ST: Python for Machine Learning (CS504)

Fall 2018

James Young

Department of Computer Science

Task Description

The purpose of this assignment is to implement a machine learning algorithm in order to attempt to train a model that can recognize the Japanese Hiragana character set. For this assignment, the ETL-4 dataset will be used. The ETL-4 dataset contains handwritten hiragana samples and is ideal for the goal of this project. This dataset can be found here.

Method Description

The chosen approach for this problem will be to implement a Convolutional Neural Network (CNN). As image recognition is at the core of solving this problem, a CNN should be well suited for such a task. Additionally, an artificial increase of the dataset will be employed. Small translation, rotation, and scaling transformations will be applied to the preexisting dataset to generate more data samples in ways that would be normally seen in varying handwriting.

The parameters that will be looked at will primarily be the complexity of the CNN model. Parameters of the image transformations will also be included but were not varied in this project. It should also be noted that only a limited number of configurations were able to be studied, due to the time it takes to train models, as well as the limited time provided to complete this assignment. Listed below are the configurations of the CNN as well as the parameters of the image transformations.

Model 1	Model 2	Model 3	Model 4
Conv2D (64)	Conv2D (32)	Conv2D (32)	Conv2D (64)
Max Pooling (2, 2)	Leaky ReLU (alpha = 0.1)	Leaky ReLU (alpha = 0.1)	Leaky ReLU (alpha = 0.1)
Dropout (0.4)	Max Pooling (2, 2)	Max Pooling (2, 2)	Max Pooling (2, 2)
Flatten	Dropout (0.25)	Dropout (0.25)	Conv2D (128)
Dense (128)	Conv2D (64)	Conv2D (64)	Leaky ReLU (alpha = 0.1)
	Leaky ReLU (alpha = 0.1)	Leaky ReLU (alpha = 0.1)	Max Pooling (1, 2)
	Max Pooling (2, 2)	Max Pooling (2, 2)	Conv2D (192)
	Dropout (0.25)	Dropout (0.25)	Leaky ReLU (alpha = 0.1)
	Conv2D (128)	Conv2D (128)	Max Pooling (2, 1)
	Leaky ReLU (alpha = 0.1)	Leaky ReLU (alpha = 0.1)	Conv2D (256)
	Max Pooling (2, 2)	Max Pooling (2, 2)	Leaky ReLU (alpha = 0.1)
	Dropout (0.4)	Dropout (0.4)	Max Pooling (4, 4)
	Flatten	Conv2D (256)	Flatten
	Dense (128)	Leaky ReLU (alpha = 0.1)	Dense (256)
	Leaky ReLU (alpha = 0.1)	Max Pooling (2, 2)	Leaky ReLU (alpha = 0.1)
	Dropout(0.3)	Dropout (0.4)	Dense (256)
		Flatten	Leaky ReLU (alpha = 0.1)
		Dense (128)	
		Leaky ReLU (alpha = 0.1)	
		Dropout(0.3)	
Softmax			

Fig. 1. Model Configurations

Model Summaries	
Model	Description
1	Very simple CNN with only 1 layer
2	More sophisticated CNN with more layers and Dropout
3	An extension of Model 2, with an additional layer on the end
4	A model with many convolutional layers

Fig. 1.1. Model Description Summary

Parameter	Value
Max Rotation Angle	1 degree
Width Shift Range	5% of total Width
Height Shift Range	5% of total Height
Shear Range	10% of
Zoom Range	90% - 110%
Fill Mode	Nearest

Fig. 2. Image Transformation Parameters

Experimental Results

Data on loss and accuracy across epochs were recorded and graphed. Additionally, refer to Fig. 5 for a summary of each model. For all runs, batch size = 256, and epochs = 40.

Fig. 3. Loss over Epochs

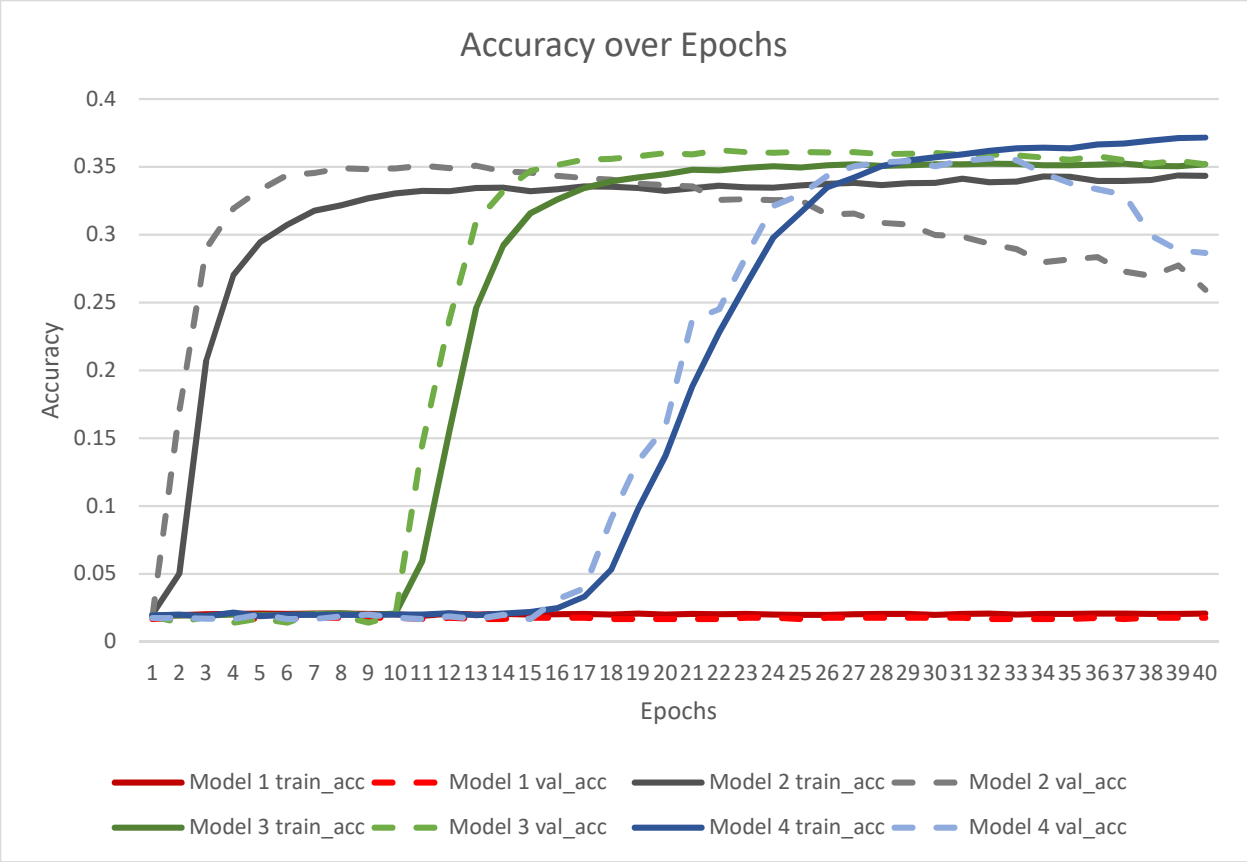

Fig. 4. Accuracy over Epochs

Results			
Model 1	Model 2	Model 3	Model 4
Test Loss: 3.9355	Test loss: 3.6848	Test loss: 3.3748	Test loss: 3.7603
Test accuracy: 0.0179	Test accuracy: 0.2502	Test accuracy: 0.3507	Test accuracy: 0.2784
Max Valid. Acc: 0.0176	Max Valid. Acc: 0.3509	Max Valid. Acc: 0.3623	Max Valid. Acc: 0.3561

Fig. 5. Summary of Results

Conclusion

As it should be apparent, Model 1 did very poorly, but every other model performed in quite a comparable manner. Model 3 appears to have done the best, beating Models 2 and 4 by about 1% in test accuracy. I believe that this is a result of extending the model of 2 and adding more convolutional layers, thus increasing the ability of the model to generalize the model. Additionally, Model 4 likely did as well as it did due to the large number of layers it has.

Unfortunately, having a maximum of 36% correctness when guessing characters is nowhere near ideal. In performing this research, I ran in the very real issue of computational limits, in that every run performed took more than an hour (excluding model 1). If I had either more time or more computational resources, I would increase the number of layers that the model has, as well as figuring out more techniques (such as something in the preprocessing stage) that could result in a higher accuracy.