Classification using DEEP Convolutional Neural Network Using TensorFlow

Explanation:

- Data is divided into two parts. First 200 images from each class are used for training. The next 20 are used as test images.

```
Found 3000 images belonging to 15 classes.
Found 300 images belonging to 15 classes.
```

- Data is generated using Image Data Generator, which generates tf.data.Dataset from image files in a directory. Data is normalized between 0-1. The sized is fixed at 256 × 256 × 3. Training is done in batch size of 50.

```
Batch shape=(50, 256, 256, 3), min=0.000, max=1.000
```

- A CNN is generated using tf.keras. The CNN has 5 convolution layers, which performs 16, 32, 64, 64, 64 convolutions respectively. Activation function being used is “Relu”. The five convolution layers are followed by a DNN which gets convolution layer result after its flattened. Its then fed to a 512 layer hidden layer and finally we get the 15 layer output (since we have 15 classes). Each output represents the probability of an image belonging to that class.
- (All the parameters were picked after hit and trial keeping in mind the accuracy and computational complexity.)

```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 256x256 with 3 bytes color
    # This is the first convolution
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(256, 256, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fifth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    # 15 output neuron since we have 15 categories
    tf.keras.layers.Dense(15, activation='softmax')
])
```

- Model summary is shown in the figure.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 254, 254, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 127, 127, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 125, 125, 32)      4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 62, 62, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 60, 60, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1180160
_________________________________________________________________
dense_1 (Dense)              (None, 15)                7695
=================================================================
Total params: 1,285,295
Trainable params: 1,285,295
Non-trainable params: 0
_________________________________________________________________
```

- Training is done on the 3000 training images. Epochs are set at 100 with callback to stop iterating when accuracy of training data reaches 70% to avoid overfitting.

Results:

At the 80th epoch training set accuracy was achieved to be greater than 0.7 so callback stopped iterating. It had slightly lower accuracy on testing data as expected. This is shown in the following figure.

```
1/4 [======>.......................] - ETA: 5s - loss: 0.9800 - acc: 0.6600
2/4 [==============>...............] - ETA: 3s - loss: 1.0878 - acc: 0.6200
3/4 [======================>........] - ETA: 1s - loss: 1.0006 - acc: 0.6533
4/4 [==============================] - 7s 2s/step - loss: 0.9998 - acc: 0.6450
Epoch 80/100

1/4 [======>.......................] - ETA: 5s - loss: 0.9281 - acc: 0.6800
2/4 [==============>...............] - ETA: 3s - loss: 0.8535 - acc: 0.7100
3/4 [======================>........] - ETA: 1s - loss: 0.8851 - acc: 0.7000
Reached 70% accuracy so cancelling training!

4/4 [==============================] - 7s 2s/step - loss: 0.8658 - acc: 0.7050

Done Training. Evaluating Performance on Testing Data

 1/10 [==>...........................] - ETA: 11s - loss: 1.3075 - acc: 0.6250
 2/10 [=====>........................] - ETA: 8s - loss: 1.1736 - acc: 0.6094
 3/10 [========>.....................] - ETA: 6s - loss: 1.1469 - acc: 0.6146
 4/10 [===========>..................] - ETA: 4s - loss: 1.1275 - acc: 0.6172
 5/10 [==============>...............] - ETA: 3s - loss: 1.0786 - acc: 0.6375
 6/10 [=================>............] - ETA: 2s - loss: 1.1705 - acc: 0.6094
 7/10 [====================>.........] - ETA: 2s - loss: 1.1460 - acc: 0.6027
 8/10 [========================>.....] - ETA: 1s - loss: 1.1046 - acc: 0.6172
 9/10 [===========================>...] - ETA: 0s - loss: 1.0982 - acc: 0.6250
10/10 [==============================] - 6s 585ms/step - loss: 1.2177 - acc: 0.6133
```