Deep Learning for Advanced Robot Perception

Assignment 4: Defeating .99% Deep Learning error in MNIST final evaluation

Classify Handwritten Digits

AKSHAY KUMAR | WPI ID - 842954269

Downloading the MNIST dataset available from *keras.datasets* and develop a multilayer perceptron as well as another convolutional neural network to classify the data. The dataset is a collection of handwritten digits.

Convolutional Neural Network

Since the MNIST dataset is not a very huge image dataset because each image is only 28 x 28 in resolution, a simplete CNN with one convolutional layer and one max pooling layer followed by hidden layers after flattening are enough to generate very acceptable results.

The target was to obtain a model with prediction error of less than 1% and that has been achieved in the implemented architecture.

The model summary is shown as below:

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 24, 24)	832
max_pooling2d_1 (MaxPooling2	(None, 32, 12, 12)	0
dropout_1 (Dropout)	(None, 32, 12, 12)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_1 (Dense)	(None, 128)	589952
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 10)	1290

Total params: 592,074
Trainable params: 592,074
Non-trainable params: 0

The model was able to perform satisfactorily with the following scores.

```
Epoch 9/10
60000/60000 [==============================] - 127s 2ms/step - loss: 0.0243 - acc: 0.9933 - val_loss: 0.0317 - val_acc: 0.9904
Epoch 10/10
60000/60000 [==============================] - 131s 2ms/step - loss: 0.0201 - acc: 0.9946 - val_loss: 0.0311 - val_acc: 0.9911
```

CNN Error: 0.96%

The corresponding code script has been attached as ***cnn_mnist_data.py*** with all the tuned parameters.

Multi-Layer Perceptron

While the idea of convolutional neural networks is limited only to the images, the general neural networks with multiple hidden layers can work for any given input. However, MLP for images classification can be used only on images with low resolution like the MNIST handwritten digits dataset with only 28 x 28 resolution.

The model summary is shown as below:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 784)               615440
_________________________________________________________________
dense_2 (Dense)              (None, 256)               200960
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                2570
=================================================================
Total params: 818,970
Trainable params: 818,970
Non-trainable params: 0
```

The model was able to perform satisfactorily with the following scores that comply with the desired performance.

```
Epoch 19/20
60000/60000 [==============================] - 5s 84us/step - loss: 0.0245 - acc: 0.9938 - val_loss: 0.0605 - val_acc: 0.9907
Epoch 20/20
60000/60000 [==============================] - 5s 83us/step - loss: 0.0217 - acc: 0.9945 - val_loss: 0.0515 - val_acc: 0.9909
MLP Error: 0.94%
```

The corresponding code script has been attached as ***mlp_mnist_data.py*** with all the tuned parameters.