Convolutional Neural Network Models to Classify Hiragana Characters in Kuzushiji-MNIST Data Set

MATH 156

Oi Ting Cheung (UID: 005544379), Benjamin Khothsombath (UID: 505155800),
Nathan Yuen (UID: 305100413), James Chen (UID: 605328259)

March 23, 2022

1 Contributions

Benjamin Khothsombath: Creates and improves the model
Nathan Yuen: Improves the model and graph plotting
James Chen: Typed the report
Oi Ting Chenug: Typed the report

2 Abstract

We developed and trained several convolutional neural networks (CNN) to classify handwritten digits from the Kuzushiji-MNIST data set. We started from a baseline model, and implemented strategies and techniques introduced by Tsai on their research in CNNs for Japanese characters for a different data set. We further improved on Tsai's models, inserting new layers into our model and adding batch normalization and regularization. We trained our models for 10 epochs. We implemented two models inspired by Tsai: M5 and M7*. Our regularized M7* model, was able to distinguish classes with an accuracy of 97.59% on the test data set. The high accuracy of our model indicates that the model we created was well-suited for our task. We note that further improvements to accuracy may be found with more epochs and/or more computational power.

This report will demonstrate the most appropriate technique in getting the most accurate model on our data set. The "Data Overview and Cleaning" part will provide a brief description of the Kuzushiji-MNIST data set and types of prepocessing we did on the data set. The "Methodology" part will provide the methods and techniques we tried to improve the accuracy of the model. Results part will provide the performance of the model under all the modifications and techniques we applied. The "Discussion" part will discuss the difference between the research paper we replicate and our model. We will also discuss the limitations to our model and some possible solutions in that section. The "Conclusion" part will provide the techniques we decided to apply in creating the best model.

The code to our project can be found at `https://github.com/n-yuen/math156-project`.

3 Data Overview and Cleaning

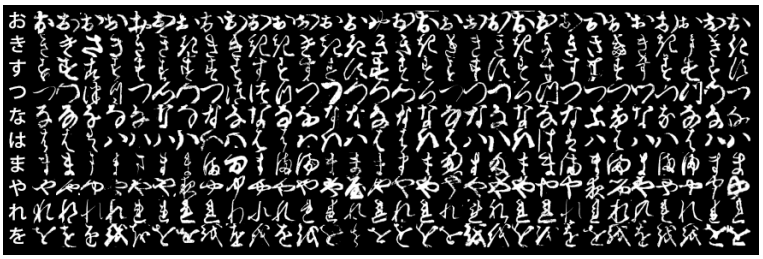

Figure 1: Kuzujishi-MNIST data set

The Kuzushiji-MNIST data set (3) is a large data set containing hand-written Hiragana characters. Hiragana is a phonetic lettering system for the Japanese language, in which each letter represents a different phonetic sound. The Kuzushiji-MNIST data set contains 70,000 images with 28 $\times$ 28 grayscales, representing 10 different Hiragana characters. Kuzushiju-MNIST data set restricts the user to choose 10 Japanese characters in creating 10 classes (rows) of characters in Kuzushiji style. The data set is already originally split into train and test data, with 60,000 images reserved for training and 10,000 reserved for testing. The data set is balanced, with a roughly equal amount of images belonging in each classification class.

4 Methodology

We chose to use a Convolutional Neural Network (CNN) due to its widespread use in computer vision. It is mainly used in feature extraction and classification such as face recognition and object identification. A CNN takes in images as input, which go through different layers to get our desired output. The convolutional layer slices the images in the data set into smaller images depending on the choice of filter size (3 $\times$ 3), in order to learn the features of the images. Its output will be a 2D-matrix. Because the CNN takes in a 3 $\times$ 3 filter, it will reduce each dimension of the input by 2; for example, a 28 $\times$ 28 image fed into a convolution layer will produce a 26 $\times$ 26 output. The activation layer is where we apply the ReLU function (Rectified Linear Unit). It provides non-linearity without introducing vanishing gradients. The pooling layer then reduces the dimension of the convolutional layer's output, in order to generalize to images where patterns can be found in different sections of the image. The final layer uses a softmax classifier which is is a generalization of binary Logistic Regression classifier (sigmoid function) to multiple classes. The goal of our model is to classify 10 different classes of Hiragana characters. We will further discuss the structure of our model in the following sections.

4.1 Network Architectures

For the convolutional front-end, we started with a single convolutional layer with 32 small filters of size (3,3) followed by a an activation layer and finishing with a max pooling layer. We then repeat the three layers to increase the size of convolutional layer at each depth. At the end, the filter maps can then be flattened to provide features to the dense classifier. There will be 3 fully connected layers before the final layer.

4.2 Baseline Classification Framework

We first built a vanilla CNN for MNIST Handwritten Digit classification from (Brownlee, 2019) from the ground up and re-purposed it to recognize handwritten Japanese characters. This baseline model used 2 convolutional layers, 1 fully connected layer, and an output layer. It achieved an accuracy of 90.75%.

4.3 Modifications on Baseline Model

We originally attempted to use the models found in (Tsai, 2016) directly on our data set. In particular, we attempted to implement model M7-2, which achieved the highest test accuracy in Tsai's Hiragana data set. However, we found that due to the dimensions of our data (which were of size 28×28), where we had significantly fewer dimensions than Tsai's data set (which were of size 64×64), the convolutional and maxpool layers would reduce the dimensions to zero before reaching the output layer. Hence, we attempted to instead implement model M5, which reduced dimension to 1×1 at the output layer. The M5 model is the same as Tsai's M6-2 model but without one of the final fully-connected layer.

The training is performed by optimizing softmax loss using ADAM. ADAM is an optimization algorithm using the stochastic gradient descent method with an adaptive estimation of first-order and second-order moments. Then, weight initializations are performed using batch normalization after each weight layer and before each activation layer. Batch normalization is a technique to train the deep neural network. Each mini-batch standardizes the input to each layer. The training carried out for at least 40 epochs for all models (longer for some). Gird search is then applied on the training. It is a technique that provides the optimal values of the hyperparameters. It turns out that learning rate of 10^{-4} and mini-batch size of 16 leads to largest decrease in training loss in 5 epochs. Learning rate annealed using step factor decay of 0.1 every 20 epochs. The training data consists of 80% training data and 20% cross-validation. The training stops when validation loss and accuracy starts to plateau. We pre-process the remaining images following how the data was trained in the remaining images for testing. Finally, we implemented it in `tensorflow` using the Keras interface.

We further designed a novel neural network, which we name M7*. M7* is similar to M5, but with the insertion of two more convolutional layers. The design comparison is shown in Table 1.

M5	M7*
conv3-64	conv3-64 conv3-128
Maxpool	
conv3-128	conv3-192 conv3-256
Maxpool	
conv3-512	conv3-512
Maxpool	
FC-4096	
FC-10	

Table 1: Neural Network Architecture of Our Improved Models

We also built models using the same M5 architecture, but this time using both l_1 and l_2 regularizers. We found that for both, the optimal value for λ was to use $\lambda = 10^{-6}$, which we selected by building models starting with $\lambda = 10^{-2}$ and reducing λ by a factor of 10 until we got the best accuracy.

4.4 Hyperparameters

Following Tsai, we used a batch size of 16. We trained for 10 epochs, with a 0.1 validation split ratio.

5 Results

Baseline	M5	M7*	Regularized M7*
90.75%	94.85%	97.46%	97.59%

Table 2: Test Accuracy of the Different Models

We noticed some evidence of overfitting. For example, the M7* model's test accuracy is 0.9741 while its training accuracy is 0.9940 and validation accuracy is 0.9882. Thus, we attempted to add regularization to the model. While the regularization did slightly improve the model's test accuracy to 0.9759, it is unclear whether this improvement is due to the regularization, or if it is just by pure randomness, since the improvement is not very large.

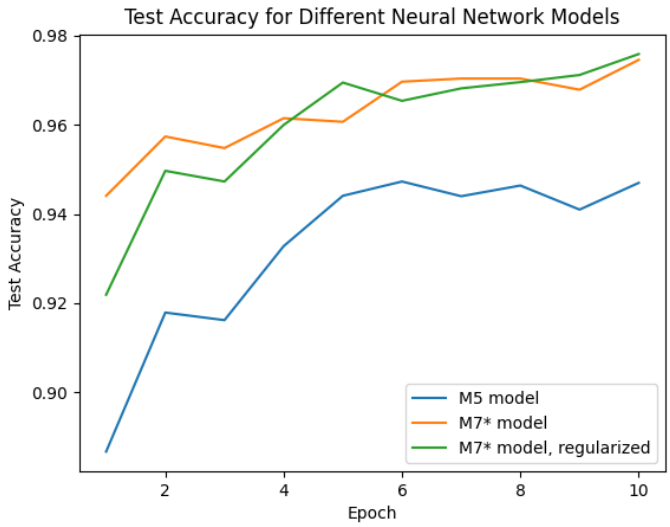

Figure 2: Testing Accuracy for Different Models vs. Epoch

As seen in Figure 2, the accuracy seems to go up as the number of layers increases, as well as the number of epochs. However, we do expect the accuracy to go down after a certain point. We do not reach this point because of our computing limitations.
We also trained models with increased values of λ for regularization. However, these models did not perform as well. For example, the model with $\lambda = 0.00001$ had test accuracy of 0.9622, lower than the accuracy with the other models.

6 Discussion

Tsai's M7-2 model is a CNN with 7 weight layers that was trained for 20 epochs. Our final M7* model differs from this model; although 7 weight layers are also used, they are arranged in a different way. We trained for 10 epochs. Initially, we tried to build the same model as the research paper, but we realized that the research paper takes in 64×64 images, whereas our data set is 28×28. Thus, the dimensions for each layer will not line up exactly, so we used a different number of layers with different sizes.

We found that the M5 model was not sufficiently descriptive on our data set. After 10 epochs, the model attained 0.9470 test accuracy, and accuracy did not significantly change over the next 10 epochs; at epoch 24, it attained 0.9486 test accuracy. This indicates that the model has underfit the data set, i.e. it is not accurate enough. Our M7* model attained much higher accuracy: 0.9741 on the test set after 10 epochs. This increased accuracy came at the cost of significantly increased training time, where each epoch would take about 5 minutes to train on our model as opposed to 1 minute to train for M5. There is some merit to having a model that can be trained in much less time, but given that M7* was only trained for one hour, we do not recommend using M5 over M7*.

We can attribute the increased accuracy of the M7* model to the increased number of convolution layers. By increasing

We used a smaller number of training epochs due to limitations in computation. With more computational power or time, we might be able to train a more accurate model. Our use of `tensorflow` on Windows further limited our ability to train a powerful model, as Windows does not allow `tensorflow` to leverage CUDA technology to utilize the GPU for additional compute power.

Tsai's research did not include any regularization. The research paper's goal was to find the best architecture for classification. We were able to expand on this by trying different values of λ to prevent overfitting. We experimented with different values of λ, and did not find that any λ values significantly improved test accuracy on the data set. This is possibly a result of the relatively large size of the data set, where overfitting is not likely to occur when we are training on so much data. Thus, we do not think it is necessary to run a regularizer.

Lastly, the Kuzujishi-MNIST data set is balanced in that the number of images within each of the 10 output classes is roughly equal. Thus, test accuracy is a good indicator of model performance, as the model's ability to make a prediction is roughly equal between classes.

6.1 Limitations to our Model

We would like to apply more epochs on models for better accuracy. We also tried cross-validation to select more appropriate parameters for regularization. However, both of them requires expensive model building since the models took about 30-50 minutes to build.

7 Conclusion

We were able to train a relatively accurate model fairly quickly. Our best performing model, the regularized M7* model, attained a 0.9759 test accuracy after 10 epochs of training. This indicates that our convolutional neural network was suitably designed for the task of image classification.

After improving upon the baseline model first with a pre-structured architecture drawn from examples in the field and then by original experimentation, we were able to optimize the CNN approach to classification of the Kuzujishi-MNIST data set to achieve a final test set classification accuracy of 97.59%.

8 References

Brownlee, J. (2019, May 7). How to Develop a CNN for MNIST Handwritten Digit Classification. Machine Learning Mastery. `https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/`

Tsai, C. (2016). Recognizing Handwritten Japanese Characters Using Deep Convolutional Neural Networks.

Repository for Kuzushiji-MNIST, Kuzushiji-49, and Kuzushiji-Kanji `https://github.com/rois-codh/kmnist`, 2020