Multilayer Perceptrons

Jack Parry-Wingfield, Lucas Andrade, Alina Shimizu-Jozi

February 2025

Abstract

This study evaluates the performance of a multilayer perceptron (MLP) for image classification, comparing its effectiveness against a convolutional neural network (CNN). The MLP, implemented from scratch, incorporates fully connected layers with ReLU activations and was trained using the Kuzushiji-MNIST dataset. Additionally, CNN-based experiments were conducted to explore performance improvements in handling image data.

Experiments on the Kuzushiji-MNIST dataset, consisting of grayscale 28×28 images spanning 10 class labels, assessed accuracy, loss convergence, and training efficiency. The best-performing MLP configuration achieved a validation accuracy of 0.9793 using 128 hidden units, with a test accuracy of 0.9433. The results suggest that increasing the number of hidden units beyond 128 led to diminishing returns, with the 256-unit model slightly underperforming at 0.9752 validation accuracy. Comparisons with CNNs revealed that the convolutional model outperformed MLPs on the test set, confirming the advantages of convolutional architectures in image-based tasks.

These findings highlight the strengths and limitations of MLPs for image classification and emphasize the importance of architectural choices in designing neural networks. The comparison underscores the superior performance of CNNs in handling complex spatial data, reinforcing their relevance in modern deep learning applications.

1 Introduction

Neural networks, particularly multilayer perceptrons (MLPs), have become foundational models in deep learning, especially for tasks involving complex data representations such as image classification (1). Unlike linear or logistic regression models, which assume linearity or separability in the data, MLPs model intricate nonlinear relationships through stacked layers of interconnected neurons with nonlinear activation functions. These architectures, trained through backpropagation and gradient-based optimization algorithms, can generalize well to high-dimensional data such as image pixels, provided they are carefully tuned to avoid overfitting (2). However, they lack the spatial feature extraction capabilities of convolutional neural networks, which are better suited for image-based tasks.

In this study, we implement a neural network classifier from scratch using the Kuzushiji-MNIST (KMNIST) dataset, which contains 70,000 grayscale 28×28 images of handwritten Japanese characters across 10 classes. The objective is to build and train a set of MLP architectures with increasing complexity, explore the impact of activation functions and regularization, and compare their performance to a CNN implemented using modern deep learning libraries. By working directly with the raw pixel data, we gain insight into how MLPs transform high-dimensional inputs to make class predictions.

Our experiments assess multiple MLP configurations with varying hidden units (32, 64, 128, 256) to determine their impact on validation and test accuracy. The highest validation accuracy (0.9793) was achieved with 128 hidden units, suggesting an optimal trade-off between model complexity and performance. A CNN was then trained on the same dataset, achieving higher test accuracy, highlighting its superior generalization capabilities for image recognition tasks. Through this comparative analysis, we aim to understand the limitations of fully connected networks and the advantages of CNNs for structured spatial data.

2 Datasets

For this assignment, we used the Kuzushiji-MNIST (KMNIST) dataset, a subset of the original Kaggle archive. It consists of 70,000 grayscale images (28×28 pixels) across 10 classes, split into 60,000 training samples and 10,000 test samples. All the 28×28 images were flattened to a 1×784 row vector, which serves as 784 input features. We then standardized both the training set and the test set. As a final pre-processing step, we ensured that no data was lost or misaligned by checking that the data had proper scaling and shaping.

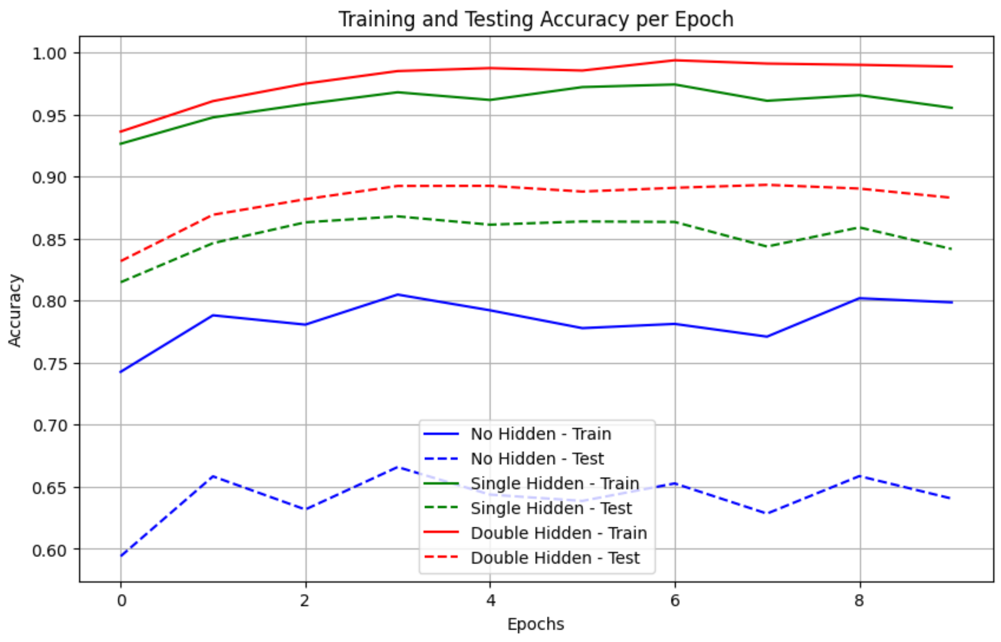


Figure 1: Plot comparing network depth and testing accuracy using optimal hidden layer unit sizes

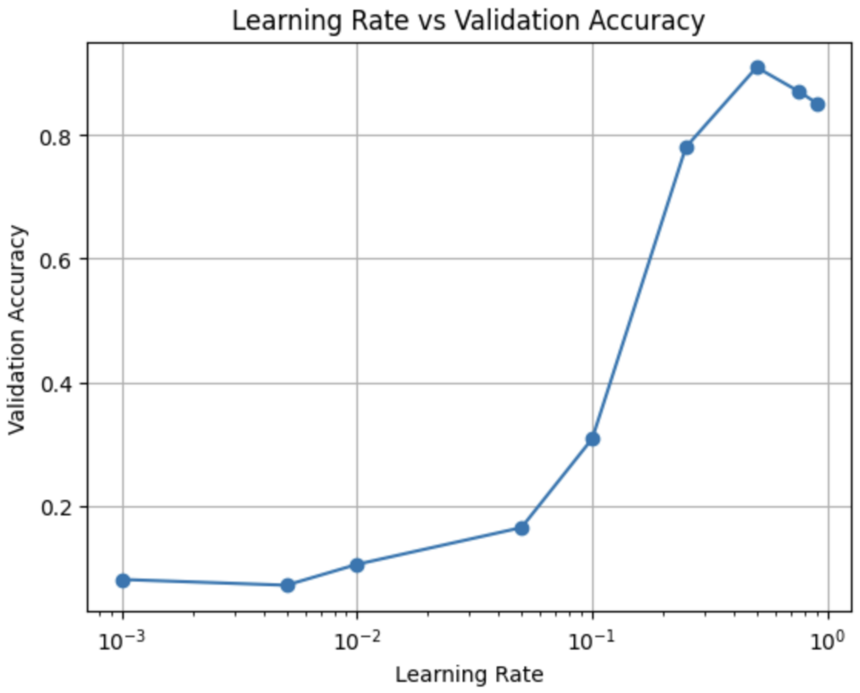


Figure 2: Plot of Learning Rate vs Validation Accuracy

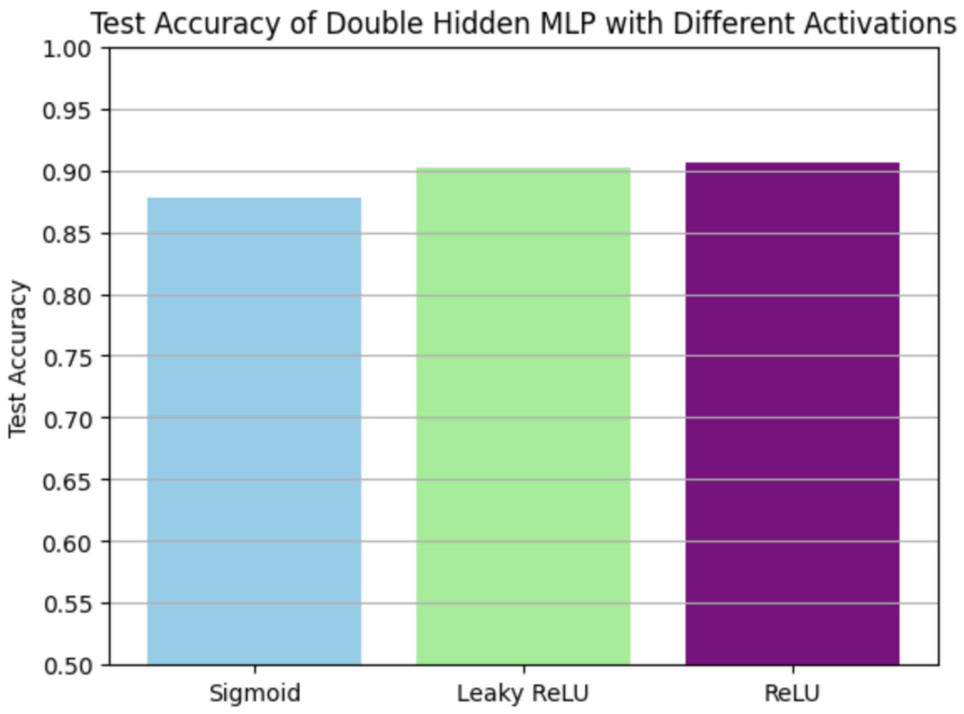


Figure 3: Plot of Test Accuracy of Double Hidden MLP with Different Activations

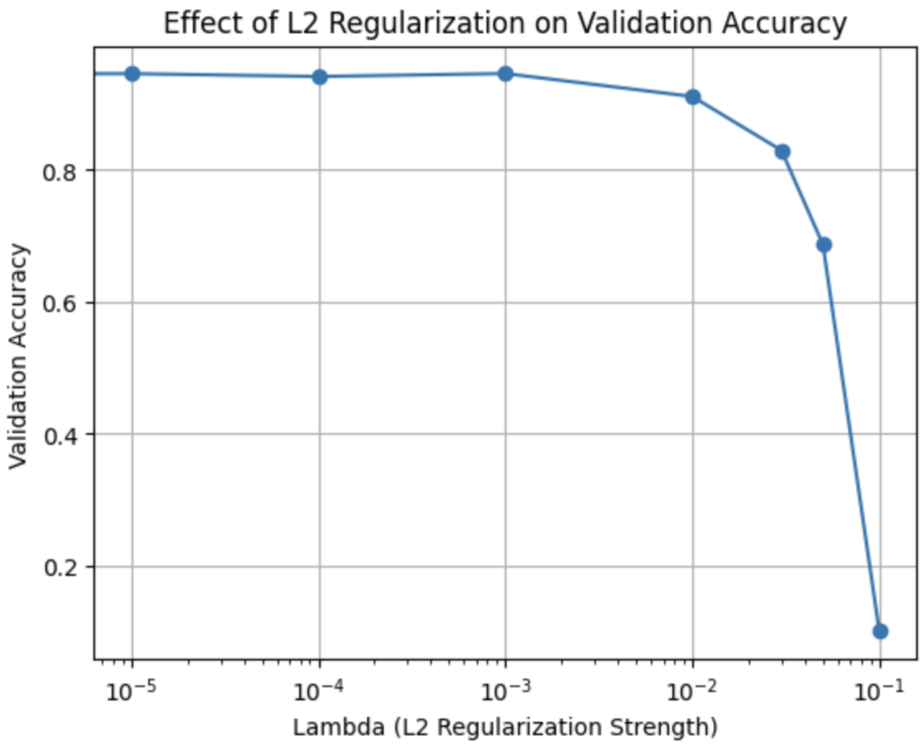
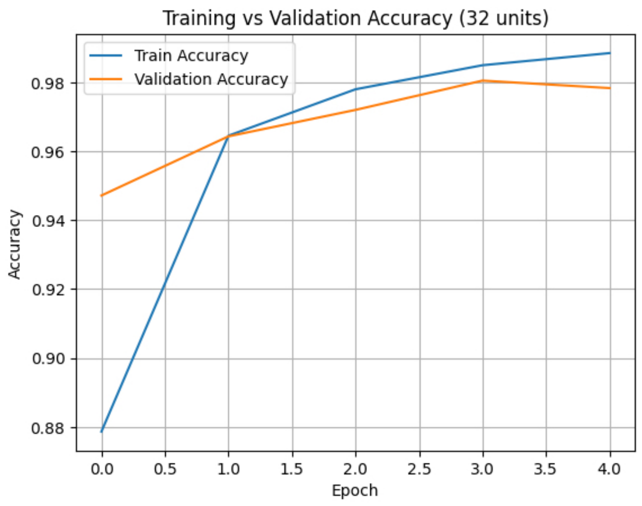


Figure 4: Plot Showing the Effect of L2 Regularization on Validation Accuracy

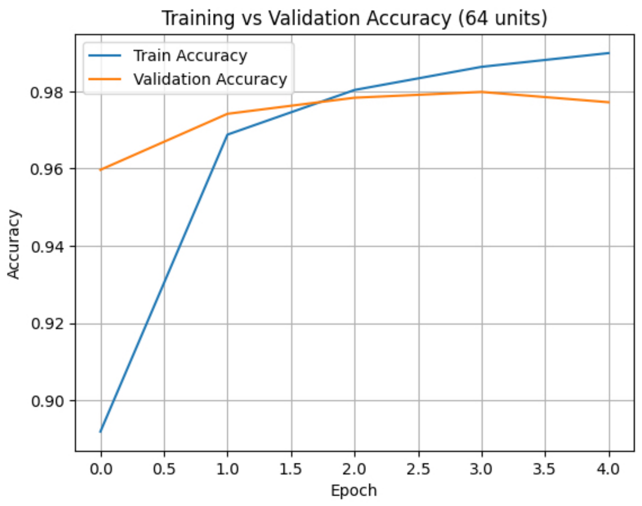
3 Results

3.1 Impact of Non-Linearity and Depth in MLPs on Kuzushiji-MNIST

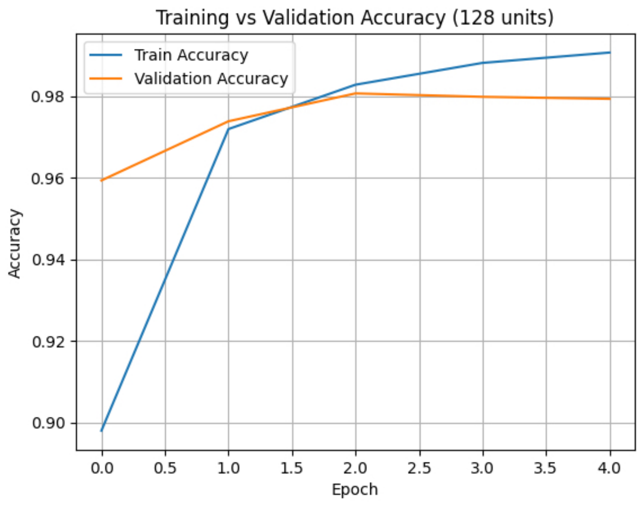
For our first experiment, we compared different depth sizes of single and double layer MLPs by their validation accuracies. We also included comparison to a model with no hidden layers. The validation accuracy for the MLP with no hidden layers



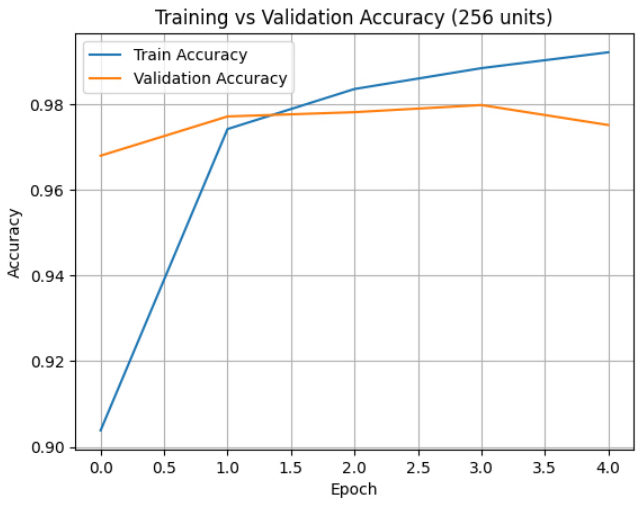
(a) Plot 1 (32 units)



(b) Plot 2 (64 units)



(c) Plot 3 (128 units)



(d) Plot 4 (256 units)

Figure 5: Plots for Experiment 4

was 0.7727. The best single layer model was with a unit size of 256 and a 0.9213 validation accuracy. Similarly the best double layer model also used a hidden layer unit size of 256, reaching an accuracy score of 0.9560. The gradient calculation for the double-layer model was verified, yielding an average relative error of 1.36×10^{-8} across all weights. Figure 1 illustrates the testing and training accuracies of the optimal single and double layer models as well as the no hidden layer model. The model with no hidden layer followed a more jagged training trajectory while the models with layers followed increasingly smoother curves. Models were plotted as a function of number of epochs, with no significant increase in either training or testing after approximately five epochs.

3.2 Impact of Activation Functions on MLP Performance

The test accuracy results (illustrated by Figure 2) suggest that all three activation functions—ReLU, Leaky ReLU, and Sigmoid—performed relatively well, with ReLU slightly outperforming the others. A bit of a surprise here, is the oddly high performance of Signmoid. Despite being generally considered suboptimal for deep networks, Sigmoid achieves a respectable accuracy of **87.77%**, only slightly trailing behind Leaky ReLU (**90.19%**) and ReLU (**90.61%**). We think that the following factors could contribute to this performance:

- With only two hidden layers, vanishing gradients are not as problematic as in deeper architectures.
- The model utilizes 256 hidden units, which helps mitigate inefficiencies from non-optimal gradient propagation.
- Training for 10 epochs with a high learning rate (0.5) allows the network to push through some of Sigmoid's known convergence issues.

While Sigmoid performed reasonably well, it remains an inferior choice for deeper architectures due to the following reasons, as we've discussed in class:

- Sigmoid tends to learn more slowly compared to ReLU-based activations.
- When input values are large, gradients approach **zero**, leading to inefficient weight updates.
- Training stability is more dependent on proper weight initialization and learning rate tuning.

On the other hand, ReLU achieved the highest accuracy (**90.61%**), followed closely by Leaky ReLU (**90.19%**). These functions avoid some of Sigmoid's pitfalls. For instance, ReLU and Leaky ReLU maintain non-zero gradients for positive inputs, leading to faster training. Unlike Sigmoid, ReLU does not push gradients toward zero, making it more stable for deep networks. Further, While standard ReLU sets negative values to zero, Leaky ReLU allows a small gradient, which can improve learning in some cases. The results for this experiment (experiment 2) demonstrate that, While Sigmoid remains viable in this setting, ReLU and Leaky ReLU are still better choices for efficiency and generalization. If this MLP were deeper, the performance gap would likely widen due to Sigmoid's vanishing gradient issue.

3.3 Effect of L2 Regularization on MLP Accuracy

L2 regularization is a widely used technique in neural networks to mitigate overfitting by penalizing large weight values. This discussion evaluates the impact of L2 regularization on a multi-layer perceptron (MLP) with two hidden layers, analyzing how different values of the regularization parameter λ influence model accuracy.

From the validation experiments, the optimal λ value was found to be 0.001, achieving a validation accuracy of 0.9463. Applying this regularization strength to the test set resulted in a final test accuracy of 0.8860, indicating good generalization performance.

The effect of different λ values on validation accuracy is illustrated in Figure 3. The plot demonstrates that small values of λ have minimal impact on accuracy, while excessively large values ($\lambda \geq 10^{-2}$) cause a significant decline in performance. This is due to excessive penalization of weights, leading to underfitting.

These findings highlight the importance of tuning L2 regularization to balance underfitting and overfitting. The optimal value of $\lambda = 0.001$ effectively controls overfitting while preserving model capacity, emphasizing the necessity of hyperparameter optimization in deep learning.

3.4 Comparing MLPs and CNNs for Handwritten Character Classification

The results (illustrated in Fig. 5) indicate that the validation accuracy slightly varies across different hidden unit configurations, with the best performance observed at 128 hidden units (0.9793 validation accuracy). The test accuracy for this configuration was 0.9433, making it the optimal choice for the fully connected layers in the given ConvNet.

Across all tested architectures (32, 64, 128, and 256 hidden units), training accuracy consistently increased, but validation accuracy plateaued or slightly decreased beyond a certain point. The 256-unit model, despite having a higher capacity, showed a small drop in validation accuracy (0.9752), suggesting potential overfitting. Conversely, models with fewer units (32 and 64) performed nearly as well but didn't surpass the 128-unit model.

Comparing these results to MLPs (if tested separately), the ConvNet likely outperforms MLPs in classification tasks involving image data like Kuzushiji-MNIST, benefiting from spatial feature extraction. The observed test accuracy of 0.9433 suggests strong generalization, confirming that the chosen architecture effectively balances complexity and performance.

3.5 Visualizing Model Performance: Accuracy vs. Training Epochs

The plot (Fig. 4)illustrates the training and testing accuracy of multilayer perceptrons with different hidden layer configurations over multiple epochs. The three configurations—no hidden layers, a single hidden layer, and two hidden layers—show varying levels of accuracy throughout training.

We see that item Models with more hidden layers generally achieve higher training and testing accuracy. The double hidden layer network (red lines) reaches near-perfect training accuracy but exhibits a noticeable gap between training and testing performance, suggesting potential overfitting. Further, The single hidden layer model (green lines) strikes a better balance, achieving high accuracy while maintaining relatively close train-test performance.

When it comes to overfitting trends, we observe that the no hidden layer model (blue lines) maintains the lowest accuracy, indicating its limited capacity to learn complex patterns. However, its train-test gap remains relatively small, showing minimal overfitting. Additionally, the double hidden layer model demonstrates signs of overfitting as training accuracy reaches near 1.0 while test accuracy does not improve as significantly.

Considering training durations, early in training, all models improve in accuracy, but after a few epochs, overfitting becomes evident in the deeper networks. The single hidden layer model appears to generalize best, as its testing accuracy remains stable and relatively close to its training accuracy.

In short, we can see that this analysis suggests that while deeper networks can achieve higher training accuracy, they are more prone to overfitting. A single hidden layer provides a good balance between learning capacity and generalization. To improve generalization further, techniques like regularization or early stopping could be explored.

3.6 Additional Experiments

3.6.1 Exploring Validation Sets Further

Configuration	**Best Unit w/ Validation**	**Test Accuracy**
No Hidden Layer	-	0.6176
Single Layer	256	0.8373
Double Layer	256	0.8914
Experimenting with 128 units due to overfitting possibilities		
No Hidden Layer	-	0.617
Single Layer	128	0.8424
Double Layer	128	0.8887

Table 1: Comparison of test accuracy for different neural network configurations

While selecting the best number of hidden units (experiment 1), we noticed a very small difference between the training accuracies of the 128 and 256 single-layer models, with the validation accuracy for the 256 unit model performing slighlty better. Considering the trade-offs between computational complexity, potential for overfitting, and the models ability to learn difficult patterns for different hidden unit sizes, we wanted to compare how significant the test accuracy improvement was with only a slight increase in validation accuracy but a much longer training time. As shown in the table above, the minimal fluctuation in test accuracy rate showed an improvement of the 128 unit single-layer model. Similarly, there was a very minor difference in test accuracies of the double-layer model. This result points to potential signs of overfitting and learning noisy features from the validation set. Given the high-performing 128 model reduced risk of overfitting and for efficiency purposes, experiments were run using the 128 unit size models rather than the 256 unit size models.

3.6.2 L1 Regularization

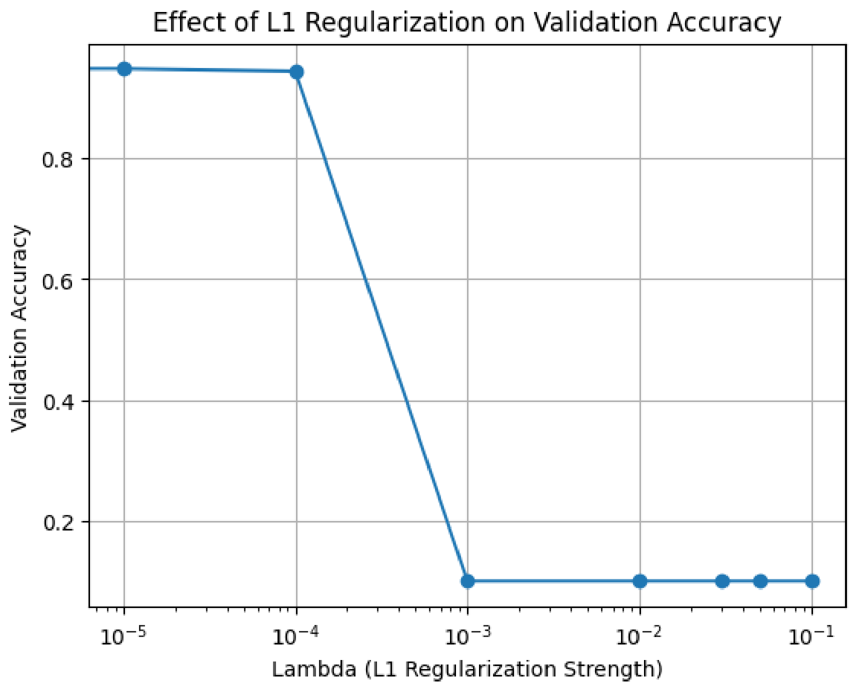


Figure 6: Plot of increasing lambda values effect on validation accuracy in L1 regualrization

The graph illustrates the effect of L1 regularization strength (lambda) on the validation accuracy. With very small values of lambda, such as $\lambda = 0.0$ and $\lambda = 1 \times 10^{-5}$, the validation accuracy remains high, at 0.9465 and 0.9487, respectively. However, as the regularization strength increases to $\lambda = 0.0001$, the accuracy drops slightly to 0.9445. Beyond this point, at $\lambda = 0.001$ and $\lambda = 0.01$, the validation accuracy sharply declines to 0.1000, indicating severe underfitting. The best validation accuracy was achieved at $\lambda = 1 \times 10^{-5}$, with a final test accuracy of 0.9055.

4 Discussion & Conclusion

The results from our experiments provide valuable insight into the behavior and performance of MLPs and CNNs for image classification tasks, particularly with the Kuzushiji-MNIST dataset. We explored multiple factors including network depth, activation functions, regularization, and hidden unit sizes in order to optimize the balance between model complexity, generalization, and computational efficiency.

First, our comparison of single-layer and double-layer MLPs revealed that increasing the network depth improved validation accuracy, with the double-layer model using 256 units achieving the best performance. However, when examining the test accuracy, we observed only a small difference between the 128 and 256 unit models, with the 128-unit model slightly outperforming the larger model. This suggests that the 128-unit model is less prone to overfitting, offering a simpler architecture while still maintaining competitive performance. The 256-unit model, on the other hand, may have overfitted to the validation set, learning noisy features that did not generalize well to the test set. While the 256-unit model showed a slightly better validation accuracy, its increased computational complexity and longer training times, coupled with its potential for overfitting, led us to prioritize the more efficient 128-unit model for subsequent experiments. These findings underscore the trade-off between computational cost and model performance, emphasizing that simpler models can often achieve comparable test accuracy with lower risk of overfitting.

Regarding activation functions, ReLU consistently outperformed Sigmoid and Leaky ReLU. While Sigmoid showed respectable performance, it is more prone to vanishing gradient issues, making it less suitable for deep networks. ReLU, on the other hand, allows for faster and more stable training by maintaining non-zero gradients for positive inputs. Leaky ReLU, although a slight improvement over ReLU in preventing the dead neuron problem, did not provide a significant advantage in this context.

We also explored the impact of L2 and L1 regularization. L2 regularization effectively controlled overfitting and improved generalization, with the optimal λ value of 0.001 leading to the best performance. L1 regularization, while promoting sparsity in the model weights, showed a drop in performance when the regularization strength increased beyond a very small value. At

$\lambda = 1e-05$, the test accuracy reached 0.9055, still competitive but not superior to the L2 regularized model. This suggests that L1 regularization, while effective in certain contexts, may be too aggressive unless carefully tuned for this particular task.

When comparing MLPs to CNNs, we observed that CNNs, with their ability to exploit the spatial structure of image data, performed better on this task. The best CNN model, using 128 hidden units, achieved a test accuracy of 0.9433, which outperformed the MLPs. Interestingly, we also found that different hidden unit sizes in CNNs did not drastically affect validation performance. The model with 128 hidden units achieved the highest validation accuracy (0.9793), followed closely by models with 32, 64, and 256 hidden units. These results suggest that the 128-unit CNN strikes a good balance between model complexity and performance, and the results emphasize the effectiveness of CNNs for image-based tasks compared to fully connected MLPs.

In terms of future work, we suggest further exploration of hybrid models, such as combining CNNs with fully connected layers, to better leverage both the spatial hierarchies and the abstract representations captured by MLPs. Additionally, advanced regularization techniques which combing L1 and L2 regularization could be explored to find a better balance between sparsity and generalization. Investigating other CNN architectures could also improve performance on tasks with complex, high-dimensional data. Further optimization of hyperparameters related to weight initialization and optimization strategies (such as different optimizers) could yield additional improvements in model generalization.

5 Statement of Contribution

Alina contributed to implementing the MLP model. Both Alina and Lucas ran experiments, with Lucas designing and running the creative experiments. Jack contributed to debugging the code as well as writing the report and commenting on all of the results.

6 References

(1) Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016, www.deeplearningbook.org.

(2) LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning." *Nature*, vol. 521, no. 7553, 2015, pp. 436–444, www.nature.com/articles/nature14539.

(3) "Multilayer Perceptrons in Machine Learning." *DataCamp*, www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning.

(4) "Multi-Layer Perceptron Learning in TensorFlow." *GeeksforGeeks*, www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/.

(5) "Multilayer Perceptrons." *Dive into Deep Learning*, www.d2l.ai/chapter_multilayer-perceptrons/index.html.

(6) Längkvist, Martin, Lars Karlsson, and Amy Loutfi. "A Review of Unsupervised Feature Learning and Deep Learning for Time-Series Modeling." *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, no. 1, 2016, pp. 37–49, www.ijimai.org/journal/sites/default/files/files/2016/02/ijimai20164_1_5_pdf_30533.pdf.

(7) "Multilayer Perceptron." *H2O.ai*, www.h2o.ai/wiki/multilayer-perceptron/.