Assignment 3: Classification of Image Data

Group #63

Members: Kynda Nashif, Aditi Potnis, Leonardo Falvo

Abstract

In this assignment, we explore the implementation and performance of neural networks for image classification using the Kuzushiji-MNIST dataset. We implement a multilayer perceptron (MLP) from scratch and conduct experiments to investigate the effects of network depth, activation functions, and regularization on classification accuracy. A convolutional neural network (ConvNet) is also developed using existing libraries to provide a comparative baseline. Our study found that 2 hidden layers performed the best with 256 hidden units and a RELU activation function. We found that the accuracy of the ConvNet was higher than that of using MLPs on the test set.

1. Introduction

Image classification has emerged as a cornerstone of modern machine learning, powering applications from autonomous systems to medical diagnostics. Recent advances in deep learning have propelled neural networks, such as multi-layer perceptrons (MLPs) and convolutional neural networks (ConvNets), to the forefront of image recognition research.

In this assignment, we focus on classifying images from a subset of the Kuzushiji-MNIST dataset which features *Kuzushiji*, cursive Japanese writing. Before the recent advancements in image classification machine learning, only a few experts could read Japanese books written or published over 250 years ago [2]. Now, neural networks can tackle this challenge and bridge the gap in Japanese literature, with accuracies of up to 98.33% [2].

The primary objective of this project is a comparative analysis enabling us to investigate the effects of key design choices such as non-linearity, network depth, activation function selection, and regularization strategies on model performance. First, we implement an MLP from scratch to gain a deeper understanding of the core training mechanisms, including backpropagation and gradient descent. The MLP is designed to allow flexibility in the number of hidden layers and units per layer, and supports various activation functions including ReLU, Sigmoid, and Leaky-ReLU. [*Placeholder: Describe the implementation details and hyperparameter settings.*] Second, we compare its performance with that of a ConvNet constructed using popular deep learning libraries. ConvNet is implemented using libraries such as Keras or PyTorch. The ConvNet architecture comprises three convolutional layers followed by two fully connected layers, with ReLU activations throughout. [*Placeholder: Provide details on the network architecture, hyperparameter tuning, and training procedure.*]

Our study is motivated by the need to bridge theoretical insights with practical application. While an MLP provides a straightforward framework for exploring fundamental neural network operations, ConvNets are renowned for their ability to capture spatial hierarchies inherent in image data—an advantage that has been validated in seminal works [1]. By juxtaposing these two architectures, we aim to elucidate the trade-offs between model simplicity and performance, ultimately informing more effective design decisions for image classification tasks.

2. Datasets

The dataset used in this project is a subset of the Kuzushiji-MNIST dataset, consisting of 70,000 grayscale images (28x28 pixels) corresponding to 10 distinct classes. These classes represent each row of the Hiragana alphabet The training set contains 60,000 images and the test set 10,000 images. Before training, images are flattened into 784-dimensional vectors and standardized. Furthermore, our exploratory analysis through class distribution revealed that the classes were simply uniformly distributed.

3. Results

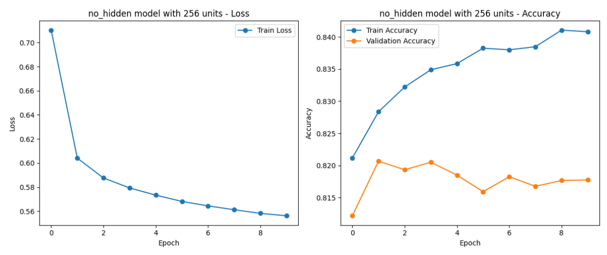
3.1. Hidden Layers Experiment

We evaluate three different MLP configurations:

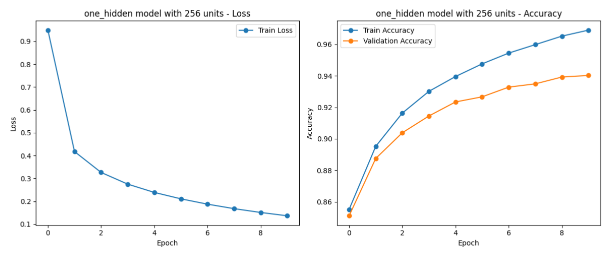
Table 1: Accuracy Results for different hidden layer configurations

Hidden Units	**No Hidden Layers**	**Single Hidden Layer**	**Two Hidden Layers**
32	0.8153	0.9149	0.9066
64	0.8157	0.9261	0.9215
128	0.8169	0.9352	0.9342
256	0.8177	0.9403	0.9417

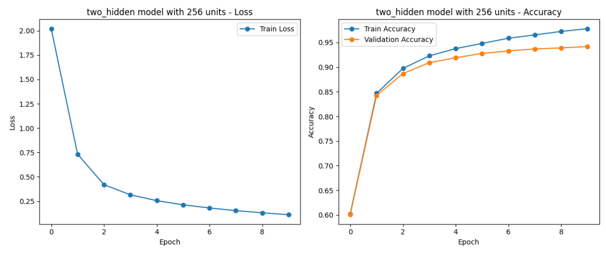
Figure 1: Loss



(a) No Hidden Layers - 256 units



(b) Single Hidden Layer - 256 units



(c) Double Hidden Layer - 256 units

The results show that adding more units and increasing network depth improve classification accuracy. Without any hidden layers, the baseline model achieved an accuracy of 69.29% on the test set. Although deeper models can introduce risks of overfitting if one is not careful, these findings demonstrate that incorporating non-linearity and additional depth enables the model to capture more complex features, which is exactly what the theory states.

3.2. Activation Function Comparison

Table 2: Accuracy Results for different Activation Functions

Function/Accuracy	**No Hidden Layers**	**Two Hidden Layers**
Relu	0.6929	0.8689
Sigmoid	0.6932	0.5755
Leaky-Relu	0.6927	0.8641

We compare the test accuracies of different models with different activation functions. We observe that for the two-hidden-layer model, Sigmoid performs incredibly poorly, while Relu and Leaky-Relu functions perform similarly. Indeed, the Sigmoid function is an activation function not meant for deep learning since the gradient shrinks to much.

To test this theory, an additional run of the models was done, but with no hidden layers. Suddenly, Sigmoid was performing slightly better than both Relu and Leaky-Relu.

In general, both leaky-relu and relu performed at almost the same levels of accuracy. Relu performed slightly better. Leaky-Relu is designed to avoid the "dying-ReLu" problem, which is not an occurrence that has affected these results.

3.3. Regularization Effects

We apply L2 regularization to the two-hidden-layer MLP model and explore its influence on model accuracy. L2 regularization adds a penalty term proportional to the sum of squared parameters to the loss function:

$$L_{\text{total}} = L_{\text{data}} + \lambda \sum_i w_i^2.$$

In this experiment, we investigate the impact of adding L2 regularization on a two-hidden layer MLP with ReLU activations and a softmax output layer. We trained the model using various regularization strengths, $\lambda \in \{0.0, 0.001, 0.01, 0.1\}$, while keeping the learning rate, number of epochs, and batch size constant. The training loss and test accuracy for each λ value were recorded.

For $\lambda = 0.0$, the training loss decreased steadily over 10 epochs, reaching a final loss of 0.0115, with a test accuracy of 86.81%. This serves as our baseline performance.

When a small regularization of $\lambda = 0.001$ was applied, the training loss was slightly higher (stabilizing around 0.0606) and the test accuracy dropped marginally to 86.28%. This suggests that a mild regularization constraint is beginning to influence the weight updates without dramatically affecting performance.

However, increasing the regularization strength to $\lambda = 0.01$ resulted in a significantly higher training loss (approximately 0.2651 at the final epoch) and a further drop in test accuracy to 83.84%. This indicates that the model starts to underfit, as the penalty on the weight magnitudes becomes too strong.

Finally, with a regularization of $\lambda = 0.1$, the model exhibited severe underfitting. The training loss remained high throughout the epochs (ending at 0.3962), and the test accuracy plummeted to 27.68%. This clearly demonstrates that excessive regularization can hinder the model from fitting the training data, leading to poor generalization.

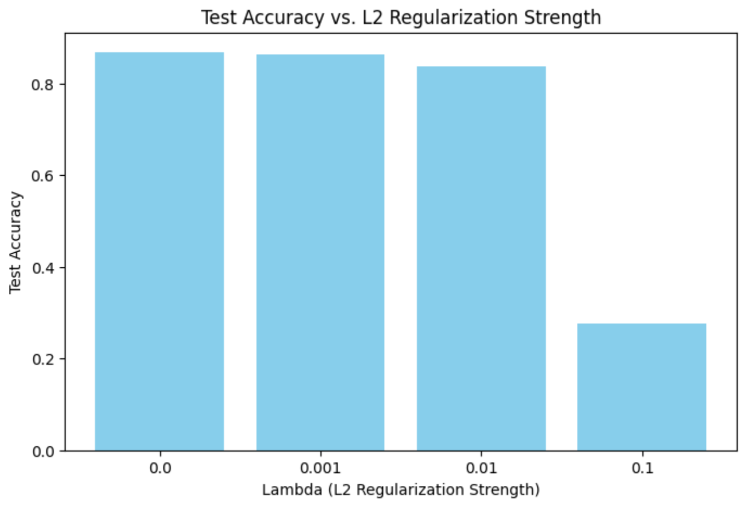


Figure 2: Test Accuracies for varying λ values

Overall, these results are consistent with the expected behavior of L2 regularization. While a small amount of regularization can help prevent overfitting, our baseline model does not seem to be severely overfitting. Therefore, even a slight regularization leads to a minor drop in performance, and too strong a penalty causes significant underfitting. This experiment highlights the importance of choosing an appropriate regularization strength to balance the trade-off between bias and variance.

3.4. ConvNet Experiments

We trained a convolutional neural network (CNN) on the same dataset to compare its performance with our MLP models. The CNN comprises several convolutional layers with ReLU activations. After experimenting with various configurations, we found that using 256 units in the final fully connected layer yielded the best performance.

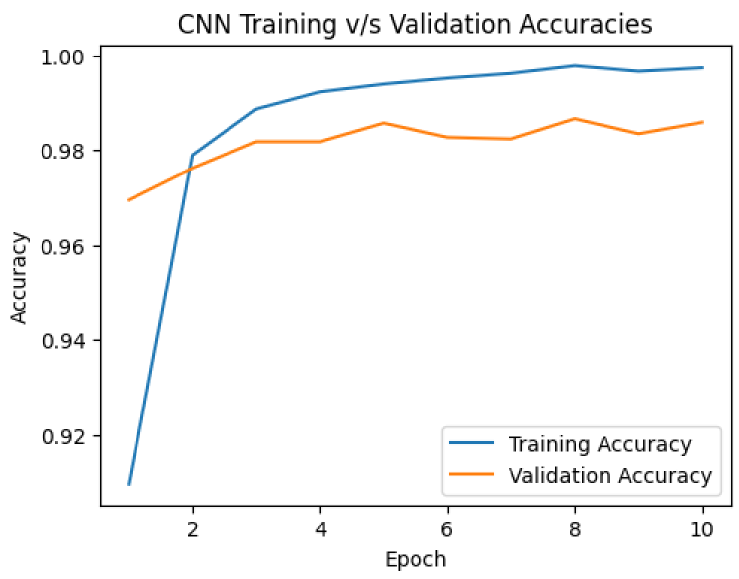


Figure 3: CNN Training vs Validation Accuracies

In the final evaluation, the best ConvNet model achieved a test accuracy of 96.34%, outperforming the best MLP model, which reached 94.08%. This improvement can be attributed to the CNN's ability to capture local spatial features and its use of weight sharing, which together enhance generalization by reducing the number of parameters.

The data was processed again differently for the CNN. The input images were first reshaped to preserve their spatial structure by converting them to dimensions $28 \times 28 \times 1$. The pixel values, originally in the range $[0, 255]$, were then converted to float and scaled to $[0, 1]$ by dividing by 255.0. This normalization helps stabilize training and improve convergence. Finally, the class labels were one-hot encoded to facilitate multi-class classification.

Moreover, the CNN architecture is naturally better suited for image data, as it leverages translational invariance. This property allows the network to detect important patterns regardless of their spatial location in the input, thereby learning more robust and discriminative features.

Overall, these experiments demonstrate that a convolutional neural network converges more rapidly and achieves higher classification accuracy compared to a standard MLP. The results underscore the advantage of using CNNs for image classification tasks, where capturing local dependencies and hierarchical feature representations is crucial.

4. Discussion and Conclusion

Our experiments indicate that network architecture and hyperparameter choices are critical to achieving high classification accuracy. The MLP models demonstrate how increasing depth and incorporating non-linearities can improve performance. Indeed, two hidden layers with 256 units, the largest number of hidden units that were used, performed the best. However, it is important to note the risks of overfitting when adding too many hidden units.

The ConvNet provided a robust alternative with potentially lower training times, however, its design is incredibly complex and sophisticated, whereas MLP was easily implemented from scratch. However, one of the main challenges was the lengthy runtime for running MLPs.

For future work, it would be beneficial to weigh accuracy against runtime and computation. Furthermore, to continue the comparative analysis of MLPs and CovNet it would be beneficial to utilize different datasets (other than image classification) to explore the advantages of MLP over ConvNet.

5. Statement of Contributions

Leo Falvo - Implemented MLP class and check_grad function.
Aditi Potnis - Performed experiment 1, 3, and 4. Worked on the writeups for each experiment. Kynda Nashif - Worked on data preprocessing, exp 2, and written report.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition, 1998.

[2] Asanobu Kitamoto Alex Lamb Kazuaki Yamamoto David Ha Tarin Clanuwat, Mikel Bober-Irizar. -.