

# NEURAL NETWORK & DEEP LEARNING(CS-5720)

## (CRN:31196)

### ASSIGNMENT - 3

**Name** : Vamsi Krishna Mekala

**ID** : 700742751

**Date** : 07/25/2023

**Github** : <https://github.com/vamsi-mekala/Neural-networks-assignment-3>

**Google Drive**:[https://drive.google.com/file/d/1HFU5CSvk9eC47XUTuDvMj-NBCzCRhE\\_M/view?usp=sharing](https://drive.google.com/file/d/1HFU5CSvk9eC47XUTuDvMj-NBCzCRhE_M/view?usp=sharing)

#### Question 1:

Follow the instruction below and then report how the performance changed.(apply all at once)

- Convolutional input layer, 32 feature maps with a size of  $3 \times 3$  and a rectifier activation function.
- Dropout layer at 20%.
- Convolutional layer, 32 feature maps with a size of  $3 \times 3$  and a rectifier activation function.
- Max Pool layer with size  $2 \times 2$ .
- Convolutional layer, 64 feature maps with a size of  $3 \times 3$  and a rectifier activation function.
- Dropout layer at 20%.
- Convolutional layer, 64 feature maps with a size of  $3 \times 3$  and a rectifier activation function.
- Max Pool layer with size  $2 \times 2$ .
- Convolutional layer, 128 feature maps with a size of  $3 \times 3$  and a rectifier activation function.
- Dropout layer at 20%.
- Convolutional layer, 128 feature maps with a size of  $3 \times 3$  and a rectifier activation function.
- Max Pool layer with size  $2 \times 2$ .
- Flatten layer.
- Dropout layer at 20%.
- Fully connected layer with 1024 units and a rectifier activation function.
- Dropout layer at 20%.

- Fully connected layer with 512 units and a rectifier activation function.
- Dropout layer at 20%.
- Fully connected output layer with 10 units and a Softmax activation function

The above specification are coded in as the model shown below:

```
In [6]: model = Sequential()
model.add(Conv2D(32,(3,3),activation='relu',input_shape=input_shape,padding='same'))
model.add(Conv2D(32,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Flatten())
model.add(Dense(1024,activation='relu'))
model.add(Dense(512,activation='relu'))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

Then the model is fit to the training data and then the model is evaluated and tested against some test data:

```
In [9]: history = model.fit(x=x_train,y=one_hot_y_train, batch_size=512,epochs=5,verbose=1)

WARNING:tensorflow:From c:\users\hp\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\ops\math_grad.py:1250: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From c:\users\hp\appdata\local\programs\python\python37\lib\site-packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.

Epoch 1/5
50000/50000 [=====] - 168s 3ms/step - loss: 14.4306 - acc: 0.0999
Epoch 2/5
50000/50000 [=====] - 165s 3ms/step - loss: 14.5063 - acc: 0.1000
Epoch 3/5
50000/50000 [=====] - 175s 3ms/step - loss: 14.5063 - acc: 0.1000
Epoch 4/5
50000/50000 [=====] - 176s 4ms/step - loss: 14.5063 - acc: 0.1000
Epoch 5/5
50000/50000 [=====] - 177s 4ms/step - loss: 14.5063 - acc: 0.1000
```

```
In [19]: import matplotlib.pyplot as plt

# Plot the results
fig, axes = plt.subplots(1, 2, figsize=(15, 5))

# Plot training and validation accuracy
axes[1].plot(history.history['acc'])
axes[1].set_title('Model accuracy')
axes[1].set_ylabel('Accuracy')
axes[1].set_xlabel('Epoch')
axes[1].legend(['Train', 'Test'], loc='lower right')

# Plot training and validation loss
axes[0].plot(history.history['loss'])
axes[0].set_title('Model loss')
axes[0].set_ylabel('Loss')
axes[0].set_xlabel('Epoch')
axes[0].legend(['Train', 'Test'], loc='upper right')

plt.show()
```



