Answer [CM2]

(Our Own Network)

We have added few dropout layers and a convolution layers with increasing number of filters. This is to overcome the overfitting happening the previous model after 10 epochs.

In [25]:

```python
# Model
model = Sequential()
# Add convolution 2D
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 kernel_initializer='he_normal',
                 input_shape=(IMG_ROWS, IMG_COLS, 1)))
model.add(MaxPooling2D((2, 2)))
# Add dropouts to the model
model.add(Dropout(0.25))
model.add(Conv2D(64,
                 kernel_size=(3, 3),
                 activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
# Add dropouts to the model
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), activation='relu'))
# Add dropouts to the model
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
# Add dropouts to the model
model.add(Dropout(0.3))
model.add(Dense(NUM_CLASSES, activation='softmax'))


model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer='adam',
              metrics=['accuracy'])
```

In [26]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_2 (Conv2D)            (None, 26, 26, 32)        320       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0         
_________________________________________________________________
dropout (Dropout)            (None, 13, 13, 32)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 11, 11, 64)        18496     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 5, 64)          0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 3, 3, 128)         73856     
_________________________________________________________________
dropout_2 (Dropout)          (None, 3, 3, 128)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 128)               147584    
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 645       
=================================================================
Total params: 240,901
Trainable params: 240,901
Non-trainable params: 0
_________________________________________________________________
```

In [27]:
```
print(len(model.layers))
```

```
12
```

In [28]:
```
train_new_model = model.fit(X_train, y_train,
                            batch_size=64,
                            epochs=25,
                            verbose=1,
                            validation_data=(X_val, y_val))
```

```
750/750 [==============================] - 9s 13ms/step - loss: 0.1710 -
accuracy: 0.9371 - val_loss: 0.1556 - val_accuracy: 0.9441
Epoch 20/25
750/750 [==============================] - 9s 12ms/step - loss: 0.1720 -
accuracy: 0.9361 - val_loss: 0.1546 - val_accuracy: 0.9448
Epoch 21/25
750/750 [==============================] - 9s 12ms/step - loss: 0.1694 -
accuracy: 0.9375 - val_loss: 0.1755 - val_accuracy: 0.9359
Epoch 22/25
750/750 [==============================] - 9s 12ms/step - loss: 0.1689 -
accuracy: 0.9368 - val_loss: 0.1577 - val_accuracy: 0.9428
Epoch 23/25
750/750 [==============================] - 9s 12ms/step - loss: 0.1670 -
accuracy: 0.9387 - val_loss: 0.1562 - val_accuracy: 0.9444
Epoch 24/25
750/750 [==============================] - 9s 12ms/step - loss: 0.1634 -
accuracy: 0.9380 - val_loss: 0.1537 - val_accuracy: 0.9445
Epoch 25/25
750/750 [==============================] - 9s 12ms/step - loss: 0.1645 -
accuracy: 0.9391 - val_loss: 0.1541 - val_accuracy: 0.9442
```

- Maximum training accuracy is 94.13 % at epoch = 25
- Maximum validation accuracy is 94.29% % at epoch = 25

Here we can see that hte model is performing better than the previous model in terms of training loss vs validation loss and dropout layes seems to be helping in stoping overfitting.

In [30]:
```
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.17063970863819122
Test accuracy: 0.9391000270843506
[0.17063970863819122, 0.9391000270843506]
```

Test accuracy of the new model is 93.91% at tes loss of .17 which is half of the previous models.

- So even after increasing conv layers and filters sizes plus number of epochs compare to previous model, the training accuracy vs validation accuracy is more stabel and we are getting less val loss and test loss.

More of this we will discuss in [CM3].

In [ ]:

In [ ]: