

INT 248

Advanced Machine Learning



L OVELY
P ROFESSIONAL
U NIVERSITY

CA-1

BISHNU PRASAD NAYAK

11803586

Roll No: – RKMO72A15

Introduction

This project aims to use Deep Learning model to identify handwritten characters using CNN. The project mainly revolves around analysing images of handwritten characters to predict a newly scribbled character. To make things more interactive we have used tkinter (Python library for GUI). To enable deep learning we have used tensorflow. Apart from this the project uses the following basic requirements for handling data and reshaping them:-

- Python 3.7
- Tkinter
- Tensorflow
- Keras
- Scikit-learn
- Pillow 7.1

Handwritten text is a very general term, and we wanted to narrow down the scope of the project by specifying the meaning of handwritten text for our purposes. In this project, we took on the challenge of classifying the image of any handwritten word, which might be of the form of cursive or block writing.

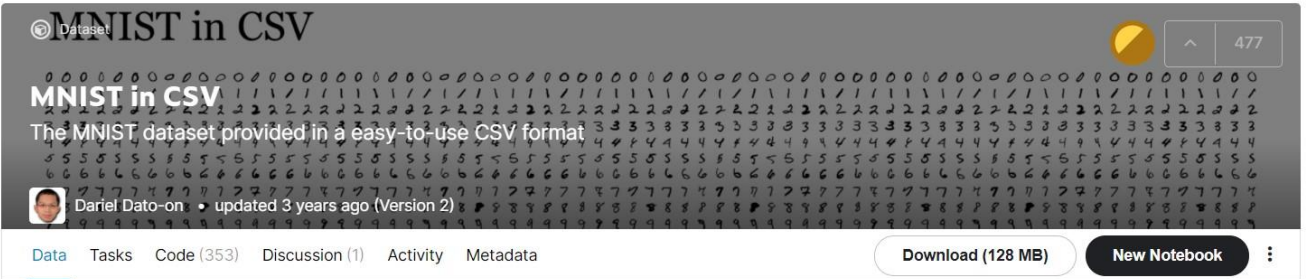
The project is available on Github at <https://github.com/code-evince/Machine-Learning-CA-1>

Dataset Used

The dataset use for training and testing the model are the following:-

1. MNIST data for numbers – Training
2. NIST data for numbers – Testing
3. A-Z alphabet dataset

Basic CNN model trained using MNIST and NIST dataset to predict handwritten characters (letters and digits), each image is resized to grayscale 28x28px image. The .csv format dataset is retrieved from Kaggle.com.



MNIST in CSV

The MNIST dataset provided in a easy-to-use CSV format

Daniel Dato-on • updated 3 years ago (Version 2)

Data Tasks Code (353) Discussion (1) Activity Metadata

Download (128 MB) New Notebook

Usability 8.2 License CC0: Public Domain Tags computer science, image data, beginner

Description

The MNIST dataset provided in a easy-to-use CSV format

The [original dataset](#) is in a format that is difficult for beginners to use. This dataset uses the work of [Joseph Redmon](#) to provide the [MNIST dataset in a CSV format](#).

The dataset consists of two files:

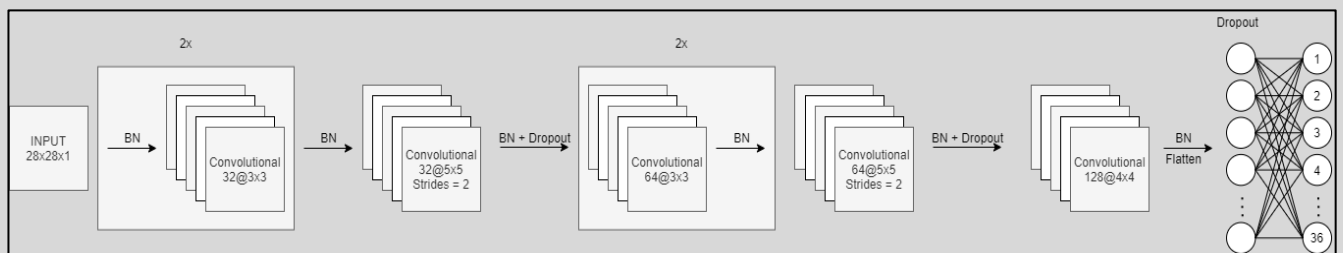
1. `mnist_train.csv`
2. `mnist_test.csv`

The `mnist_train.csv` file contains the 60,000 training examples and labels. The `mnist_test.csv` contains 10,000 test examples and labels. Each row consists of 785 values: the first value is the label (a number from 0 to 9) and the remaining 784 values are the pixel values (a number from 0 to 255).

Proposed Architecture

CNN model is trained using the ADAM(adaptive moment optimization) gradient descent algorithm, logarithmic loss, and a mini-batch gradient descent with mini-batch size 64 then saved model will be used to predict canvas image in Tkinter GUI.

Adam Optimizer is formed by the combination of two Optimizers in Deep Learning, which are Momentum Optimizer and RMSprop Optimizer.



```
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

history = model.fit_generator(generator.flow(x_train, y_train, batch_size=64), epochs=40,
model.save('model_v2.h5')
```

Results

Because of the characteristics of high redundancy, high parallelism and nonlinearity in the handwritten character recognition model, the convolutional neural networks (CNNs) are becoming the first choice to solve these complex problems. The complexity, the types of characters, the character similarity of the handwritten character dataset, and the choice of optimizers all have a great impact on the network model, resulting in low accuracy, high loss, and other problems. After many experiments, the Adam optimization algorithm is finally chosen to optimize the network model.

Finally, the high accuracy recognition of handwritten characters is achieved. The experimental results show that the recognition accuracy of the handwritten characters reached at 88% in the test set, and the loss is low.

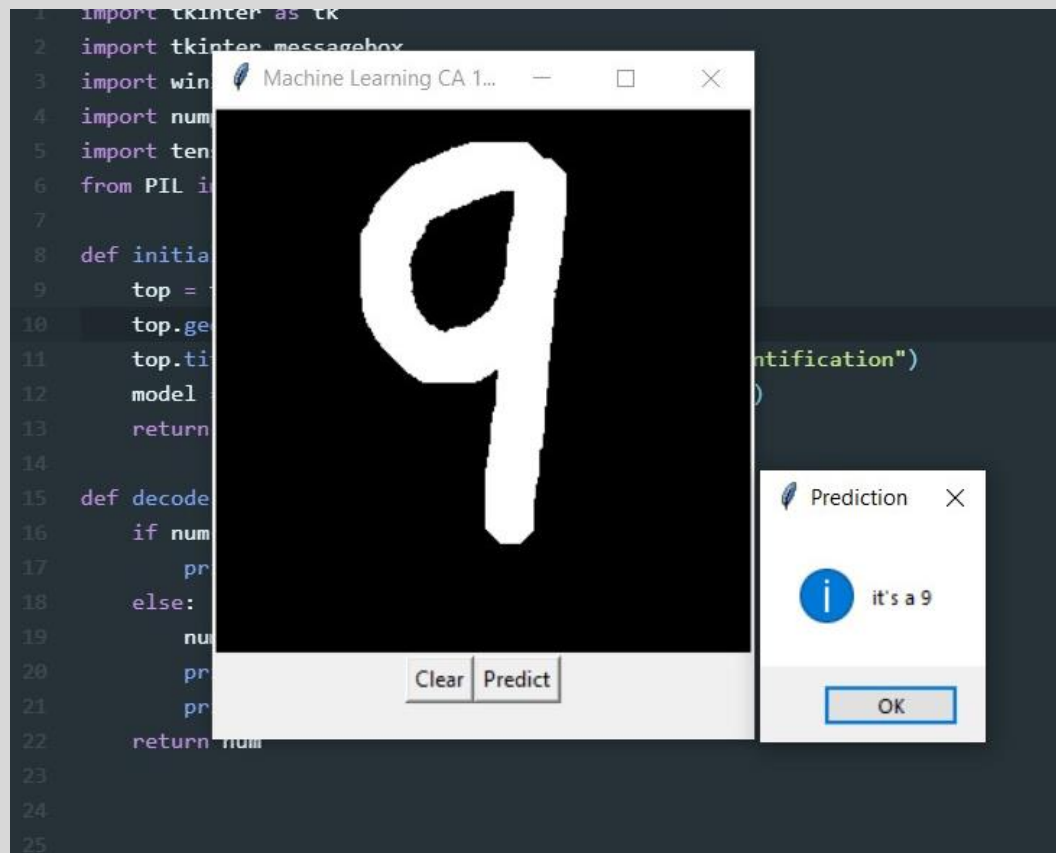
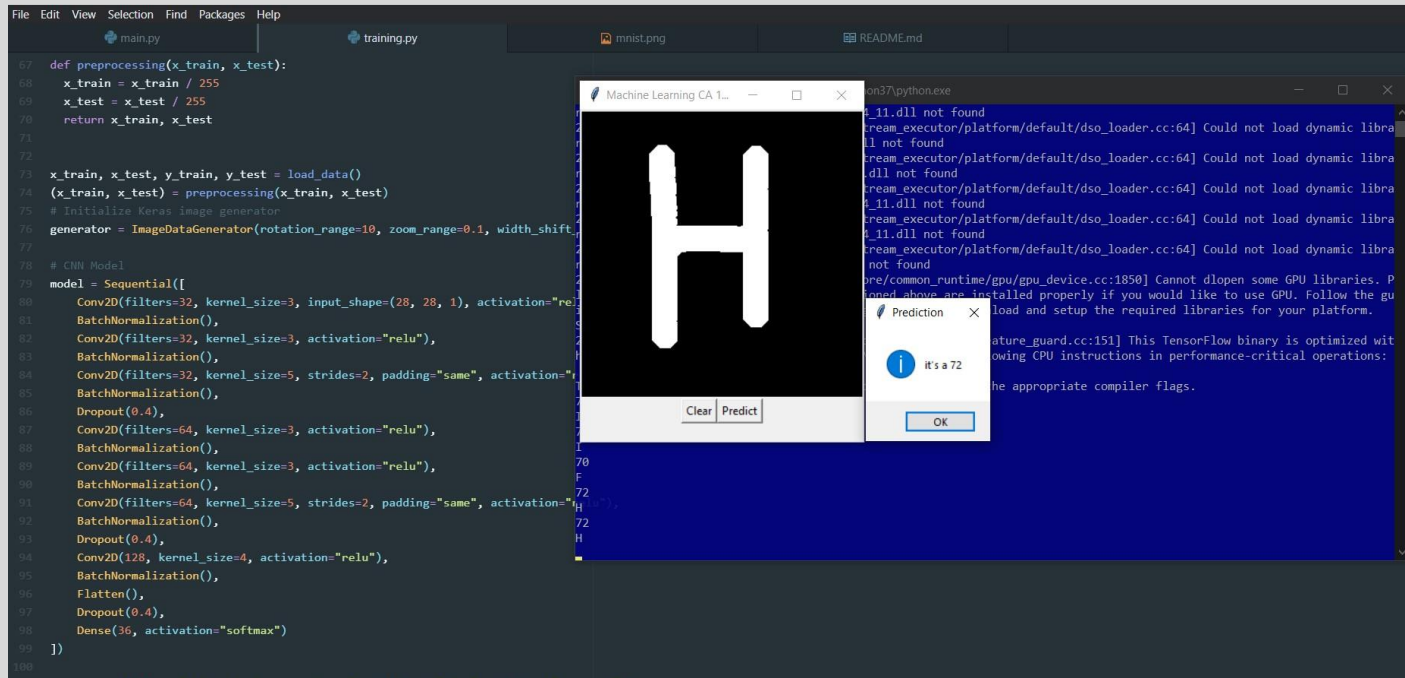
```
Epoch 36/40
506/506 [=====] - 78s 154ms/step - loss: 0.1371 - accuracy: 0.9520 - val_loss: 0.1532 - val_accuracy: 0.9414
Epoch 37/40
506/506 [=====] - 77s 153ms/step - loss: 0.1258 - accuracy: 0.9534 - val_loss: 0.1720 - val_accuracy: 0.9372
Epoch 38/40
506/506 [=====] - 77s 151ms/step - loss: 0.1347 - accuracy: 0.9520 - val_loss: 0.1401 - val_accuracy: 0.9458
Epoch 39/40
506/506 [=====] - 76s 151ms/step - loss: 0.1274 - accuracy: 0.9533 - val_loss: 0.1302 - val_accuracy: 0.9503
Epoch 40/40
506/506 [=====] - 79s 157ms/step - loss: 0.1279 - accuracy: 0.9545 - val_loss: 0.1362 - val_accuracy: 0.9517

Process returned 0 (0x0)      execution time : 3126.279 s
Press any key to continue . . .
```



Report

Screenshot



Future Scope

- Electronic form filling
- Writing electronic applications in one's own handwriting and nativescript
- Alternative to hardware and software keyboards
- Putting in the mathematical equations by simple handwriting

Conclusion

Handwritten character recognition is done in this project. The result which was got was correct up to more than 90% of the cases, but it would be improved at the end. The method I came up with gave efficient and effective result both for feature extraction as well as recognition. There are also different methods through which 'handwritten character recognition' is achieved.

Adam optimization algorithm is finally chosen to optimize the network model. Then, for processing character similarity problems on the pre-processed EMNIST dataset, the dataset is divided into different parts and to be processed. A better-divided result is selected after the comparative experiments. Finally, the high accuracy recognition of handwritten characters is achieved. The experimental results show that the recognition accuracy of the handwritten characters reached at 88% in the test set, and the loss is low.

References

1. Dataset :

https://www.kaggle.com/oddrationalle/mnist-in-csv/version/2?select=mnist_train.csv

2. Tensorflow :

https://www.tensorflow.org/guide/keras/save_and_serialize

3. ADAM Gradient Decent Algorithm

<https://ruder.io/optimizing-gradient-descent/>

4. Keras Documentation :

<https://keras.io/>

5. Convolutional Neural Network Benchmarks:

<https://github.com/jcjohnson/cnn-benchmarks>