning of the regularization strength is essential to strike the right balance between model complexity and generalization.

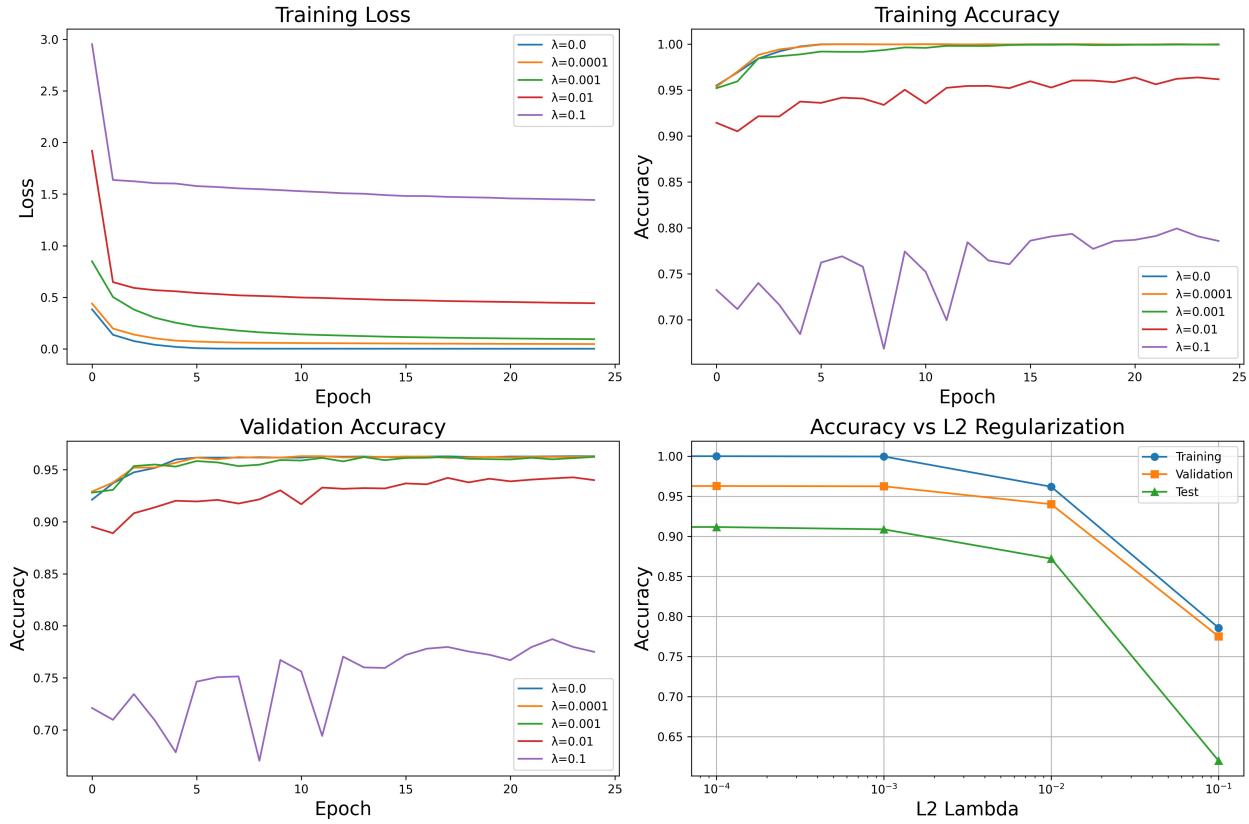


Figure 8: Effects of various L2 regularization strengths on a two-hidden-layer MLP, showing training loss, training and validation accuracies, and final test accuracy trends.

Convolutional Neural Network (CNN) Experiments

Finally, we used existing libraries such as PyTorch to implement a CNN model consisting of three convolutional layers followed by two fully connected layers, with ReLU activations used consistently throughout. The network architecture is as follows: the first convolutional layer converts the 28x28 single-channel input into 32 feature maps (3x3 kernel, stride 1, padding 1) followed by 2x2 max pooling. The second layer maps 32 channels to 64 channels (with similar settings) and is also followed by pooling. The third layer maps 64 to 128 channels. The output is then flattened and passed through an fully connected layer (fc1) whose hidden unit size was varied among 32, 64, 128, and 256, followed by a final fully connected layer (fc2) that produces 10 output classes. A dropout layer with a rate of 0.25 is applied after fc1 for regularization.

For training, we used both a batch size of 128 over 15 epochs with the Adam optimizer (learning rate = 0.001) and a StepLR scheduler (step size = 5, gamma = 0.1). We employed Cross-Entropy loss as the loss function. The experiments showed test accuracies of 95.80%, 96.22%, 96.45%, and 96.68% for fc1 sizes of 32, 64, 128, and 256 units respectively, with the best performance using 256 hidden units (Fig. 9).

Compared to our MLP models which achieved test accuracies up to 90.84%, the CNN improves performance by leveraging its convolutional layers to better capture spatial features in the images.

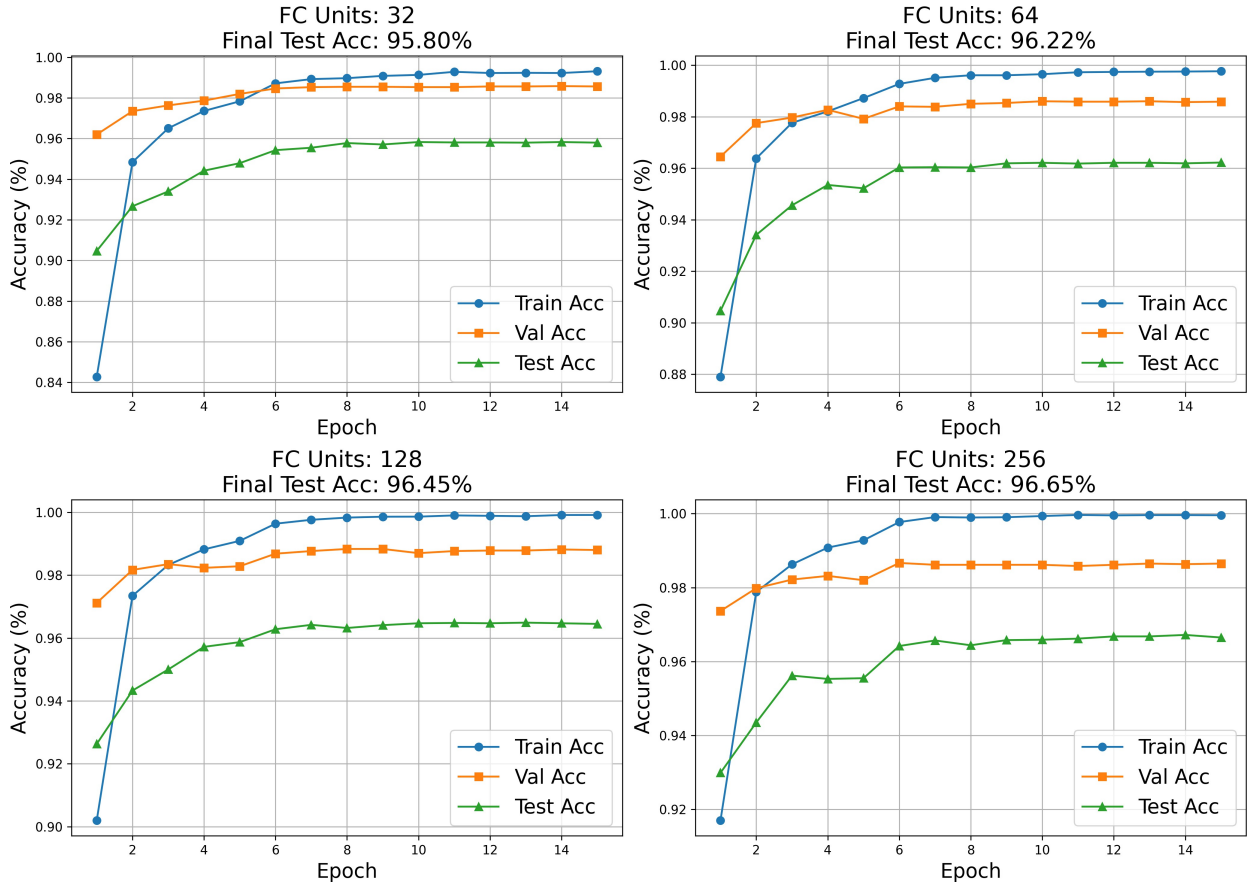


Figure 9: Learning curves for CNN models with varying fully connected (FC) hidden units (32, 64, 128, 256). Each subplot shows the training, validation, and test accuracy over 15 epochs, with the final test accuracy noted in the title.

5 Discussion and Conclusion

Our experiments demonstrate that increasing model complexity, whether by adding hidden layers or increasing the number of hidden units, improves classification performance on the Kuzushiji-MNIST dataset. The baseline logistic regression model (no hidden layers) achieved the lowest accuracy. However, our results revealed that a two-layer MLP with 256 hidden units per layer outperformed both the shallow (single-layer) and the deep (four- layer) architectures. In fact, while the deep MLP with four hidden layers approached similar performance, it did not provide a significant improvement over the two-layer model, suggesting there is an optimal balance between depth and capacity that maximizes performance without incurring unnecessary computational load. In the CNN experiments, we observed that models with larger fully connected layers (up to 256 units) converged to higher test accuracies, with the best model achieving nearly 96.7% test accuracy, which is a large improvement over the MLP counterparts. Gradient checking confirmed the correctness of backpropagation implementation, ensuring that our training process was reliable. Future work could explore alternative regularization strategies, different activation functions, or more advanced architectures such as residual networks to further improve performance on this challenging dataset.

6 Statement of Contributions

All group members produced unique code and the final Google Colab is a cumulation of everyone's code. Genevieve drafted the report with editing done by Luke and Luca.

References

[1] Y. Chen, M. Nazhamaiti, H. Xu, and et al. All-analog photoelectronic chip for high-speed vision tasks. *Nature*, 623:48–57, 2023.

[2] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization?, 2021.