Hand Written Digit Recognition using Keras

Sumit Behal
Machine Learning Engineer Intern
AI Tech-Systems
https://ai-techsystems.com/
New Delhi, India

***Abstract*—**Technology is making our life simple and Computer Vision is literally revolutionary. It involves using the dataset of certain images or videos and using them to train the Neural Networks. Neural Networks (NN) are generally used in solving Computer Vision problems. In this report, a NN has been used to identify the digits.

INTRODUCTION

MNIST ("Modified National Institute of Standards and Technology") is the de facto “hello world” dataset of computer vision. It was released in 1989 for research purpose. As new machine learning techniques emerge, MNIST remains a reliable resource for researchers and learners alike.

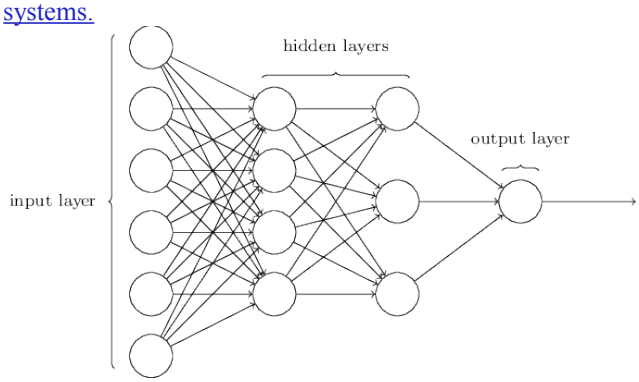
In this project, my goal is to correctly identify digits from a dataset of tens of thousands of handwritten images. I’ll be using those images to train the model.

TOOLS USED

- Python
- Keras
- Tensorflow
- Numpy
- Pandas

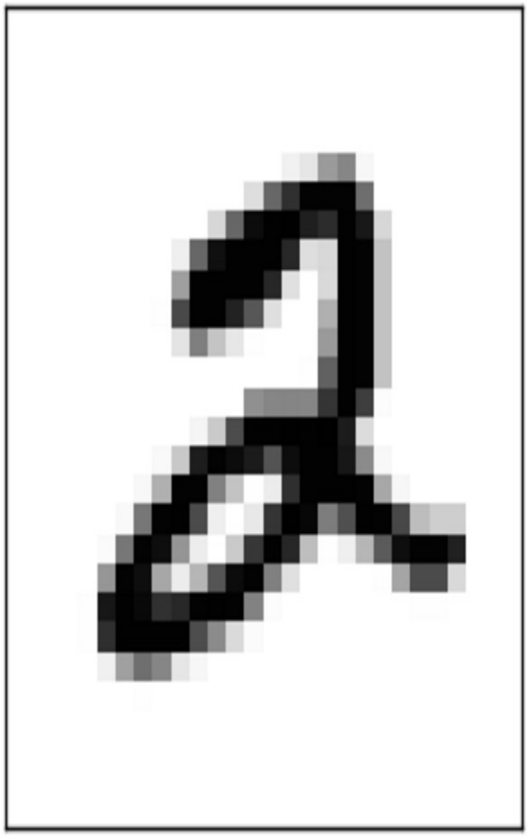
NEURAL NETWORKS

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.



DATASET

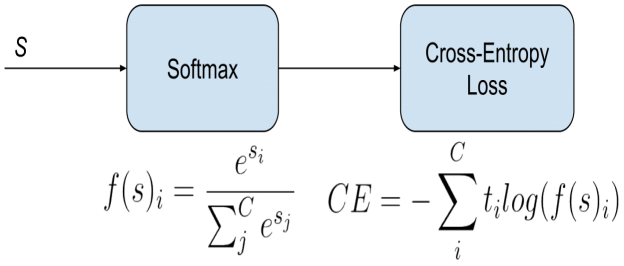
The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size- normalized and centered in a fixed-size image.
It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on pre processing and formatting.



DATA PREPROCESSING

I have reshape the image data into array and have divided the original pixel values by 255 to get the pixel values in between 0 to 1 range. After this, it is converted to categorical form by using keras.utils.

EQUATION



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

TRAINING THE MODEL

I have trained the model using 4 different optimisers. Rest parameters looks like this:

Matrix: Accuracy

Loss: Categorical Cross Entropy

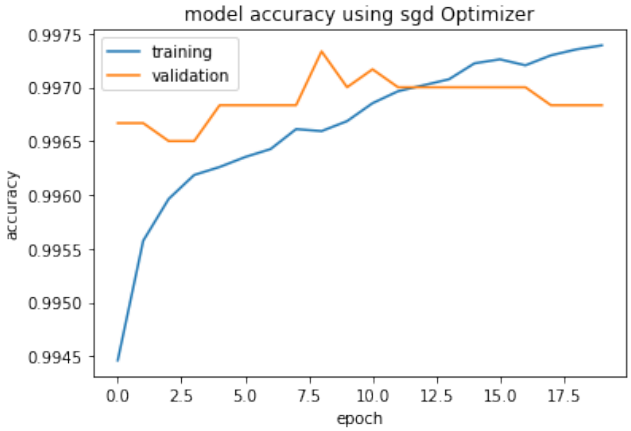
```
W0819 16:20:14.903898 140204807530368 deprecation.py:323] From /usr/local/lib/python3.6/dist-pack
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
W0819 16:20:14.960144 140204807530368 deprecation_wrapper.py:119] From /usr/local/lib/python3.6/d

Epoch 1/5
60000/60000 [==============================] - 6s 102us/step - loss: 0.2671 - acc: 0.9247
Epoch 2/5
60000/60000 [==============================] - 5s 84us/step - loss: 0.1095 - acc: 0.9679
Epoch 3/5
60000/60000 [==============================] - 5s 84us/step - loss: 0.0700 - acc: 0.9794
Epoch 4/5
60000/60000 [==============================] - 5s 88us/step - loss: 0.0506 - acc: 0.9852
Epoch 5/5
60000/60000 [==============================] - 6s 92us/step - loss: 0.0360 - acc: 0.9896
<keras.callbacks.History at 0x7f83bd0285f8>
```

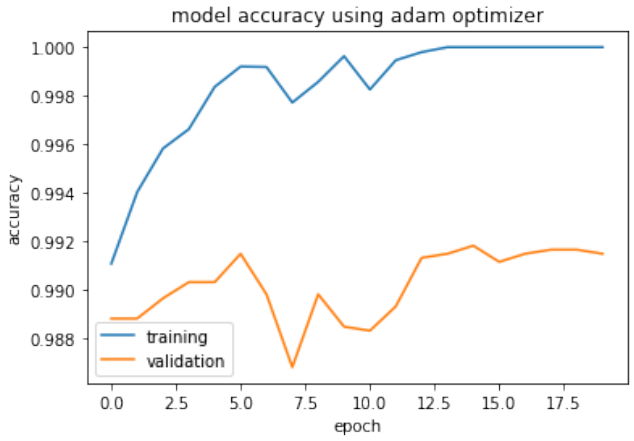
RESULT

Optimizers used:

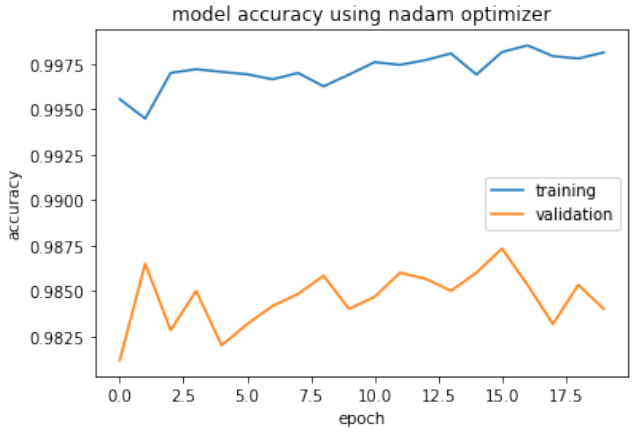
- **SGD**
- **ADAM**
- **NADM**
- **ADADELTA**



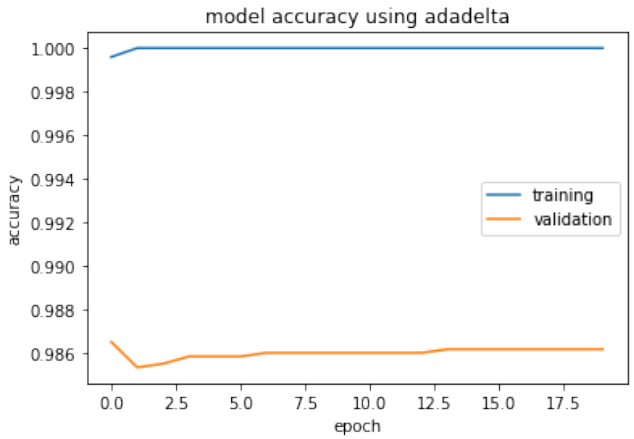

```
Test loss: 0.0552
Test accuracy: 0.982
```




```
Test loss: 0.0714
Test accuracy: 0.984
```




```
Test loss: 0.13
Test accuracy: 0.98
```




```
Test loss: 0.105
Test accuracy: 0.983
```

CONCLUSION

SGD and ADAM optimizers are giving good results in comparison to other optimizer.

.Neural Networks has a bright future and will surely be a revolutionary game changer in solving Computer Vision problems.

Reference:

1.CNN-Based Recognition of Handwritten Digits in MNIST Database Huimin Wu Research School of Computer Science, The Australia National University, Canberra {Huimin Wu}@u6342493@anu.edu.au

2.http://neuralnetworksanddeeplearning.com/chap1.html