Master International E3A

PRACTICAL REPORT

FOR

DEEP LEARNING PRACTICAL

Classification of Handwritten Digits by a Convolutional Neural Network (CNN)

BY:

ABODE DANIEL

SUBMITTED TO:

Prof. Mounim A. EL Yacoubi

DATE: 05/06/2020

Abstract

This practical is about classification of handwritten digits by a convolutional neural network using the MNIST data set of handwritten numbers from 0 to 9. The aim is to understand how to build a convolutional neural network, to train it and to test its performance on some test data set. Further analysis were conducted by varying some of the hyper parameters of the model and building a new architecture in order to appreciate the effects of such hyper parameters on the performance of the model.

Table of Contents

Classification of Handwritten Digits by a Convolutional Neural Network

1.1 Introduction

Deep learning is a technique of machine learning that uses multiple layers to extract high-level features from the input data. In this laboratory exercise, we made use of convolutional neural network; a deep learning algorithm for the classification of Handwritten Digits.

The laboratory exercise consist of 6 parts;

1 Data reading and splitting.

2. Data visualisation.

3. Define the model architecture

4. Model fitting

5. Model evaluation

6. New model architecture & evaluation

1.2 Experimental Setup

The experimental setup involve the use of python programming language development environment, for this practical, the Jupyter development environment was utilized. Some required API were also installed including;

1. Keras
2. Tensorflow

The Data used us a dataset of handwritten numerals made up of 60000 for training and 10000 for test. Each image has a size of 28x28 pixels, the gray level of each being between 0 and 255.

1.3 Objective

The practical objectives is as follow;

1. To be introduced to deep learning techniques with python using Keras API

2. To design an architecture of Convolutional Neural Network model
3. To vary the hyper parameters and study the effect on the performance of the model

1.4 Practical Procedure

1. The required libraries were loaded including Keras and termcolor
2. The dataset was read and spitted into training set and test set, the number of training and test sets was printed.
3. The data was visualized using the matplotlib.pyplot library which was first imported, the first 200 data was shown as in figure 1.
4. The model architecture was created as shown in figure 2.
5. The model was trained with the training data and validation using batch size of 100, epochs of 10, verbose of 1, validation split of 0.3, the result of the training process is as shown in figure 3.
6. The model was evaluated with the test data and the result for the test loss and the test accuracy was obtained.
7. The model result was analyzed and the confusion matrix was plotted as shown in figure 4.
8. One of the hyper parameters (validation split) was changed to see its effect on the performance batch_size = 100, epochs = 10, validation split = 0.25, the test loss and test accuracy were compared.
9. One of the hyparameters (epoch) was changed to see its effect on the performance batch_size = 100, epochs = 15, validation split = 0.25, the test loss and test accuracy were compared.
10. One of the hyparameters (batch size) was changed to see its effect on the performance batch_size = 200, epochs = 15, validation split = 0.25, the test loss and test accuracy were compared.
11. Finally, a new model with the architecture below was implemented, the plot was fitted, and the performance on some test data was analyzed. For the training, we followed up from our result from varying batch size = 100, epochs = 10, and validation split = 0.25, which yielded better result.

```
Convolutional layer with 30 feature maps of size 5×5.
Pooling layer taking the max over 2*2 patches.
Convolutional layer with 15 feature maps of size 3×3.
Pooling layer taking the max over 2*2 patches.
Dropout layer with a probability of 20%.
Flatten layer.
Fully connected layer with 128 neurons and rectifier activation.
Fully connected layer with 50 neurons and rectifier activation.
Output layer.
```

1.5 Result and Discussion

1. After the dataset was loaded, it was spitted into training and test sets as shown below

```
60000 train samples
10000 test samples
```

2. A sample of the data set and produced by step 3 of the practical procedure is shown below in figure 1.
3. Figure 2 shows the architecture of the convolutional neural network designed
4. It can be seen from figure 3 that when the training and validation was performed, the loss for training decreased from 0.3024 to 0.0285 after 10 epochs and the loss the accuracy increased from 0.9076 to 0.9909. In the same way, the loss for validation

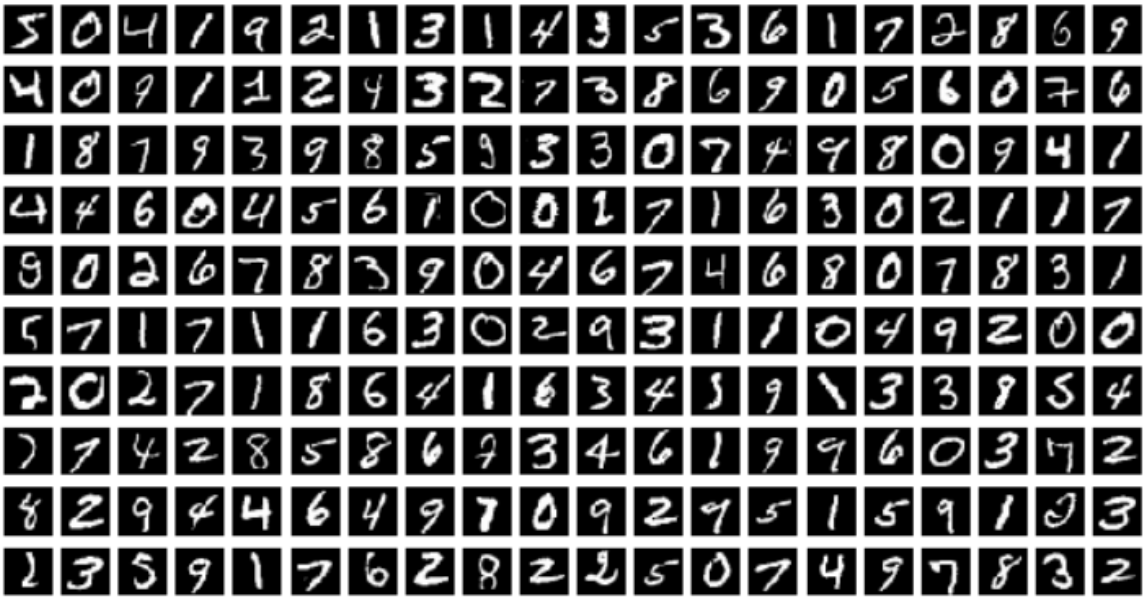

Figure 1: Sample of data set

Model: "sequential_6"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 26, 26, 32)	320
conv2d_4 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 64)	0
dropout_3 (Dropout)	(None, 12, 12, 64)	0
flatten_2 (Flatten)	(None, 9216)	0
dense_3 (Dense)	(None, 128)	1179776
dropout_4 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 10)	1290

Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0

Figure 2: The model's architecture

decreased from 0.0827 to 0.0438 and the accuracy increased from 0.9750 to 0.9889

over 10 epochs.

5. After the evaluation of the model with test datasets, the test loss and test accuracy is as shown below;

```
Test loss: 0.029528830357495282
Test accuracy: 0.9914000034332275
```

6. The confusion matrix is as shown in the figure 4. The y_test_labels is on the y axis and the prediction is on the x axis, the number of test samples that was predicted right is shown in the diagonal for different maps of the input, which are handwritten numerals from 0 to 9.

```
Train on 42000 samples, validate on 18000 samples
Epoch 1/10
42000/42000 [==============================] - 46s 1ms/step - loss: 0.3024 - accuracy: 0.9076 - val_l
oss: 0.0827 - val_accuracy: 0.9750
Epoch 2/10
42000/42000 [==============================] - 47s 1ms/step - loss: 0.0979 - accuracy: 0.9707 - val_l
oss: 0.0565 - val_accuracy: 0.9829
Epoch 3/10
42000/42000 [==============================] - 45s 1ms/step - loss: 0.0718 - accuracy: 0.9778 - val_l
oss: 0.0512 - val_accuracy: 0.9844
Epoch 4/10
42000/42000 [==============================] - 46s 1ms/step - loss: 0.0589 - accuracy: 0.9824 - val_l
oss: 0.0469 - val_accuracy: 0.9869
Epoch 5/10
42000/42000 [==============================] - 46s 1ms/step - loss: 0.0520 - accuracy: 0.9837 - val_l
oss: 0.0429 - val_accuracy: 0.9870
Epoch 6/10
42000/42000 [==============================] - 46s 1ms/step - loss: 0.0436 - accuracy: 0.9858 - val_l
oss: 0.0457 - val_accuracy: 0.9877
Epoch 7/10
42000/42000 [==============================] - 46s 1ms/step - loss: 0.0396 - accuracy: 0.9878 - val_l
oss: 0.0446 - val_accuracy: 0.9878
Epoch 8/10
42000/42000 [==============================] - 46s 1ms/step - loss: 0.0370 - accuracy: 0.9892 - val_l
oss: 0.0474 - val_accuracy: 0.9881
Epoch 9/10
42000/42000 [==============================] - 47s 1ms/step - loss: 0.0326 - accuracy: 0.9901 - val_l
oss: 0.0425 - val_accuracy: 0.9894
Epoch 10/10
42000/42000 [==============================] - 46s 1ms/step - loss: 0.0285 - accuracy: 0.9909 - val_l
oss: 0.0438 - val_accuracy: 0.9889
```

Figure 3: Performance on training and validation over 10 epochs

```
Confusion Matrix
 [[ 977    0    0    0    0    1    0    1    1    0]
 [   0 1132    2    0    0    0    1    0    0    0]
 [   1    0 1027    0    1    0    0    3    0    0]
 [   0    0    2 1004    0    2    0    1    1    0]
 [   0    0    0    0  977    0    0    0    2    3]
 [   2    0    1    6    0  881    2    0    0    0]
 [   5    2    0    1    5    1  943    0    1    0]
 [   1    1    4    2    0    0    0 1019    1    0]
 [   3    1    1    1    0    0    1    2  962    3]
 [   3    1    0    1    4    3    0    3    2  992]]
```

Figure 4: Confusion Matrix

7. After we changed the validation split from 0.3 to 0.25 the following result was obtained

```
Epoch 10/10
45000/45000 [==============================] - 49s 1ms/step - loss: 0.0286 -
accuracy: 0.9910 - val_loss: 0.0426 - val_accuracy: 0.9893
```

```
Test loss: 0.028770944652544175
Test accuracy: 0.9916999936103821
```

```
Confusion Matrix
 [[ 978    0    0    0    0    0    0    1    1    0]
 [   0 1132    1    0    0    1    1    0    0    0]
 [   2    1 1021    0    1    0    0    6    1    0]
 [   0    0    3 1004    0    2    0    0    1    0]
 [   1    0    0    0  976    0    0    0    1    4]
 [   2    0    0    4    0  883    3    0    0    0]
 [   5    2    0    0    1    4  944    0    2    0]
 [   0    1    5    2    0    0    0 1019    1    0]
 [   3    0    1    0    0    0    0    2  966    2]
 [   2    2    0    0    3    3    0    3    2  994]]
```

Figure 5: Performance after validation split was reduced from 0.3 to 0.25

Observation: It was observed that there was an improvement in the performance when the validation split was reduced to 0.25, the training accuracy, validation accuracy and test accuracy all increased compared to when validation split was 0.30.

8. After we changed the number of epochs from 10 to 15 the following result was obtained

```
Epoch 15/15
45000/45000 [==============================] - 46s 1ms/step - loss: 0.0232 -
accuracy: 0.9931 - val_loss: 0.0481 - val_accuracy: 0.9894

Test loss: 0.031907799031840114
Test accuracy: 0.9900000095367432

Confusion Matrix
 [[ 977    0    1    0    0    0    1    1    0    0]
 [   0 1133    1    1    0    0    0    0    0    0]
 [   1    2 1027    0    0    0    0    2    0    0]
 [   0    0    3 1004    0    2    0    0    1    0]
 [   0    0    0    0  976    0    1    0    1    4]
 [   2    0    0    6    0  881    3    0    0    0]
 [   4    2    0    1    1    1  948    0    1    0]
 [   1    3    9    1    0    0    0 1013    1    0]
 [   3    1    2    1    0    0    1    2  962    2]
 [   2    1    0    1    2    1    0    6    1  995]]
```

Figure 6: Performance after number of epochs was increased from 10 to 15

Observation: It was observed that there was an improvement training accuracy from 0.9910 to 0.9931 and also an improvement in validation accuracy from 0.9893 to 0.9894, but the performance on the test set was reduced, accuracy in test reduced from 0.9913 to 0.9900. This shows that for this architecture, increasing the number of epochs does not translate to a better performance.

9. After we changed the batch size from 100 to 200 the following result was obtained

```
Epoch 15/15
45000/45000 [==============================] - 46s 1ms/step - loss: 0.0220 -
accuracy: 0.9928 - val_loss: 0.0420 - val_accuracy: 0.9885

Test loss: 0.027180946334531 02
Test accuracy: 0.991599977016449

Confusion Matrix
 [[ 975    0    2    0    0    1    1    1    0    0]
 [   0 1128    2    0    0    1    1    2    1    0]
 [   1    0 1028    0    0    0    0    3    0    0]
 [   0    0    2 1003    0    2    0    2    1    0]
 [   0    0    0    0  973    0    2    0    1    6]
 [   1    0    0    3    0  887    1    0    0    0]
 [   5    2    0    1    1    3  944    0    2    0]
 [   0    0    5    1    0    0    0 1021    1    0]
 [   3    0    3    0    0    1    0    1  965    1]
 [   3    2    0    0    3    3    0    3    3  992]]
```

Figure 7: Performance after batch size was increased from 100 to 200

Observation: It was observed that there was a reduction of training accuracy from 0.9931 to 0.9928 after 15 epochs and also a reduction in validation accuracy from 0.9894 to 0.9885, but the performance on the test set was improved, accuracy in test increased from 0.9900 to 0.9915. This shows that for this architecture, increasing the number of epochs does not translate to a better performance.

From our previous observation, the best performance on the test was observed when validation split was reduced to 0.25, while batch size and epochs remained as 100 and 10 respectively.

10. The new model created and its performance is shown in below

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 24, 24, 30)        780
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 30)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 10, 10, 15)        4065
_________________________________________________________________
dropout_4 (Dropout)          (None, 10, 10, 15)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1500)              0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               192128
_________________________________________________________________
dense_7 (Dense)              (None, 50)                6450
_________________________________________________________________
dense_8 (Dense)              (None, 10)                510
=================================================================
Total params: 203,933
Trainable params: 203,933
Non-trainable params: 0
```

```
Epoch 10/10
45000/45000 [==============================] - 11s 250us/step - loss: 0.0138
- accuracy: 0.9958 - val_loss: 0.0387 - val_accuracy: 0.9898

New Model Error: 0.81%
Test loss: 0.024273677057947498
Test accuracy: 0.9919000267982483
```

```
Confusion Matrix
 [[ 977    0    0    0    1    0    1    1    0    0]
 [   0 1131    2    0    0    0    1    0    1    0]
 [   1    0 1026    0    1    0    0    3    1    0]
 [   0    0    2  998    0    3    0    3    1    3]
 [   0    0    1    0  976    0    2    0    1    2]
 [   2    0    0    8    0  875    2    2    1    2]
 [   5    2    0    0    2    1  944    0    4    0]
 [   0    1    2    1    0    0    0 1023    0    1]
 [   3    0    4    1    0    1    0    1  960    4]
 [   0    2    0    0    5    3    0    8    1  990]]
```

Figure 8: Performance of new model

Observation: the performance of this architecture with training hyper parameters; batch size set to 100, epoch to 10 and validation split 0.25, gave the best performance so far on the test data with a test accuracy of 0.9919 compared to the best performance in the previous model that gave a test accuracy of 0.9916. The improvement is minute

1.6 Conclusion

In this practical, we successfully learn how to build a convolutional neural network architecture for classification of handwritten numeric digits. We were able to evaluate the performance of the model using the result of the test accuracy, loss and the confusion matrix. By varying some of the hyper-parameters, we were able to evaluate their effects on the performance of the model. When we reduced the validation split to 0.25, meaning allowing more of the training data for the training procedure, we saw an improvement on the accuracy of the model on the test data. However, increasing the batch size and number of epochs led to reduction in the accuracy of the model on test data, but an increase on accuracy of model on validation data, this could mean an effect of overfitting. We created a new model that produce a better performance than the initial model. I learnt that quite some tuning process of the hyper-parameters and architecture of the model could either improve performance or reduce the performance of the model.