BUSINESS ACADEMY SOUTHWEST

MACHINE LEARNING

MINI PROJECT

EMNIST Digits Image Classifier

Author
Tobias RINGHOLM

March 28, 2022

1 Introduction

In the following report and belonging Jupyter notebook there will be made an examination of the EMNIST Digits training and test datasets provided by Kaggle [1]. The object of the Jupyter notebook is to develop and train a machine learning solution based on the training dataset to make the solution able to recognize handwritten digits and classify the test dataset correctly. This will be done by using the multilayer perceptron machine learning architecture.

2 Chosen Dataset

The EMNIST Digits dataset provide balanced handwritten digit datasets. The total dataset contains 280.000 samples which are divided into a training set and a test set. The training set contains 240.000 samples while the test set contains 40.000 samples. There are 10 different classes in this dataset.

I decided to install emnist by using:

```
python3 -m pip install emnist
```

This is to download and load the dataset within the Jupyter notebook.

3 Model Architecture

The multilayer perceptron was made with one input layer, one or more hidden layers and one output layer. The input layer was made with *Flatten* such that each image is converted from a 2-dimensional array to a 1-dimensional array. The hidden layers was made with *Dense* to make each node fully connected. The output layer contain 10 nodes because there are 10 different classes in our dataset. Since this is a multilayer classifier the *Softmax* activation function is used in the output layer.

The model was compiled with the loss function *sparse_categorical_crossentropy* and *momentum optimization* as optimizer because of the same reason as above. The *momentum optimization* adds momentum to the *Stochastic Gradient Descent* such that previous gradients contribute to the computation of a new gradient.

The number of hidden layers and nodes in the hidden layers was fine tuned using the hyperparameter tuner *RandomizedSearchCV*. Two search with *RandomizedSearchCV* was conducted. The first search was done with the activation function in the hidden layers set to *ReLU*. In the second search the activation function was set to *SELU* and the kernel initializer was set to *LeCun*.
Both search was set with 25 interations and 60 epochs. The *RandomizedSearchCV* was able to select $[1, 2, 3, 7, 10]$ hidden layers and $[100, 200, 300, 400, 600]$ neurons in each hidden layer. This was to find the most optimal number of hidden layers and nodes within the hidden layers from the two given lists for both of the activation functions *ReLU* and *SELU* with its *LeCun* kernel initializer.

To avoid overfitting a regularization method called *Dropout* was used.

Both *AlphaDropout* and *Dropout* was used in turns when the model used *ReLU* to explore which would give the highest accuracy score. When the model used *SELU* only the *AlphaDropout* can be used.

The model was trained with *ModelCheckpoint*, *EarlyStopping*, *TensorBoard* and *ReduceLROnPlateau* callbacks. *ReduceLROnPlateau* is a performance scheduler which is a learning scheduler that vary the learning rate during training to find the best learning rate value.

The model which got the highest accuracy was a model using 300 neurons in each of the two hidden layers and *AlphaDropout*. The *ReLU* activation function was used. All of the four callbacks was used. The accuracy was 0.9913750290870667.

4 Repository

A implemention of the multilayer perceptron classifier is found within the Jupyter notebook which is available on GitHub.[1]

[1] `https://github.com/TobiasRingholm/EMNIST-Digits-Image-Classifier`

References

[1] *EMNIST (Extended MNIST)*. `https://www.kaggle.com/crawford/emnist/`. Accessed: March 28, 2022.