Experimental Deisgn on Hiragana Classification using Multi-layer Perceptron and Convolutional Neural Network

Du Xiang
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA
xd00099@berkeley.edu

Zhiwei Zheng
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA
zhiwei.zheng@berkeley.edu

Jinmeng Zhang
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA
jinmeng_zhang@berkeley.edu

Abstract

In this work, we investigate the effects of model capacity and experimental design on the performance of multi-layer perceptron (MLP) and convolutional neural network (CNN) models for classifying Hiragana characters. We address questions such as: What is the variable the machine learner is supposed to predict? How accurate is the labeling? What is the annotator agreement? What is the required accuracy metric for success? How much data do we have to train the prediction of the variable? Are the classes balanced? How many modalities could be exploited in the data? Is there temporal information? How much noise are we expecting? Do we expect bias?

We train MLP and CNN models on a dataset of Hiragana characters, and vary the model capacity of each type of model by changing the number of hidden layers and the width of each layer in the MLP, and by adding CNN blocks to the model. We evaluate the performance of the models using metrics such as accuracy, precision, and recall.

Our experiments show that increasing the model capacity of both types of models improves their performance, but also increases the risk of overfitting. We also find that the performance of the CNN model is consistently better than that of the MLP model, regardless of the model capacity. In addition to comparing the performance of the MLP and CNN models, we also discuss the factors that affect model capacity, such as the number of layers in the network, the size of the training dataset, and the amount of regularization applied to the model. Our results provide insights into the effects of model capacity on the performance of different types of neural networks for Hiragana classification.

1 What is the variable the machine learner is supposed to predict? How accurate is the labeling? What is the annotator agreement (measured)?

In this project, the machine learner is an image classifier that is designed to predict the Hiragana character represented in a given 28 X 28 grayscale image. The data used for training and evaluating the model comes from the Kuzushiji-MNIST dataset, which consists of 70,000 images [1]. Same as the MNIST split, the training set consists of 60,000 images and the testing set consists of 10,000 images. Each image is labeled with one of the 10 rows of Hiragana characters, and the classes are balanced across the dataset.

It is reasonable to assume that the labeling of the images in the K-MNIST dataset is highly accurate, as the dataset has been widely used in previous research on Hiragana character classification. In addition, the Hiragana characters are well-defined and universally agreed-upon categories, which reduces the potential for labeling errors.

It is also likely that the annotator agreement for the K-MNIST dataset is high, as the images are labeled with specific rows of Hiragana characters. This clear and well-defined labeling scheme should result in high agreement among annotators. Additionally, the use of a widely accepted and standardized dataset like Kuzushiji-MNIST further increases the likelihood of high annotator agreement.

2 What is the required accuracy metric for success? How much data do we have to train the prediction of the variable? Are the classes balanced? How many modalities could be exploited in the data? Is there temporal information? How much noise are we expecting? Do we expect bias?

For this project, we will use the accuracy metric to evaluate the performance of the machine learning model. This metric is appropriate for this study because the dataset is balanced and it provides a realistic measure of the model's performance.

The data used for training and evaluating the model comes from the KMNIST dataset, which consists of 60,000 training images and 10,000 test images. This dataset should provide sufficient data to train the model and achieve good performance on the classification task.

The classes in the KMNIST dataset are balanced, with an equal number of images for each of the 10 rows of Hiragana characters. This is important for ensuring that the model is not biased towards any particular class.

The data in the KMNIST dataset consists of 28 X 28 grayscale images, which represent a single modality of data. While additional modalities could potentially be exploited, such as color information or spatial relationships between characters, the use of grayscale images is sufficient for the purposes of this study. The KMNIST dataset does not contain temporal information, as the images are not ordered in any particular sequence. This means that the model will not be able to make use of any temporal relationships between images.

However, the KMNIST dataset may contain some noise, as the same characters can be written in different ways (modern or traditional). This could result in slight variations in the appearance of the characters, which could affect the performance of the model. Despite this potential source of noise, most of the writings are pretty consistent and should not introduce intrinsic biases, and the KMNIST dataset should provide a good basis for training and evaluating the model.

3 What is the Memory Equivalent Capacity for the data (as a dictionary). What is the expected Memory Equivalent Capacity for a neural network?

One of the key challenges in image classification is designing a model with sufficient capacity to learn and represent the complex patterns in the data. Model capacity refers to the ability of a model to capture the underlying structure of the data, and is influenced by factors such as the number of layers in the network, the size of the training dataset, and the amount of regularization applied to the model. In this section, we estimate the Model Equivalent Capacity (MEC) of the KMNIST dataset to provide a baseline for comparing the performance of different types of models.

To estimate the capacity of the KMNIST dataset, we first compute the number of parameters in a model that can perfectly fit the training data, assuming no regularization is applied. This provides an upper bound on the capacity of the dataset, and gives us an idea of how complex the patterns in the data are. We compute the upper bound of MEC by counting the algorithm of all possible instances given by a MNIST picture. We get 47,160,001 bits from the following equations:

$$\begin{aligned} &MEC(upperbound) \\ &= (minthreshs * (d+1)) + (minthreshs + 1) \\ &= (60000 * (784+1)) + (60000+1) \\ &= 47,160,001 bits \end{aligned}$$

We then compute the expected MEC to be about 12,477 bits by taking the log_2:

$$\begin{aligned} &MEC(expected) \\ &= (log_2(minthreshs) * (d+1)) + (log_2(minthreshs) + 1) \\ &= (log_2(60000+1) * (784+1)) + (log_2(60000+1) + 1) \\ &= 12,476.94 bits \end{aligned}$$

where 784 denotes 28 X 28 (image dims), 60000 denotes 60000 thresholds in worst case.

Figure 1: KMNIST example data

Since the expected MEC is a more reasonable estimate, we will use 12,477 bits as the expected capacity of KMNIST dataset. This indicates that the capacity of patterns in the data are about 12,477 bits, and that we expect a model with about 12,477 bits of MEC would be able to memorize the dataset.

4 What is the expected generalization in bits/bit and as a consequence the average resilience in dB? Is that resilience enough for the task? How bad can adversarial examples be? Do we expect data drift?

We first calculate the generalization and the average resilience according to the formulas below, based on the result of CNN model whose structure is [128, 16, 16, 10].

$$Generalization = \frac{CorrectlyClassfiedInstances}{MEC}$$

$$Resilience = 20 * log10G[dB]$$

The results are 14.8393 bits/bit and 23.4283 dB. We believe the resilience is enough for the task. Due to the uniformity of the dataset, we don't expect there would be too many adversarial examples. Some adversarial examples in the test datasets are that characters which are inherently scribbled. Also, we don't expect data drift because the dataset we used are already low-resolution and gray-scale ones. It might can but we don't expect too much.

5 Is there enough data? How does the capacity progression look like?

Here we use the following formula to calculate the upper bound Memory-equivalent Capacity expected for a three-layer neural network of subsets(1%, 2%, 5%, 10%, 20%, 30%, 50%, 80%, 100%). When we create a subset, we take an equal number of samples from each category.

$$MEC(expected) \\ = (log_2(minthreshs) * (d+1)) + (log_2(minthreshs) + 1)$$

For instance, We here show how we calculate the MEC of the subset consisting of only 1% raw data. In worst case, the minimum number of thresholds would then be 600, and the dimension remains as 784. The result would then be 7256.73. We achieve the following figure by doing the similar calculation.

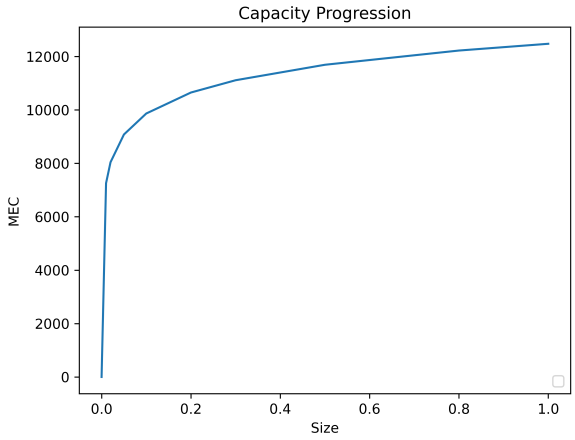

Figure 2: Capacity Progression

As the size grows larger, the MEC grows slower and the curve gets gradual. Since the slope of the curve is almost equal to zero, we would say there is enough data for us to use.

6 Train your machine learner for accuracy at memory equivalent capacity. Can you reach near 100% memorization? If not, why (diagnose)?

In this project, we trained our model from simple to complex structures, ranging from small MEC to large MEC. The accuracy results of each structure are shown in the table below. As shown before, the expected MEC of our model is 12,476.94 bits. For the MLP with the architecture of [784, 16, 10], the model MEC is 12,576 bits, which we would expect the model to completely overfit/memorize the training set. Similarly, when we the structure of the model as [784, 16, 16, 10], where the model MEC is 12,592, we would also expect perfect memorization. After stress training on the training data; however, in these situations, the accuracy of the training set is about 92%, which means we are not reaching 100% memorization.

As we examine through the dataset, we found that this could be due to the biases in the dataset, where some of the calligraphy differs by a lot with the same labels. This could result in a more difficult task for the machine learner to be trained. It could also happen due to the poor optimization. In out experiment set up, we only used a fixed learning rate to train our model. Without using a dynamic and more responsive optimizer, the machine learner could fall into local maximums and fail to reach the global optimum. Lastly, the MEC we estimated was the expected MEC, and it could happen that at expected MEC, the model cannot reach 100% memorization.

Structure	Accuracy(test)	Accuracy(train)	Model MEC/bits
[784, 8, 10]	0.6567	0.8457	6,288
[784, 16, 10]	0.7203	0.9216	12,576
[784, 32, 10]	0.7838	0.9946	25,152
[784, 64, 10]	0.8243	0.99+	50,304
[784, 8, 8, 10]	0.6535	0.8463	6,296
[784, 16, 16, 10]	0.7235	0.9249	12,592
[784, 64, 64, 10]	0.8245	0.99+	50,368
[784, 128, 128, 10]	0.8516	0.99+	100,736

Table 1: FCN

7 Train your machine learner for generalization: Plot the accuracy/capacity curve. What is the expected accuracy and generalization ratio at the point you decided to stop? Do you need to try a different machine learner? How well did your generalization prediction hold on the independent test data? Explain results. How confident are you in the results?

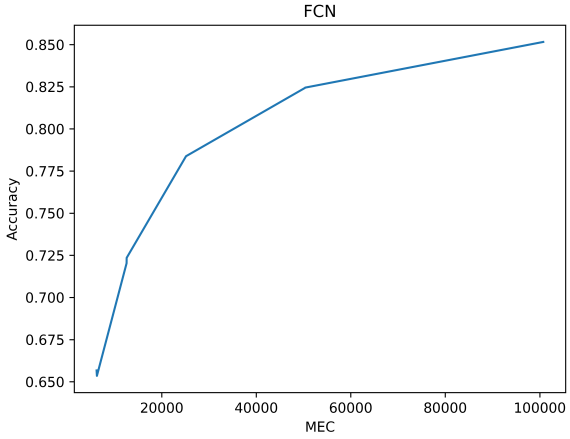

Figure 3: FCN Generalization

One point we believe is good enough is when MEC is set to 50,368 and the corresponding accuracy is 0.8245. One reason is that before this point, the accuracy still increases rapidly as the MEC of the network grows larger, and after it, the cost of generalization is huge. Even after doubling the suggested MEC, the accuracy only improves a little, from 0.8245 to 0.8516.

Out of curiosity, we built several Convolution Neural Networks and train and finally evaluated them on the test dataset like what we did before. The convolution part is shown as follows.

```
# Model
model = Sequential()
# Add convolution 2D
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu', padding="same",
        kernel_initializer='he_normal',input_shape=(IMG_ROWS, IMG_COLS, 1)))

model.add(BatchNormalization())

model.add(Conv2D(32,kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(32,kernel_size=5,strides=2,padding='same',activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(BatchNormalization())

model.add(Conv2D(64, kernel_size=(3, 3), strides=2,padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size=(3, 3), strides=2,padding='same', activation='relu'))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
```

Figure 4: Convolution Part

After convolution layers, each image would be converted into a vector of length 125. Similar to the table above, the accuracy of each structure of decision networks is shown in the table below. What is worth mentioning is that because we have already compressed the input information, we no longer need a large-MEC network to do the classification. The table shows though the performances on training dataset might not be so good as the former network, the performances on the test dataset are much higher, proving a better generalization.

Structure	Accuracy(test)	Accuracy(train)	Model MEC/bits
[Convs, 128, 8, 10]	0.9270	0.9747	1,040
[Convs, 128, 16, 10]	0.9376	0.9780	2,080
[Convs, 128, 32, 10]	0.9399	0.9825	4,160
[Convs, 128, 64, 10]	0.9425	0.9803	8,320
[Convs, 128, 8, 8, 10]	0.9355	0.9767	1,048
[Convs, 128, 16, 16, 10]	0.9363	0.9797	2,096
[Convs, 128, 64, 64, 10]	0.9439	0.9797	8,384
[Convs, 128, 128, 128, 10]	0.9345	0.9800	16,768

Table 2: CNN

The corresponding generation figure is shown as Fig. 5. In this case, we would choose the (8384, 0.9439) as the stop point for the same reason as before. These experiments also verify that no matter how the structure of network is, as long as its MEC remains the same, the performance would also then be similar. We think the generalization prediction did quite well and we are very confident about our results, after so many experiments.

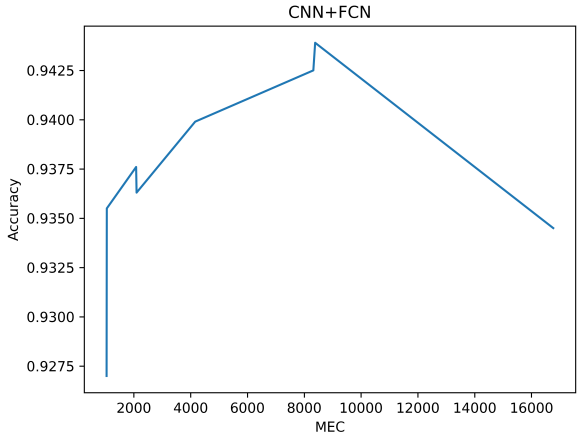

Figure 5: CNN Generalization

8 Comment on any other quality assurance measures possible to take/the authors should have taken. Are there application-specific ones? If time is present: How did you deal with it?

In our implementation, we used 10 fold cross-validation to assess the generalization performance of our models and prevent over fitting. This involved dividing the data set into 10 folds, training the model on 9 folds and evaluating it on the remaining fold. This allowed us to obtain a better estimate of the model's performance that is less sensitive to the specific split of the data by averaging across the fold performances.

We also used early stopping to prevent overfitting by interrupting the training process when the model's performance on the validation set stopped improving. This helped to ensure that the model did not overfit to the training data and was able to generalize well to unseen data.

Things we could have taken include more data augmentation to artificially increase the size of the dataset and also increase the model's generalization ability, more regularization methods such as weight decay or dropouts, and more model and hyperparameter opitmization using methods like grid search or random search.

As for application specific ones, if our dataset contains imbalanced classes, we could have used a different metric, such as measure in entropy, to measure the performance of the model instead of accuracy. Or, we could do oversampling or undersampling to balance the class distribution before training.

If time is present, we could incorporate long sequence models such as Recurrent Neural Networks(RNNs) or Long short-term memory (LSTMs) to process time related sequential data.

9 How does your experimental design ensure repeatability and reproducibility?

The repeatability and reproducibility are guaranteed by several facts. The first thing is that the dataset KMNIST we use in this project is public to all, everyone has the access to it. Second, we have set the random seed to a fixed number so as to guarantee the reproducibility. Third, we adopt ten-fold cross-validation to decrease the bias in the experiments. Last but not least, we upload our code to GitHub (https://github.com/xd00099/ML-Experimental-Design), everyone can download and reproduce our experiments.

10 Team contributions

Du Xiang: CNN Implementation and Training, Calculate MEC for CNN(FCN), Results visualization and Analysis, Collaboration on report and question answering.

Zhiwei Zheng: Multi-layer Perception Implementation and Training, Calculate MEC for MLP, Results visualization and Analysis, Collaboration on report and question answering.

Jinmeng Zhang: Multi-layer Perception Implementation and Training, Calculate MEC for MLP, Results visualization and Analysis, Collaboration on report and question answering.

References

[1] Tarin Clanuwat et al. *Deep Learning for Classical Japanese Literature*. cite arxiv:1812.01718Comment: To appear at Neural Information Processing Systems 2018 Workshop on Machine Learning for Creativity and Design. 2018. DOI: 10.20676/00000341. URL: http://arxiv.org/abs/1812.01718.