Report for Deep Learning (CS541) Homework5:

Report By:
Swapneel Dhananjay Wagholikar (WPI ID: 257598983)
Anuj Pradeep Pai Raikar (WPI ID: 181758784)

Section 1: Implementing a Neural Network with TensorFlow

We have implemented the neural network model that was linked in the homework. We have changed the parameters such as number of filters and kernel size in the convolutional filter as compared to the reference code. The screenshot of our modified model is attached below:

```
#Part 1 model

model = tf.keras.Sequential()

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, padding='same', activation='relu', input_shape=(28,28,1)))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))

model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

Results:-

We received test accuracy more than 92.47%. Please check the screenshot below for test accuracy and last 5 epoch results (gradient updates on training set):

```
 Test accuracy: 0.9247000217437744
313/313 [==============================] - 2s 5ms/step
```

```
Epoch 6/10
860/860 [==============================] - ETA: 0s - loss: 0.2244 - accuracy: 0.9165
Epoch 6: val_loss improved from 0.20920 to 0.20597, saving model to model.weights.best.hdf5
860/860 [==============================] - 31s 36ms/step - loss: 0.2244 - accuracy: 0.9165 - val_loss: 0.2060 - val_accuracy:
0.9238
Epoch 7/10
859/860 [============================>.] - ETA: 0s - loss: 0.2123 - accuracy: 0.9208
Epoch 7: val_loss improved from 0.20597 to 0.19809, saving model to model.weights.best.hdf5
860/860 [==============================] - 32s 37ms/step - loss: 0.2123 - accuracy: 0.9208 - val_loss: 0.1981 - val_accuracy:
0.9248
Epoch 8/10
860/860 [==============================] - ETA: 0s - loss: 0.1997 - accuracy: 0.9258
Epoch 8: val_loss improved from 0.19809 to 0.19147, saving model to model.weights.best.hdf5
860/860 [==============================] - 31s 36ms/step - loss: 0.1997 - accuracy: 0.9258 - val_loss: 0.1915 - val_accuracy:
0.9284
Epoch 9/10
859/860 [============================>.] - ETA: 0s - loss: 0.1868 - accuracy: 0.9305
Epoch 9: val_loss did not improve from 0.19147
860/860 [==============================] - 32s 37ms/step - loss: 0.1867 - accuracy: 0.9305 - val_loss: 0.1942 - val_accuracy:
0.9268
Epoch 10/10
859/860 [============================>.] - ETA: 0s - loss: 0.1790 - accuracy: 0.9331
Epoch 10: val_loss improved from 0.19147 to 0.18725, saving model to model.weights.best.hdf5
860/860 [==============================] - 30s 35ms/step - loss: 0.1790 - accuracy: 0.9331 - val_loss: 0.1872 - val_accuracy:
0.9310
```

Section 2: Representing a CNN as a fully connected Neural Network

1. The architecture is implemented in TensorFlow.(Code File attached)
2. Model weights are extracted using the required functions.
3. HW4 submission is expanded as per the requirement.
4. Extracted weights from the new model are applied in our fully connected neural network.
5. The new architecture and feed forward networks are used to predict the softmax distribution on a randomly selected MNIST image and it is observed that the output is same.

Please find the required screenshot below:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 26, 26, 64)        640

 max_pooling2d_2 (MaxPooling  (None, 13, 13, 64)       0
 2D)

 re_lu (ReLU)                (None, 13, 13, 64)        0

 flatten_1 (Flatten)         (None, 10816)             0

 dense_2 (Dense)             (None, 1024)              11076608

 dense_3 (Dense)             (None, 10)                10250

=================================================================
Total params: 11,087,498
Trainable params: 11,087,498
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 1/2
860/860 [==============================] - ETA: 0s - loss: 0.3653 - accuracy: 0.8686
Epoch 1: val_loss improved from inf to 0.26680, saving model to model_new.weights.best.hdf5
860/860 [==============================] - 63s 73ms/step - loss: 0.3653 - accuracy: 0.8686 - val_loss: 0.2668 - val_accuracy:
0.9052
Epoch 2/2
859/860 [============================>.] - ETA: 0s - loss: 0.2409 - accuracy: 0.9113
Epoch 2: val_loss improved from 0.26680 to 0.24375, saving model to model_new.weights.best.hdf5
860/860 [==============================] - 60s 69ms/step - loss: 0.2410 - accuracy: 0.9113 - val_loss: 0.2437 - val_accuracy:
0.9140
1/1 [==============================] - 0s 66ms/step

Test accuracy of the new model: 90.2999997138977 %
```

```
Prediction for new CNN Model

 [4.3455231e-05 1.6024942e-06 2.1142490e-02 1.6646030e-05 9.7089291e-01
 8.6463984e-07 7.8637898e-03 5.0611789e-07 3.3072207e-05 4.6679620e-06]

Prediction for Converted Feed Forward Network

 [4.34552580e-05 1.60249535e-06 2.11424956e-02 1.66460396e-05
 9.70892896e-01 8.64640535e-07 7.86379376e-03 5.06118070e-07
 3.30722345e-05 4.66796166e-06]
```

As can be seen the outputs for the the new CNN model created with tensor flow and the feedforward model are the same.