COMP551 - Assignment 3 Report

Abstract

In this project, we trained and evaluated two neural network architectures, Multilayer Perceptrons, and Convolutional Neural Networks, on the Kuzushiji-MNIST dataset, with the goal of identifying the most effective model and hyperparameter configuration for image classification. We conducted a series of experiments to compare depth, width, learning rates, regularization strength, and activation functions in order to approach a seemingly ideal model. Our results showed that CNNs performed similarly or slightly worse than MLPs on the test set, and that, within the MLP architecture, an increased number of hidden layers, each with an increased number of hidden units, here 2 layers with 256 units each, and Leaky ReLU as a choice of activation function yielded the best results.

Introduction

The goal of this project was to compare and evaluate the performance of MLPs on the Kuzushiji-MNIST dataset, with a comparative study on the performance of CNNs on the same data. The mentioned KMNIST dataset serves as a benchmark in image recognition, having been introduced as a drop-in alternative to MNIST for handwritten character recognition in Japanese script[1]. Each sample consists of a 28×28 grayscale image of a character from one of ten Hiragana classes, and the task is to classify the image correctly. The image input was flattened into a 784-unit input layer, with standardization applied. We implemented and trained both MLP and CNN models built upon these inputs, evaluating the effects of architectural design, namely layers and width, and key hyperparameters, namely learning rate, and regularization factor.

We investigated tuning the structure of MLPs by varying the number of hidden layers and the width of each layer. We found that, of the structures proposed, two hidden layers, each with 256 hidden units, yielded the best performance, achieving a test accuracy of 0.8990, compared to 0.6952 for the no-hidden-layer baseline and 0.8806 for the following best, two hidden layers, each with 128 units. This confirmed our hypothesis that complex model architecture improves variance and is key for an accurate model. We tested different learning rates and selected 0.1 as the most effective for the general MLP model. For experiments involving L2 regularization, we used a lower factor of 0.0001 to improve stability, and we found that L2 helped reduce overfitting without significantly impacting test accuracy, which remained high at 0.9039 on a model using ReLU activation.

Other activation functions were also compared, and Leaky ReLU was found to outperform Sigmoid in final accuracy. Specifically, the Leaky ReLU MLP reached a test accuracy of 0.8989, compared to 0.7872 with Sigmoid. Finally, a CNN with three convolutional layers and a fully connected layer of 32 units achieved a test accuracy of 0.9269, which slightly bested the best MLP configuration by a margin of 2.30 percentage points. This confirms that CNNs are generally equally as, if not more effective as MLPs for image classification tasks due to their spatial feature extraction capabilities[2].

[1] Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., & Ha, D. (2018). Deep learning for Classical Japanese literature. *arXiv.org*. https://doi.org/10.20676/00000341
[2] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Datasets

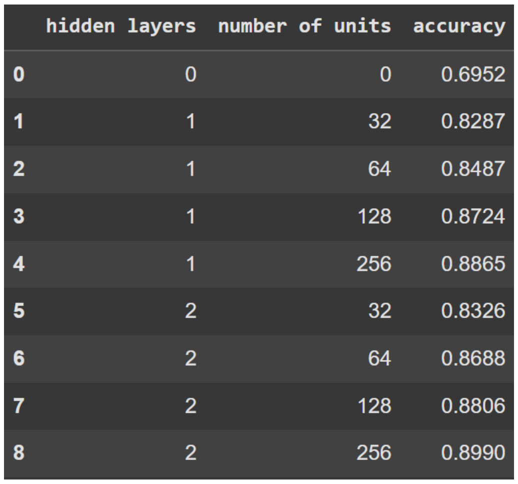
The Modified National Institute of Standards and Technology (MNIST) is a large database of handwritten digits that is commonly used for training and testing machine learning models to classify images. Our dataset, the Kuzushiji-MNIST (K-MNIST), is a substitute dataset of the classic MNIST dataset, and it contains grayscale images of handwritten Japanese characters written in an older cursive handwriting called Kuzushiji. Kuzushiji is written using intricate strokes and patterns more complex than simple digits, and thus, K-MNIST was created to provide a dataset that is more difficult to classify than the original MNIST dataset.

In total, the dataset consisted of 70,000 images with 60,000 in the training set and 10,000 in the test set, which belonged to one of the ten classes, each representing a different Hiragana character that an image would map to. To understand the dataset better, we first examined the class distribution in the training and test set. The dataset was perfectly balanced, with exactly 6,000 images per class in the training set and 1,000 per class in the test set. Compared to the digits in MNIST, the K-MNIST characters were much more complex and varied in structure, allowing us to stress test the full capabilities of our MLP and CNN models against this more challenging dataset.

Results

Experiment 1

Below is a table summarizing the results of the first task, which compared different numbers of layers of MLP's, along with several different options for the number of hidden units per layer. A learning rate of $\alpha = 0.1$ was selected through hyperparameter tuning on a held-out validation set.



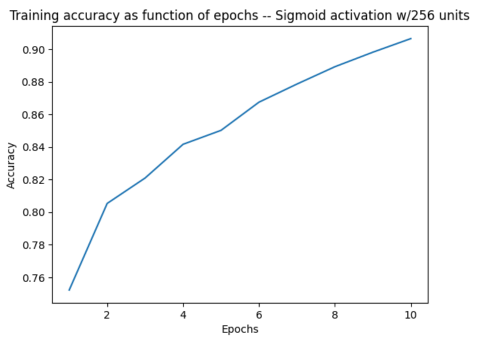
	hidden layers	number of units	accuracy
0	0	0	0.6952
1	1	32	0.8287
2	1	64	0.8487
3	1	128	0.8724
4	1	256	0.8865
5	2	32	0.8326
6	2	64	0.8688
7	2	128	0.8806
8	2	256	0.8990

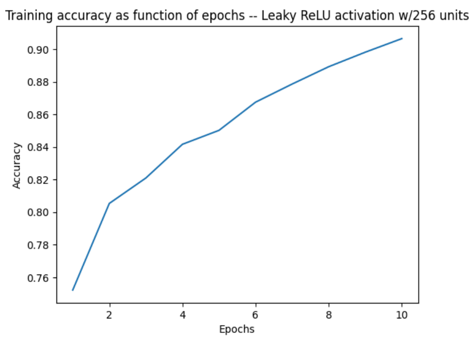
Figure 1: Table displaying the effects of various values for number of (hidden) layers as well number of units per layer on test set accuracy

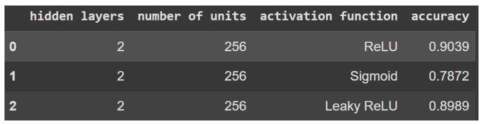
From the above table we can make some decisive conclusions. Notably, the fact that increasing the number of units within each hidden layer increased test accuracy by significant margins (5% in the case of 1 hidden layer, and 6% in the case of 2 hidden layers). It is also reasonable to conclude that the presence of hidden layers themselves helps the model's performance to a great extent, as demonstrated through the table above. By adding just one hidden layer, we increased the test accuracy by 12%, from ~70% to ~83%. Both of these observations correspond to our intuitions; by introducing hidden layers to a model that has none, we are allowing the model to learn non-linearity, which increases model complexity and allows us to better capture patterns in the data. Similarly, increasing the number of units within a hidden layer allows there to be more weights updated at each layer, therefore allowing the model to become more precise in its classification without overfitting.

Experiment 2

In this experiment we tested the effects of different activation functions on the model's test accuracy, with the number of layers held constant at 2, and number of units per layer held constant at 256.







	hidden layers	number of units	activation function	accuracy
0	2	256	ReLU	0.9039
1	2	256	Sigmoid	0.7872
2	2	256	Leaky ReLU	0.8989

Figures 2-3: Plots of training accuracy with sigmoid activation function (top left), and Leaky ReLU activation function (top right), respectively

Figure 4: Table comparison of testing accuracies of different activation functions

From the above results we observe that although both the sigmoid and Leaky ReLU activation functions perform similarly in terms of training accuracy, the sigmoid activation function experiences great generalizing to the test set, with an accuracy of ~79%, while the Leaky ReLU activation function's performance is on par with that of the regular ReLU activation function (having an accuracy of ~90%), which is the "gold standard", so to speak, in our case. This indeed lines up with our expectations, since the sigmoid activation function is a saturating function, meaning that output values tend to "cluster" near 0 or 1. In the case of values clustering near 0, this leads to what we know as the vanishing gradient problem, where having several gradients near 0 leads to an infinitely small final gradient when they are multiplied together, leading to inaccuracies in classification.

Experiment 3

Next, we ran an experiment to observe the impact of L2 regularization on the performance of our model. A regularization strength of $\lambda = 0.0001$ was chosen, again through hyperparameter tuning on a held out validation set. Other training parameters were selected based on previous experiments (i.e. the number of layers and units was chosen based on Experiment 1, and the activation function was chosen based on Experiment 2)

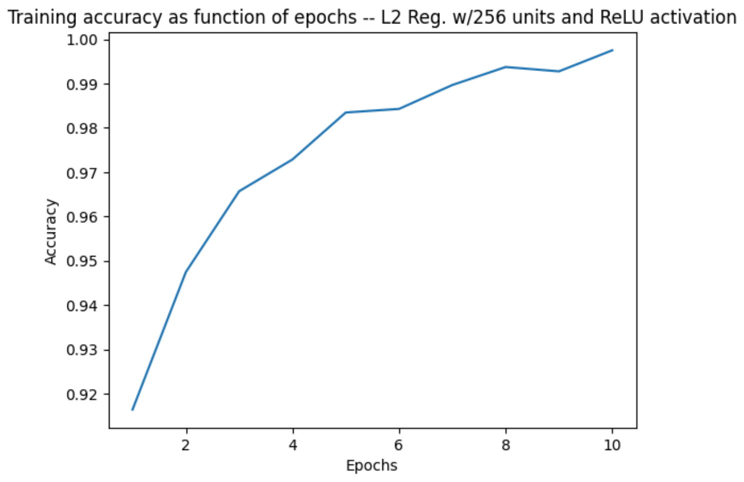


Figure 5: Training accuracy of a 2 layer, 256 unit MLP with L2 regularization as a function of training epochs

After evaluating this model's performance using the test set, we obtained an accuracy of 90.09%. This was actually a slight decrease from the best result of the previous experiment, that being an accuracy of 90.39% with the same model but without L2 regularization. We could speculate about the cause of such decrease; one possibility is that the specific λ value chosen was "incompatible" with the test set – perhaps the regularization strength was not strong enough, or, it was *too* strong. However, with such a small difference between accuracies, it is hard to say exactly what led to the decrease. The good news is that the accuracies are practically identical, which is to say that at the very least the model didn't perform significantly worse as a result of regularization.

Experiment 4

Lastly, we used a convolutional neural network as a means of comparison to see how well the MLP's performance was in relation. This CNN was initialized with 32 units per hidden layer (of which there were two), and 3 convolutional layers. The layer size of 32 was found to be the best through hyperparameter tuning on a held out validation set.

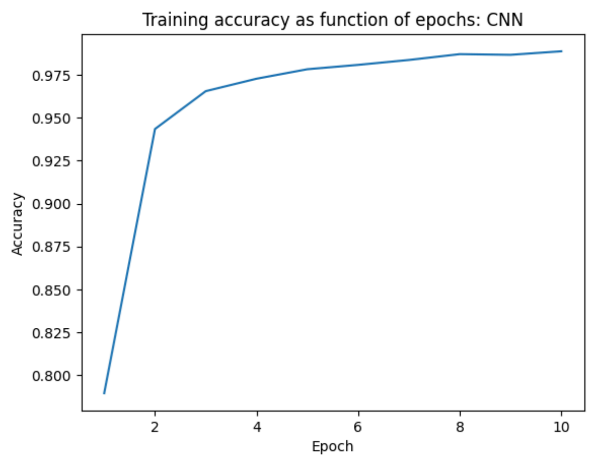


Figure 6: CNN training accuracy as a function of training epochs

After training and testing, an accuracy of 92.69% was achieved on the test set. This suggests that a CNN performs slightly better than our "best" MLP model. Intuitively, this should come as no surprise, since convolutional neural networks allow us to capture templates/patterns within images, and thus more accurately classify images as a result. Another possible explanation is the alternative view of a convolutional neural network as a partially connected MLP, instead of a fully connected MLP – by avoiding fully saturated connections between layers, we can reduce the likelihood that our model overfits and thus generalizes poorly (or, in our case, less well) to a test set.

Although the convolutional neural network did perform better than our best MLP model, this doesn't come without a trade-off, that trade-off being time. Whereas our MLP models trained in ~1 minute, sometimes even less, the CNN took > 10 minutes to train for 10 training epochs. So, in a world where resources were infinitely abundant and didn't cost a penny, the CNN would win every time. However, that is not the case, we need to strike a balance between performance and efficiency/cost effectiveness.

Discussion & Conclusion

To summarize the results from our experiments, we found that increasing the depth and width of our MLP led to consistent improvements in accuracy. Introducing hidden layers enabled the model to learn non-linear representations, and larger hidden layers allowed for more expressive power. Activation functions also had a notable impact - sigmoid activations performed poorly relative to ReLU and Leaky ReLU, which showed strong generalization. Interestingly, L2 regularization triggered lower training stability, though its effect on final performance was practically negligible. CNNs were also evaluated as a point of comparison. Our CNN model beat our best MLP model, which is in line with our expectations, given its ability to capture spatial structure. One significant drawback we encountered with CNNs was the amount of time it took to train the model. Our experiment with CNN took over ten minutes, which was at least ten times longer than the time it took to train our MLP. Thus, to choose between the two models, it may be worthwhile to consider the tradeoffs between performance and computational time. In conclusion, although CNN offers stronger performance for image classification tasks, a well-tuned MLP can be the more practical choice for larger scale tasks while being just as effective as CNN. To further explore this study, future work could try to further optimize our CNN model by tuning other hyperparameters like number of filters, stride and padding, and our MLP model by examining the effect of different dropout rates on fully connected layers.