Debarghya Dey Enrollment No.- 510519087 Assignment - 6

Project Setup

Mount Google Drive

In [7]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Importing all necessary libraries

In [8]:
```
# system info
import platform
import random
import os
import sys
from collections import OrderedDict

# data visualisation and manipulation
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
import sklearn
from sklearn.metrics import classification_report, precision_score, recall_score, f1_score, precision_recall_fscore_sup

#model selection
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score,precision_score,recall_score,confusion_matrix,roc_curve,roc_auc_score
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import LabelEncoder

#preprocess.
```

```
from keras.preprocessing.image import ImageDataGenerator

#dl libraraies
from keras import backend as K
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam,SGD,Adagrad,Adadelta,RMSprop
from keras.utils import to_categorical

# specifically for cnn
from keras.layers import Dropout, Flatten,Activation
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization, AveragePooling2D
import tensorflow as tf
import random as rn
import keras

# specifically for manipulating zipped images and getting numpy arrays of pixel values of images.
import cv2
import numpy as np
from tqdm import tqdm
import os
from random import shuffle
from zipfile import ZipFile
from PIL import Image
```

Data-Loading

Create training data

In [9]:
```
X = []
Z = []
IMG_SIZE = 80
BASE_PATH = "/content/drive/My Drive/ML_DRIVE/Assign_6/flowers_dataset/"
FLOWER_DAISY_DIR     = BASE_PATH + "daisy"
FLOWER_SUNFLOWER_DIR = BASE_PATH + "sunflower"
FLOWER_TULIP_DIR     = BASE_PATH + "tulip"
FLOWER_DANDI_DIR     = BASE_PATH + "dandelion"
FLOWER_ROSE_DIR      = BASE_PATH + "rose"
```

In [10]:
```
def assign_label(img, flower_type):
    return flower_type

def make_train_data(flower_type, DIR, mode = 'color'):
    for img in tqdm(os.listdir(DIR)):
        label = assign_label(img, flower_type)
        path = os.path.join(DIR, img)
        if (mode == 'color'):
          img = cv2.imread(path, cv2.IMREAD_COLOR)
        else:
          img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)

        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))

        X.append(img)
        Z.append(str(label))
    return
```

In [11]:
```
make_train_data('Daisy',FLOWER_DAISY_DIR, 'gray')
```

```
100%|██████████| 764/764 [00:13<00:00, 54.68it/s]
```

In [12]:
```
make_train_data('Sunflower',FLOWER_SUNFLOWER_DIR, 'gray')
```

```
100%|██████████| 733/733 [00:11<00:00, 63.90it/s]
```

In [13]:
```
make_train_data('Tulip',FLOWER_TULIP_DIR, 'gray')
```

```
100%|██████████| 984/984 [00:22<00:00, 44.56it/s]
```

In [14]:
```
make_train_data('Dandelion',FLOWER_DANDI_DIR, 'gray')
```

```
100%|██████████| 1052/1052 [00:18<00:00, 57.20it/s]
```

In [15]:
```
make_train_data('Rose',FLOWER_ROSE_DIR, 'gray')
```

```
100%|██████████| 784/784 [00:11<00:00, 70.37it/s]
```

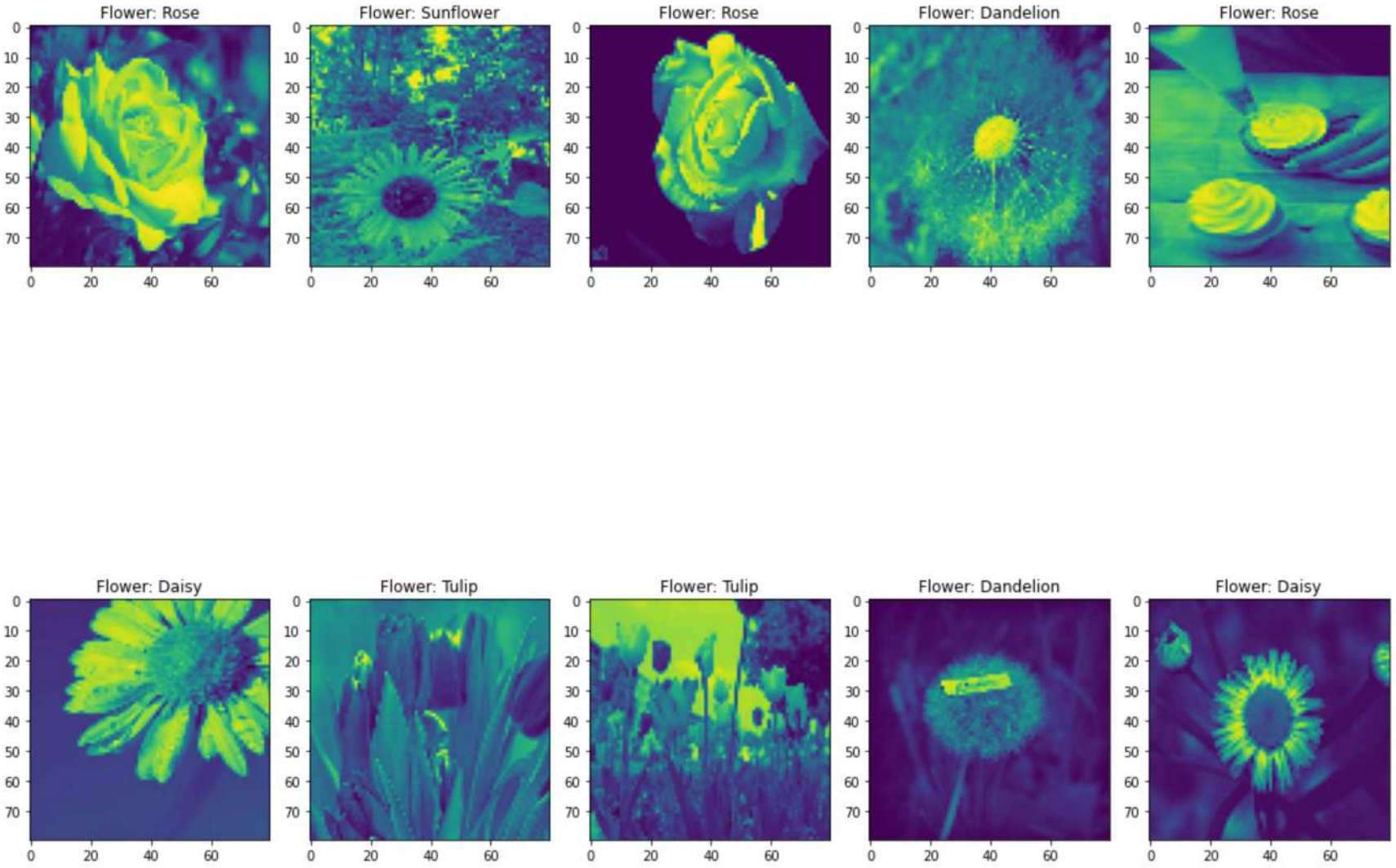
Define helper functions for plotting accuracy, loss, confusion matrix for all models

In [16]:
```
def draw_graph(history):
  plt.plot(history.history['accuracy'])
  plt.plot(history.history['val_accuracy'])
  plt.title('Model Accuracy')
  plt.ylabel('Accuracy')
  plt.xlabel('Epoch')
  plt.legend(['Training Accuracy', 'Validation Accuracy'], loc='upper left')
  plt.show()
  plt.plot(history.history['loss'])
  plt.plot(history.history['val_loss'])
  plt.title('Loss vs Epoch')
  plt.ylabel('Loss')
  plt.xlabel('Epoch')
  plt.legend(['Training Loss', 'Validation Loss'], loc='upper right')
  plt.show()

  print("Maximum training accuracy: ", max(history.history['accuracy']))
  print("Maximum validation accuracy: ", max(history.history['val_accuracy']))

def draw_matrix(model, train, test, train_labels, test_labels):
  predictions_test = []
  for item in test:
    predictions_test.append(np.argmax(model.predict(item.reshape(-1,80,80,1))))

  cm_test = sklearn.metrics.confusion_matrix(np.argmax(test_labels, axis = 1), predictions_test)
  sns.heatmap(cm_test, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_r');
  plt.ylabel('Actual label');
  plt.xlabel('Predicted label');
  all_sample_title = 'Confusion Matrix (Test samples)'
  plt.title(all_sample_title)
  plt.show()
  print(classification_report(np.argmax(test_labels, axis = 1), predictions_test))

def draw(History, model, train, test, train_labels, test_labels):
  draw_graph(History)
  draw_matrix(model, train, test, train_labels, test_labels)
```

Visualizing some of the images we loaded

In [17]:
```
fig,ax = plt.subplots(2,5)
fig.set_size_inches(15,15)
```

```
for i in range(2):
    for j in range (5):
        l = rn.randint(0, len(Z))
        ax[i,j].imshow(X[l])
        ax[i,j].set_title('Flower: '+ Z[l])

plt.tight_layout()
```



Pre-processing

One-Hot Encoding

In [18]:
```
le = LabelEncoder()
Y = le.fit_transform(Z)
Y = to_categorical(Y,5)
X = np.array(X)
X = X / 255
```

Train : Test split

In [19]:
```
train, test, train_labels, test_labels = train_test_split(X, Y, test_size=0.1, random_state=100)
train = train.reshape((-1,80,80,1))
test = test.reshape((-1,80,80,1))
```

Trying out the initial 4 models

Model 1

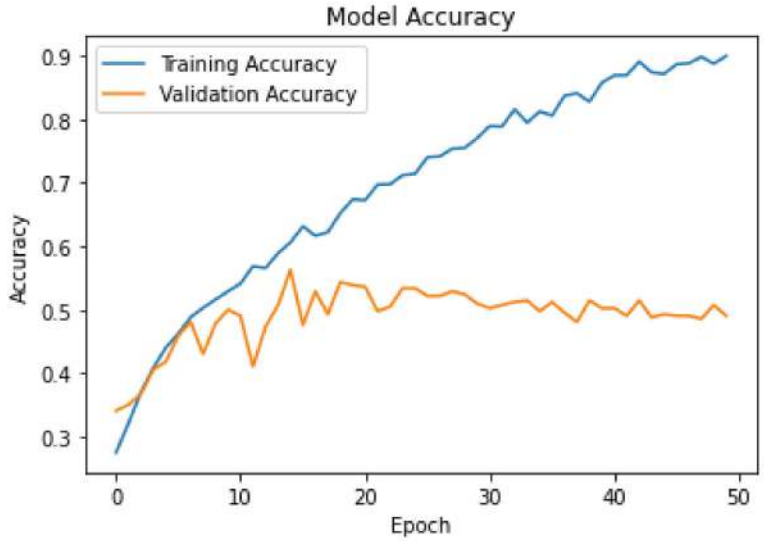
In [24]:
```
model1 = Sequential()

model1.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation ='relu', input_shape = (80,80,1)))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.1))

model1.add(Conv2D(filters = 32, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.1))

model1.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.1))

model1.add(Flatten())
model1.add(Dense(5, activation = 'softmax'))
```

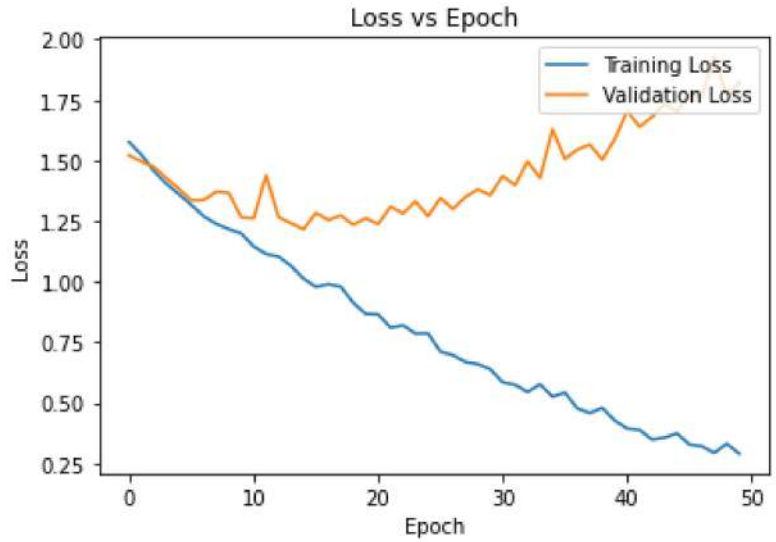
```
model1.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 80, 80, 16)        160

 max_pooling2d (MaxPooling2D  (None, 40, 40, 16)       0
 )

 dropout (Dropout)           (None, 40, 40, 16)        0

 conv2d_1 (Conv2D)           (None, 40, 40, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 20, 20, 32)       0
 2D)

 dropout_1 (Dropout)         (None, 20, 20, 32)        0

 conv2d_2 (Conv2D)           (None, 20, 20, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 10, 10, 64)       0
 2D)

 dropout_2 (Dropout)         (None, 10, 10, 64)        0

 flatten (Flatten)           (None, 6400)              0

 dense (Dense)               (None, 5)                 32005

=================================================================
Total params: 55,301
Trainable params: 55,301
Non-trainable params: 0
_________________________________________________________________
```

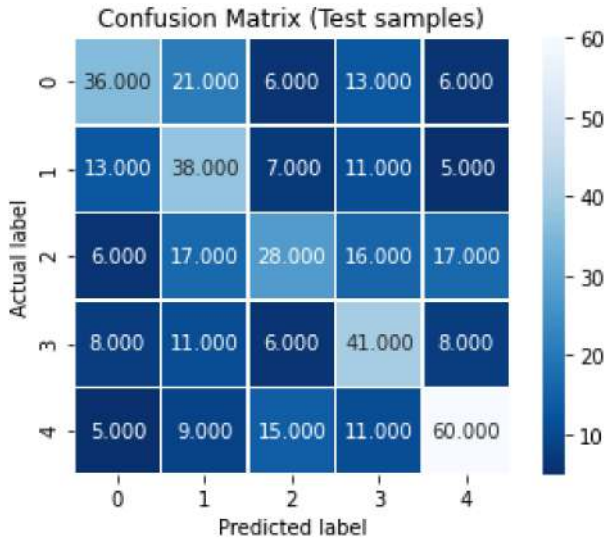
In [ ]:
```
History1 = model1.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History1, model1, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.8999999761581421
Maximum validation accuracy:  0.5628019571304321
```



	precision	recall	f1-score	support
0	0.53	0.44	0.48	82
1	0.40	0.51	0.45	74
2	0.45	0.33	0.38	84
3	0.45	0.55	0.49	74
4	0.62	0.60	0.61	100
accuracy			0.49	414
macro avg	0.49	0.49	0.48	414
weighted avg	0.50	0.49	0.49	414

Model 2

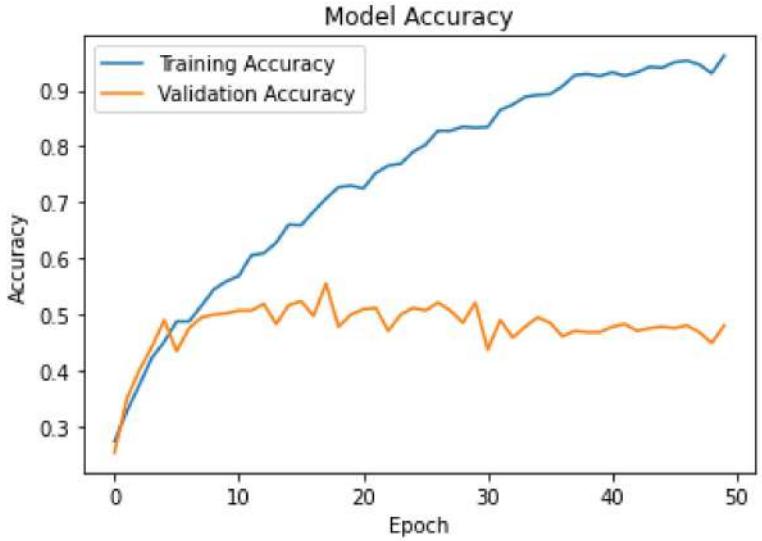
In [ ]:
```
model2 = Sequential()

model2.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation ='relu', input_shape = (80,80,1)))
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(Dropout(0.1))

model2.add(Conv2D(filters = 32, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(Dropout(0.1))

model2.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation ='relu'))
```

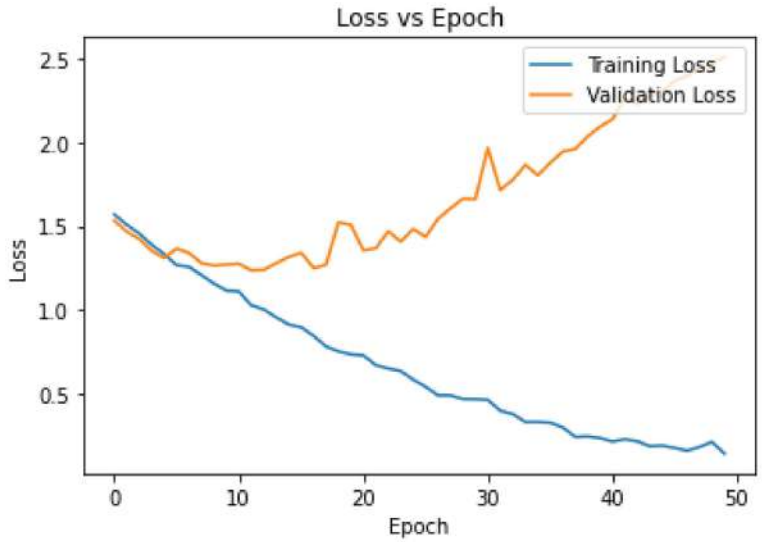
```
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(Dropout(0.1))

model2.add(Flatten())
model2.add(Dense(5, activation = 'softmax'))

model2.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 80, 80, 16)        160
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 40, 40, 16)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 40, 40, 32)        4640
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 20, 20, 32)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 10, 10, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6400)              0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 32005
=================================================================
Total params: 88,069
Trainable params: 88,069
Non-trainable params: 0
_________________________________________________________________
```

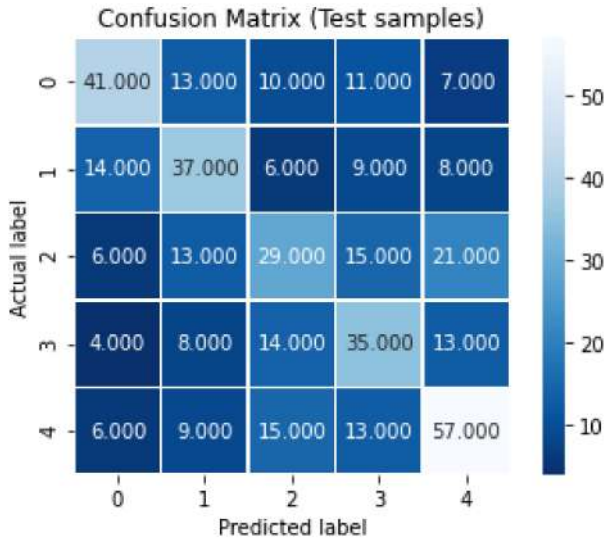
In [ ]:
```
History2 = model2.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History2, model2, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.9620967507362366
Maximum validation accuracy:  0.5555555820465088
```



	precision	recall	f1-score	support
0	0.58	0.50	0.54	82
1	0.46	0.50	0.48	74
2	0.39	0.35	0.37	84
3	0.42	0.47	0.45	74
4	0.54	0.57	0.55	100
accuracy			0.48	414
macro avg	0.48	0.48	0.48	414
weighted avg	0.48	0.48	0.48	414

Model 3

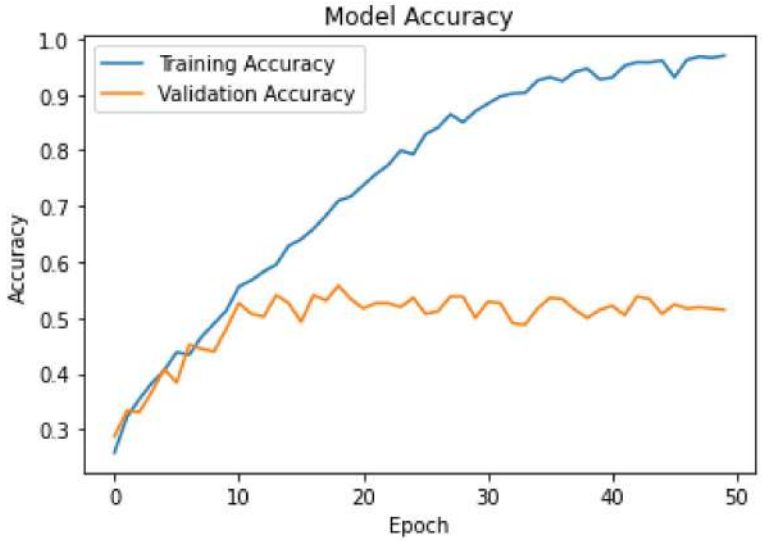
In [ ]:
```
model3 = Sequential()

model3.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation ='relu', input_shape = (80,80,1)))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Dropout(0.1))

model3.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu'))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Dropout(0.1))

model3.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation ='relu'))
```

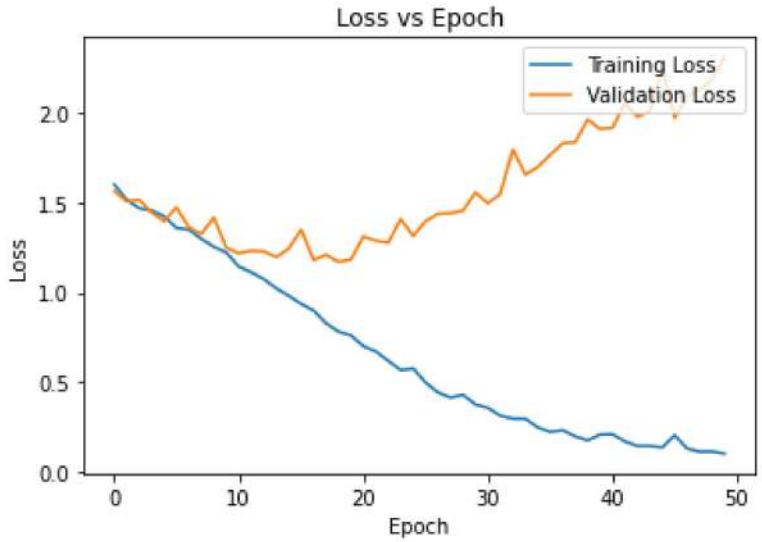
```
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Dropout(0.1))

model3.add(Flatten())
model3.add(Dense(5, activation = 'softmax'))

model3.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 80, 80, 16)        160
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 40, 40, 16)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 40, 40, 32)        12832
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 20, 20, 32)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
dropout_8 (Dropout)          (None, 10, 10, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6400)              0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 32005
=================================================================
Total params: 96,261
Trainable params: 96,261
Non-trainable params: 0
_________________________________________________________________
```

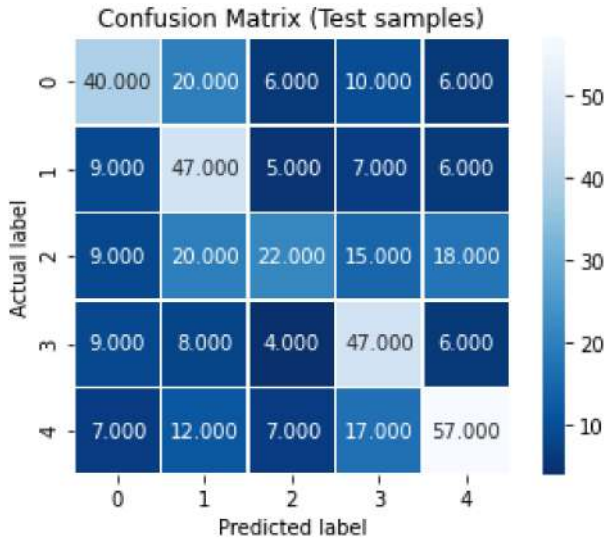
In [ ]:
```
History3 = model3.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History3, model3, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.9701613187789917
Maximum validation accuracy:  0.5579710006713867
```



	precision	recall	f1-score	support
0	0.54	0.49	0.51	82
1	0.44	0.64	0.52	74
2	0.50	0.26	0.34	84
3	0.49	0.64	0.55	74
4	0.61	0.57	0.59	100
accuracy			0.51	414
macro avg	0.52	0.52	0.50	414
weighted avg	0.52	0.51	0.51	414

Model 4

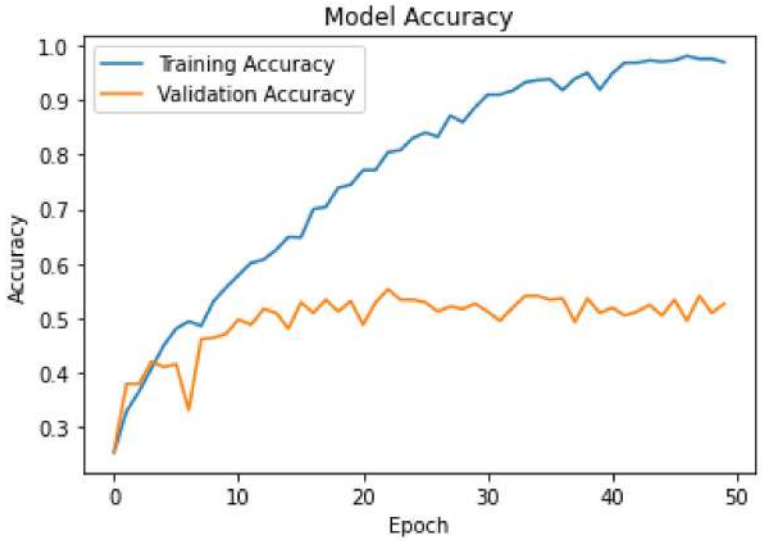
In [ ]:
```
model4 = Sequential()

model4.add(Conv2D(filters = 16, kernel_size = (5,5),padding = 'Same',activation ='relu', input_shape = (80,80,1)))
model4.add(MaxPooling2D(pool_size=(2, 2)))
model4.add(Dropout(0.1))

model4.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu'))
model4.add(MaxPooling2D(pool_size=(2, 2)))
model4.add(Dropout(0.1))

model4.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation ='relu'))
```

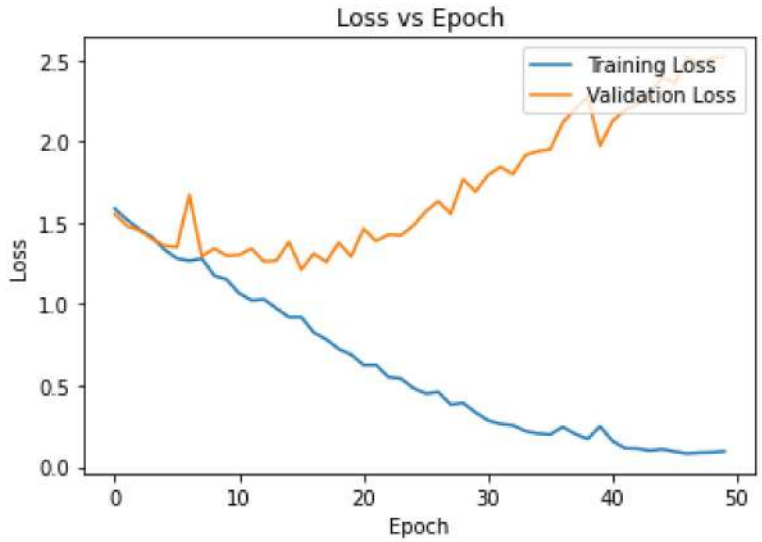
```
model4.add(MaxPooling2D(pool_size=(2, 2)))
model4.add(Dropout(0.1))

model4.add(Flatten())
model4.add(Dense(5, activation = 'softmax'))

model4.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model4.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 80, 80, 16)        416
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 40, 40, 16)        0
_________________________________________________________________
dropout_9 (Dropout)          (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 40, 40, 32)        12832
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 20, 20, 32)        0
_________________________________________________________________
dropout_10 (Dropout)         (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_11 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 6400)              0
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 32005
=================================================================
Total params: 96,517
Trainable params: 96,517
Non-trainable params: 0
_________________________________________________________________
```

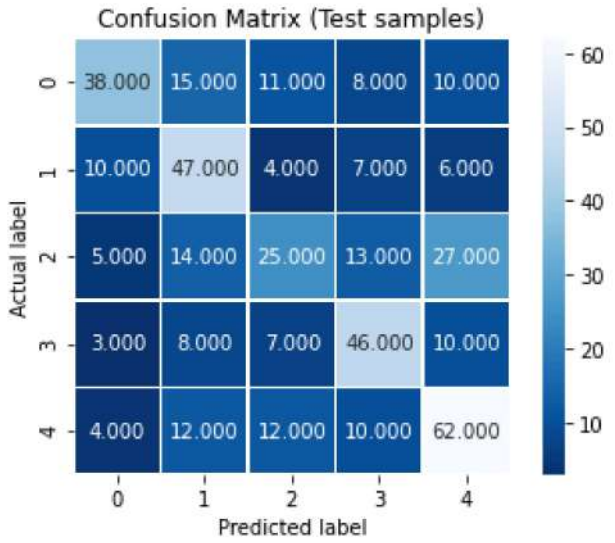
In [ ]:
```
History4 = model4.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History4, model4, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.9814516305923462
Maximum validation accuracy:  0.5531401038169861
```



	precision	recall	f1-score	support
0	0.63	0.46	0.54	82
1	0.49	0.64	0.55	74
2	0.42	0.30	0.35	84
3	0.55	0.62	0.58	74
4	0.54	0.62	0.58	100
accuracy			0.53	414
macro avg	0.53	0.53	0.52	414
weighted avg	0.53	0.53	0.52	414

Model 3 was the best model with an accuracy of 57.84% on the validation dataset.

The model is as follows:

```
model3 = Sequential()

model3.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation ='relu', input_shape =
(80,80,1)))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Dropout(0.1))
```

```
model3.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu'))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Dropout(0.1))

model3.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation ='relu'))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Dropout(0.1))

model3.add(Flatten())
model3.add(Dense(5, activation = 'softmax'))

model3.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model3.summary()
```

Now, we will be adding 2/3 fully-connected layers to this model.

Model 5

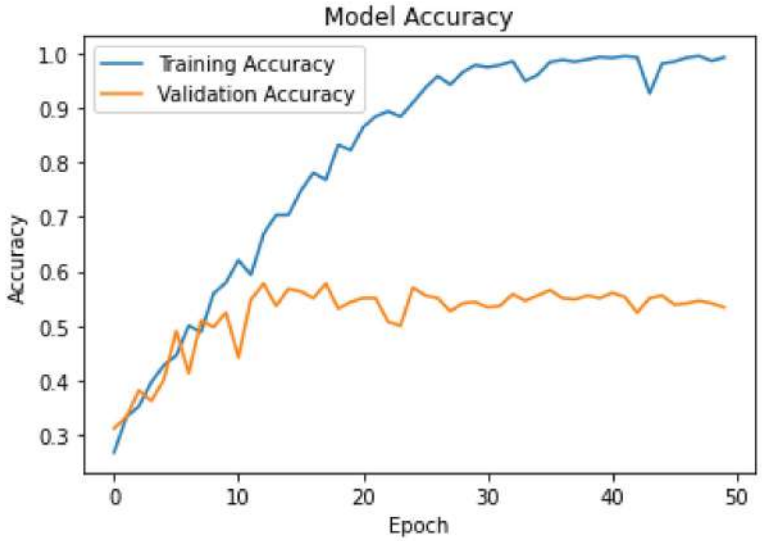
In [ ]:
```
model5 = Sequential()

model5.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation ='relu', input_shape = (80,80,1)))
model5.add(MaxPooling2D(pool_size=(2, 2)))
model5.add(Dropout(0.1))

model5.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu'))
model5.add(MaxPooling2D(pool_size=(2, 2)))
model5.add(Dropout(0.1))

model5.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation ='relu'))
model5.add(MaxPooling2D(pool_size=(2, 2)))
model5.add(Dropout(0.1))

model5.add(Flatten())
model5.add(Dense(128, activation = 'relu'))
model5.add(Dense(5, activation = 'softmax'))
```

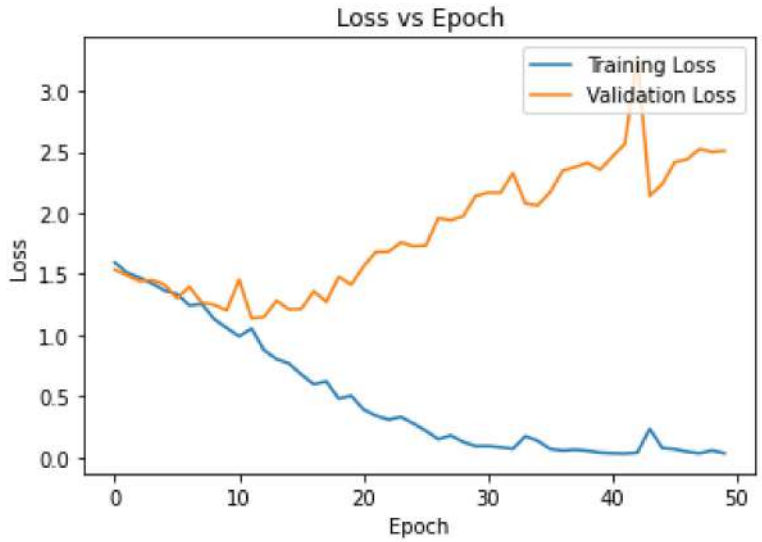
```
model5.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model5.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 80, 80, 16)        160
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 40, 40, 16)        0
_________________________________________________________________
dropout_12 (Dropout)         (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 40, 40, 32)        12832
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 20, 20, 32)        0
_________________________________________________________________
dropout_13 (Dropout)         (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_14 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 6400)              0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               819328
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 645
=================================================================
Total params: 884,229
Trainable params: 884,229
Non-trainable params: 0
_________________________________________________________________
```

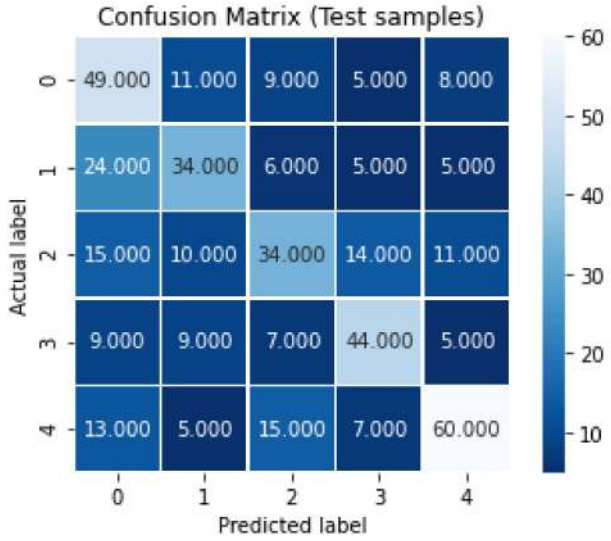
In [ ]:
```
History5 = model5.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History5, model5, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.9946236610412598
Maximum validation accuracy:  0.5772947072982788
```



	precision	recall	f1-score	support
0	0.45	0.60	0.51	82
1	0.49	0.46	0.48	74
2	0.48	0.40	0.44	84
3	0.59	0.59	0.59	74
4	0.67	0.60	0.63	100
accuracy			0.53	414
macro avg	0.54	0.53	0.53	414
weighted avg	0.54	0.53	0.53	414

Model 6

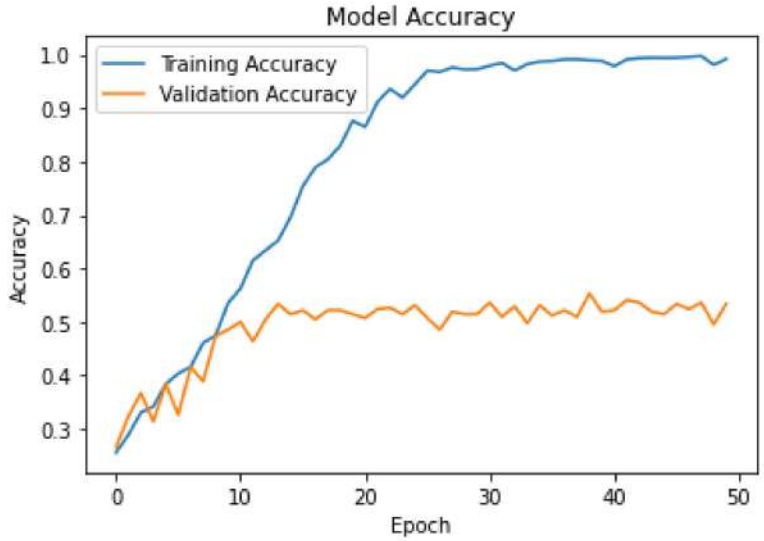
In [ ]:
```
model6 = Sequential()

model6.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation ='relu', input_shape = (80,80,1)))
model6.add(MaxPooling2D(pool_size=(2, 2)))
model6.add(Dropout(0.1))

model6.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu'))
model6.add(MaxPooling2D(pool_size=(2, 2)))
model6.add(Dropout(0.1))
model6.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation ='relu'))
model6.add(MaxPooling2D(pool_size=(2, 2)))
```

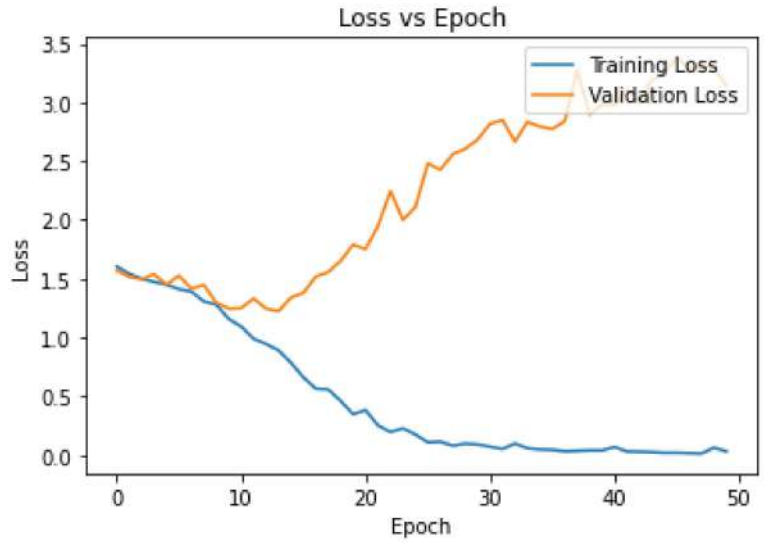
```
model6.add(Dropout(0.1))

model6.add(Flatten())
model6.add(Dense(128, activation = 'relu'))
model6.add(Dense(256, activation = 'relu'))
model6.add(Dense(5, activation = 'softmax'))

model6.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model6.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 80, 80, 16)        160
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 40, 40, 16)        0
_________________________________________________________________
dropout_15 (Dropout)         (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 40, 40, 32)        12832
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 20, 20, 32)        0
_________________________________________________________________
dropout_16 (Dropout)         (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_17 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
flatten_5 (Flatten)          (None, 6400)              0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               819328
_________________________________________________________________
dense_7 (Dense)              (None, 256)               33024
_________________________________________________________________
dense_8 (Dense)              (None, 5)                 1285
=================================================================
Total params: 917,893
Trainable params: 917,893
Non-trainable params: 0
_________________________________________________________________
```

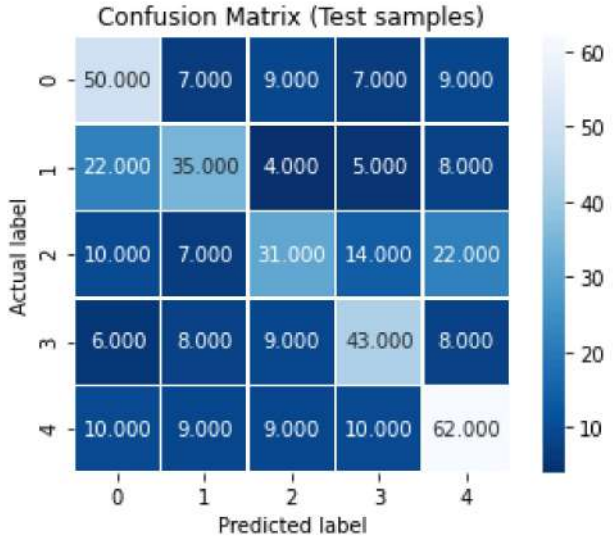
In [ ]:
```
History6 = model6.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History6, model6, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.9975806474685669
Maximum validation accuracy:  0.5531401038169861
```



	precision	recall	f1-score	support
0	0.51	0.61	0.56	82
1	0.53	0.47	0.50	74
2	0.50	0.37	0.42	84
3	0.54	0.58	0.56	74
4	0.57	0.62	0.59	100
accuracy			0.53	414
macro avg	0.53	0.53	0.53	414
weighted avg	0.53	0.53	0.53	414

Model 6 is the best model with an accuracy of 58.82% on the validation dataset.

```
model6 = Sequential()

model6.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation ='relu', input_shape =
(80,80,1)))
model6.add(MaxPooling2D(pool_size=(2, 2)))
model6.add(Dropout(0.1))

model6.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu'))
```

```
model6.add(MaxPooling2D(pool_size=(2, 2)))
model6.add(Dropout(0.1))
model6.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation ='relu'))
model6.add(MaxPooling2D(pool_size=(2, 2)))
model6.add(Dropout(0.1))

model6.add(Flatten())
model6.add(Dense(128, activation = 'relu'))
model6.add(Dense(256, activation = 'relu'))
model6.add(Dense(5, activation = 'softmax'))

model6.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model6.summary()
```

Now, we will be trying out average pooling on the same model instead of max pooling.

Model 7

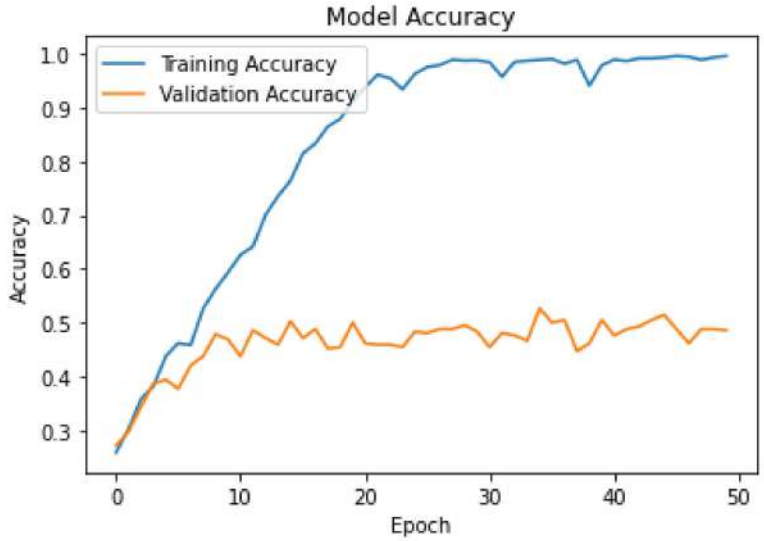
In [ ]:
```
model7 = Sequential()

model7.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation ='relu', input_shape = (80,80,1)))
model7.add(AveragePooling2D(pool_size=(2, 2)))
model7.add(Dropout(0.1))

model7.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu'))
model7.add(AveragePooling2D(pool_size=(2, 2)))
model7.add(Dropout(0.1))
model7.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation ='relu'))
model7.add(AveragePooling2D(pool_size=(2, 2)))
model7.add(Dropout(0.1))

model7.add(Flatten())
model7.add(Dense(128, activation = 'relu'))
model7.add(Dense(256, activation = 'relu'))
model7.add(Dense(5, activation = 'softmax'))
```

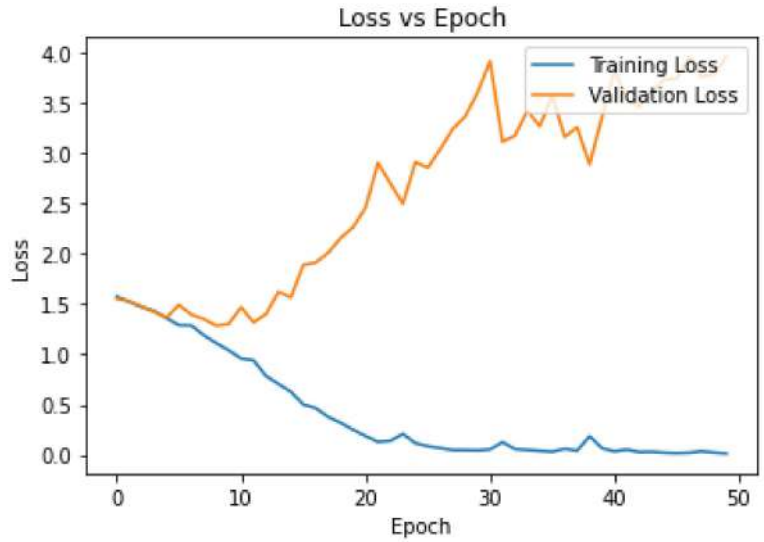
```
model7.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model7.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_18 (Conv2D)           (None, 80, 80, 16)        160
_________________________________________________________________
average_pooling2d (AveragePo (None, 40, 40, 16)        0
_________________________________________________________________
dropout_18 (Dropout)         (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 40, 40, 32)        12832
_________________________________________________________________
average_pooling2d_1 (Average (None, 20, 20, 32)        0
_________________________________________________________________
dropout_19 (Dropout)         (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
average_pooling2d_2 (Average (None, 10, 10, 64)        0
_________________________________________________________________
dropout_20 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
flatten_6 (Flatten)          (None, 6400)              0
_________________________________________________________________
dense_9 (Dense)              (None, 128)               819328
_________________________________________________________________
dense_10 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_11 (Dense)             (None, 5)                 1285
=================================================================
Total params: 917,893
Trainable params: 917,893
Non-trainable params: 0
_________________________________________________________________
```

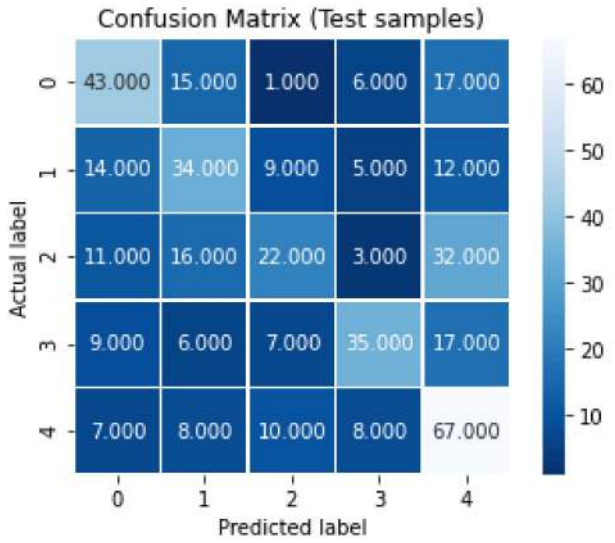
In [ ]:
```
History7 = model7.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History7, model7, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.9962365627288818
Maximum validation accuracy:  0.5265700221061707
```



	precision	recall	f1-score	support
0	0.51	0.52	0.52	82
1	0.43	0.46	0.44	74
2	0.45	0.26	0.33	84
3	0.61	0.47	0.53	74
4	0.46	0.67	0.55	100
accuracy			0.49	414
macro avg	0.49	0.48	0.47	414
weighted avg	0.49	0.49	0.48	414

We see that average pooling does not give us better results than max pooling. Hence, we stick with model 6 and vary the activation functions to see which one performs better on model 6.

Model 8

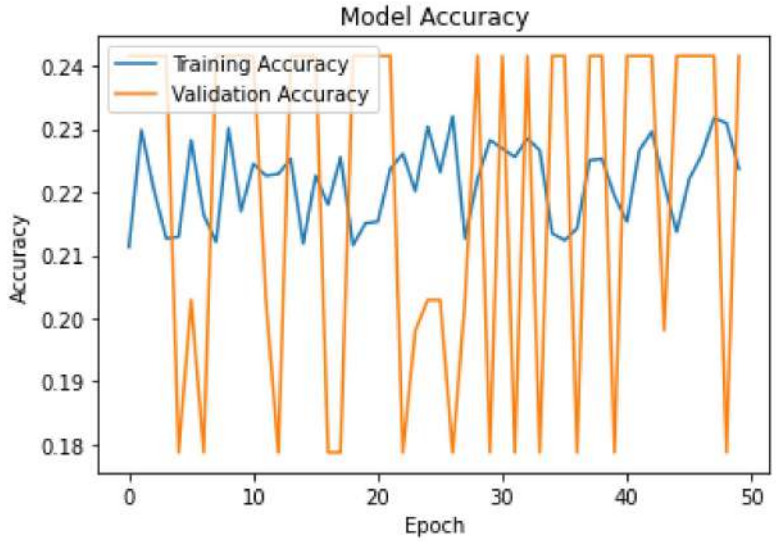
In [ ]:
```
model8 = Sequential()
```

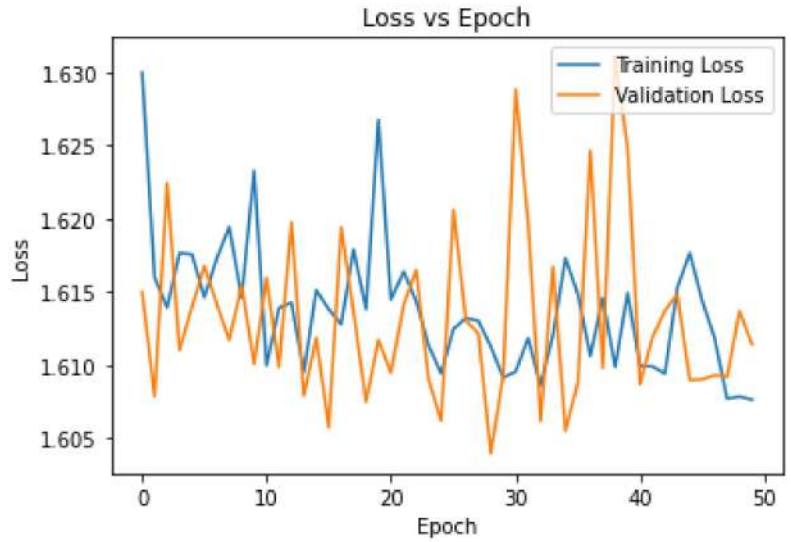
```
model8.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation ='sigmoid', input_shape = (80,80,1)))
model8.add(MaxPooling2D(pool_size=(2, 2)))
model8.add(Dropout(0.1))

model8.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='sigmoid'))
model8.add(MaxPooling2D(pool_size=(2, 2)))
model8.add(Dropout(0.1))
model8.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation ='sigmoid'))
model8.add(MaxPooling2D(pool_size=(2, 2)))
model8.add(Dropout(0.1))

model8.add(Flatten())
model8.add(Dense(128, activation ='sigmoid'))
model8.add(Dense(256, activation ='sigmoid'))
model8.add(Dense(5, activation = 'softmax'))

model8.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model8.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_21 (Conv2D)           (None, 80, 80, 16)        160
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 40, 40, 16)        0
_________________________________________________________________
dropout_21 (Dropout)         (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 40, 40, 32)        12832
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 20, 20, 32)        0
_________________________________________________________________
dropout_22 (Dropout)         (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_23 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
flatten_7 (Flatten)          (None, 6400)              0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               819328
_________________________________________________________________
dense_13 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_14 (Dense)             (None, 5)                 1285
=================================================================
Total params: 917,893
Trainable params: 917,893
Non-trainable params: 0
_________________________________________________________________
```

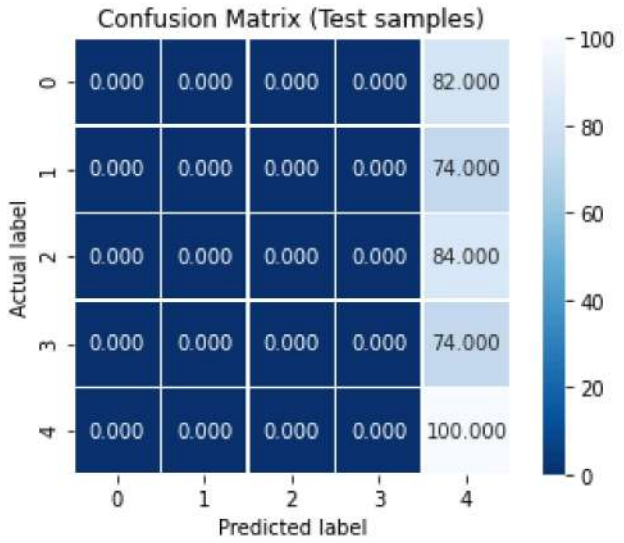
In [ ]:
```
History8 = model8.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History8, model8, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.23198924958705902
Maximum validation accuracy:  0.2415459007024765
```



	precision	recall	f1-score	support
0	0.00	0.00	0.00	82
1	0.00	0.00	0.00	74
2	0.00	0.00	0.00	84
3	0.00	0.00	0.00	74
4	0.24	1.00	0.39	100
accuracy			0.24	414
macro avg	0.05	0.20	0.08	414
weighted avg	0.06	0.24	0.09	414

```
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F
-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to contr
ol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Model 9

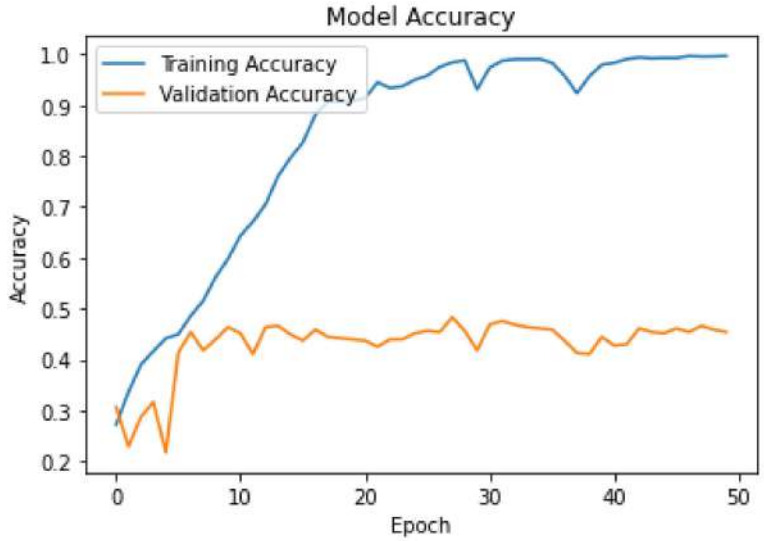
In [ ]:
```
model9 = Sequential()

model9.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation ='elu', input_shape = (80,80,1)))
model9.add(MaxPooling2D(pool_size=(2, 2)))
model9.add(Dropout(0.1))
```

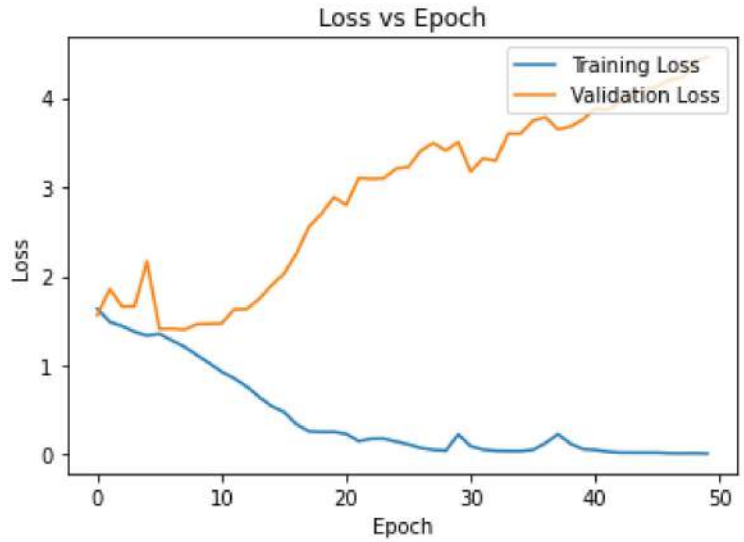
```
model9.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='elu'))
model9.add(MaxPooling2D(pool_size=(2, 2)))
model9.add(Dropout(0.1))
model9.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation ='elu'))
model9.add(MaxPooling2D(pool_size=(2, 2)))
model9.add(Dropout(0.1))

model9.add(Flatten())
model9.add(Dense(128, activation ='elu'))
model9.add(Dense(256, activation ='elu'))
model9.add(Dense(5, activation = 'softmax'))

model9.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model9.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_24 (Conv2D)           (None, 80, 80, 16)        160
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 40, 40, 16)        0
_________________________________________________________________
dropout_24 (Dropout)         (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 40, 40, 32)        12832
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 20, 20, 32)        0
_________________________________________________________________
dropout_25 (Dropout)         (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_26 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
flatten_8 (Flatten)          (None, 6400)              0
_________________________________________________________________
dense_15 (Dense)             (None, 128)               819328
_________________________________________________________________
dense_16 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_17 (Dense)             (None, 5)                 1285
=================================================================
Total params: 917,893
Trainable params: 917,893
Non-trainable params: 0
_________________________________________________________________
```

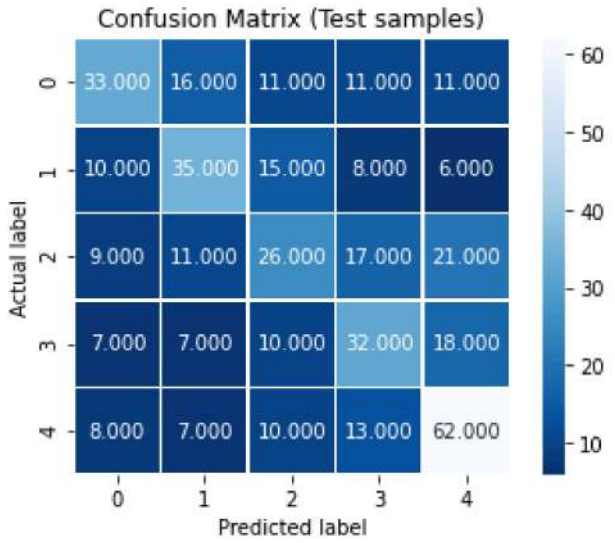
In [ ]:
```
History9 = model9.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History9, model9, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.9962365627288818
Maximum validation accuracy:  0.483091801404953
```



	precision	recall	f1-score	support
0	0.49	0.40	0.44	82
1	0.46	0.47	0.47	74
2	0.36	0.31	0.33	84
3	0.40	0.43	0.41	74
4	0.53	0.62	0.57	100
accuracy			0.45	414
macro avg	0.45	0.45	0.44	414
weighted avg	0.45	0.45	0.45	414

Model 10

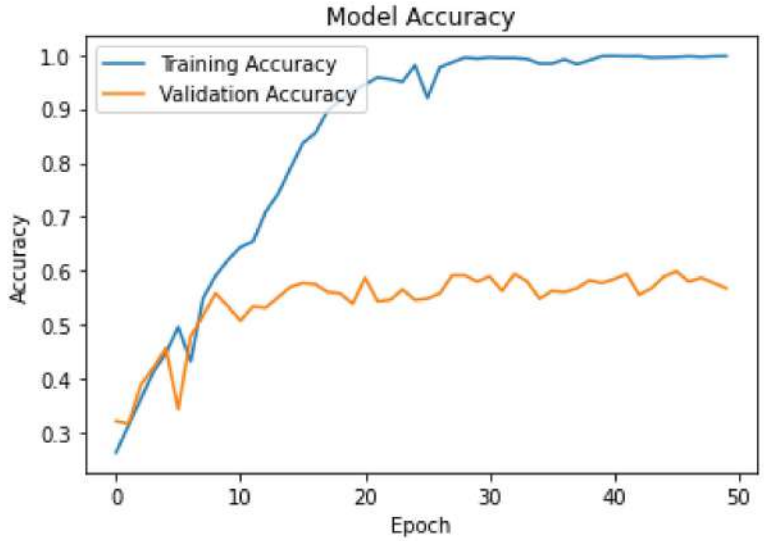
In [ ]:
```
leakyrelu = tf.keras.layers.LeakyReLU(alpha=0.01)
model10 = Sequential()

model10.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation = leakyrelu, input_shape = (80,80,1)))
model10.add(MaxPooling2D(pool_size=(2, 2)))
model10.add(Dropout(0.1))

model10.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model10.add(MaxPooling2D(pool_size=(2, 2)))
model10.add(Dropout(0.1))
model10.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
```

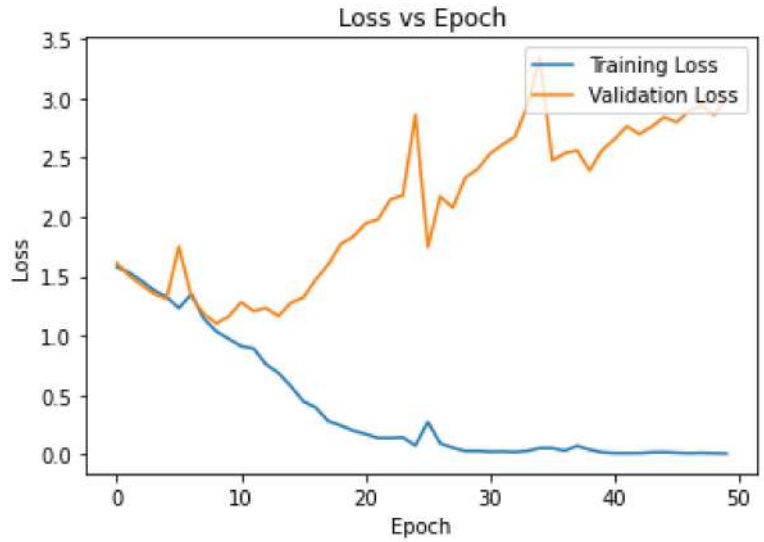
```
model10.add(MaxPooling2D(pool_size=(2, 2)))
model10.add(Dropout(0.1))

model10.add(Flatten())
model10.add(Dense(128, activation = leakyrelu))
model10.add(Dense(256, activation = leakyrelu))
model10.add(Dense(5, activation = 'softmax'))

model10.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model10.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_27 (Conv2D)           (None, 80, 80, 16)        160
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 40, 40, 16)        0
_________________________________________________________________
dropout_27 (Dropout)         (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 40, 40, 32)        12832
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 20, 20, 32)        0
_________________________________________________________________
dropout_28 (Dropout)         (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_29 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
flatten_9 (Flatten)          (None, 6400)              0
_________________________________________________________________
dense_18 (Dense)             (None, 128)               819328
_________________________________________________________________
dense_19 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_20 (Dense)             (None, 5)                 1285
=================================================================
Total params: 917,893
Trainable params: 917,893
Non-trainable params: 0
_________________________________________________________________
```

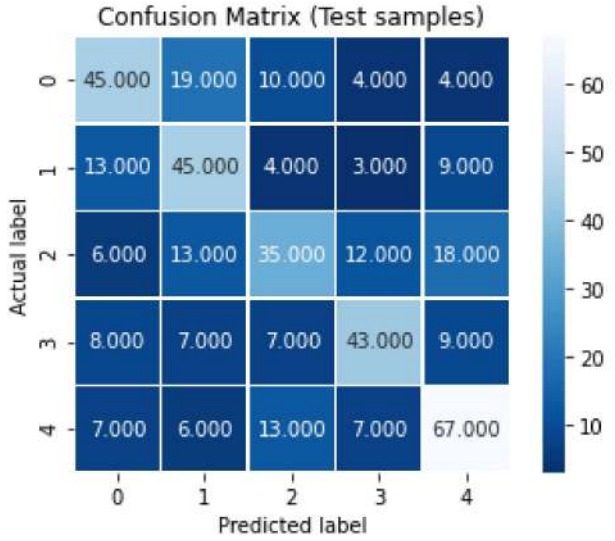
In [ ]:
```
History10 = model10.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History10, model10, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.9981182813644409
Maximum validation accuracy:  0.5990338325500488
```



	precision	recall	f1-score	support
0	0.57	0.55	0.56	82
1	0.50	0.61	0.55	74
2	0.51	0.42	0.46	84
3	0.62	0.58	0.60	74
4	0.63	0.67	0.65	100
accuracy			0.57	414
macro avg	0.57	0.56	0.56	414
weighted avg	0.57	0.57	0.57	414

After modifying the activation functions, we see that model 10's accuracy is the highest.

```
leakyrelu = tf.keras.layers.LeakyReLU(alpha=0.01)
model10 = Sequential()

model10.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation = leakyrelu, input_shape =
(80,80,1)))
model10.add(MaxPooling2D(pool_size=(2, 2)))
model10.add(Dropout(0.1))
```

```
model10.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model10.add(MaxPooling2D(pool_size=(2, 2)))
model10.add(Dropout(0.1))
model10.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model10.add(MaxPooling2D(pool_size=(2, 2)))
model10.add(Dropout(0.1))

model10.add(Flatten())
model10.add(Dense(128, activation = leakyrelu))
model10.add(Dense(256, activation = leakyrelu))
model10.add(Dense(5, activation = 'softmax'))

model10.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model10.summary()
```

Thus, we will now vary the regularization parameters for model 10.

Model 11

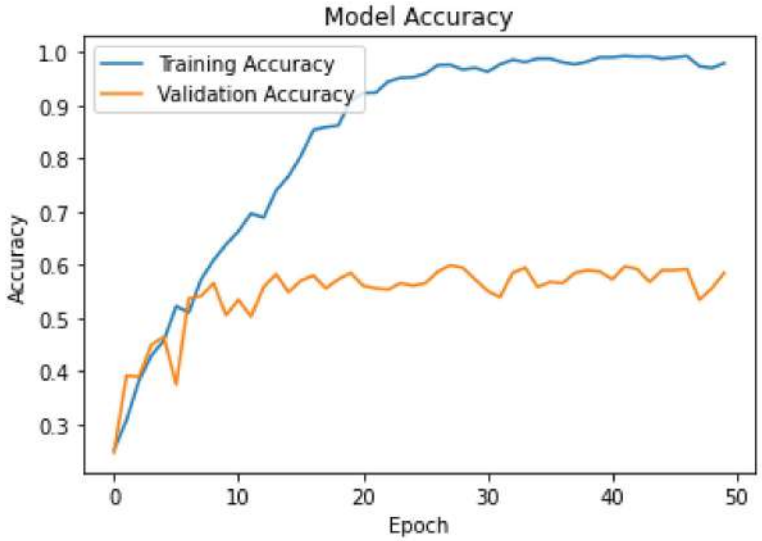
In [ ]:
```
leakyrelu = tf.keras.layers.LeakyReLU(alpha=0.01)
model11 = Sequential()

model11.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation = leakyrelu, input_shape = (80,80,1)))
model11.add(MaxPooling2D(pool_size=(2, 2)))
model11.add(Dropout(0.25))

model11.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model11.add(MaxPooling2D(pool_size=(2, 2)))
model11.add(Dropout(0.25))

model11.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model11.add(MaxPooling2D(pool_size=(2, 2)))
model11.add(Dropout(0.25))

model11.add(Flatten())
model11.add(Dense(128, activation = leakyrelu))
model11.add(Dense(256, activation = leakyrelu))
```

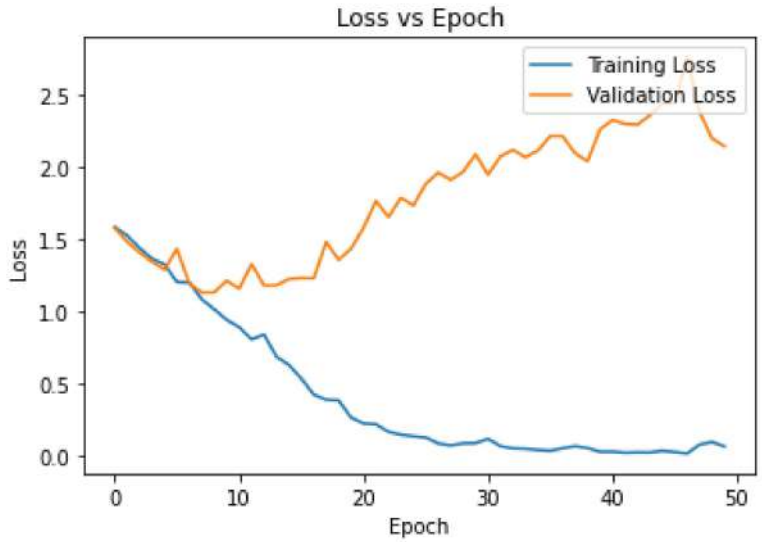
```
model11.add(Dense(5, activation = 'softmax'))

model11.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model11.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_30 (Conv2D)           (None, 80, 80, 16)        160
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 40, 40, 16)        0
_________________________________________________________________
dropout_30 (Dropout)         (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 40, 40, 32)        12832
_________________________________________________________________
max_pooling2d_28 (MaxPooling (None, 20, 20, 32)        0
_________________________________________________________________
dropout_31 (Dropout)         (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_29 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_32 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
flatten_10 (Flatten)         (None, 6400)              0
_________________________________________________________________
dense_21 (Dense)             (None, 128)               819328
_________________________________________________________________
dense_22 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_23 (Dense)             (None, 5)                 1285
=================================================================
Total params: 917,893
Trainable params: 917,893
Non-trainable params: 0
_________________________________________________________________
```

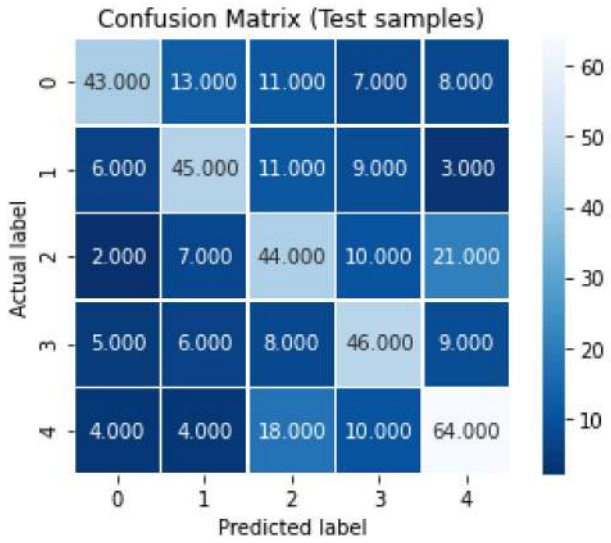
In [ ]:
```
History11 = model11.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History11, model11, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.9927419424057007
Maximum validation accuracy:  0.5990338325500488
```



	precision	recall	f1-score	support
0	0.72	0.52	0.61	82
1	0.60	0.61	0.60	74
2	0.48	0.52	0.50	84
3	0.56	0.62	0.59	74
4	0.61	0.64	0.62	100
accuracy			0.58	414
macro avg	0.59	0.58	0.58	414
weighted avg	0.59	0.58	0.59	414

Model 12

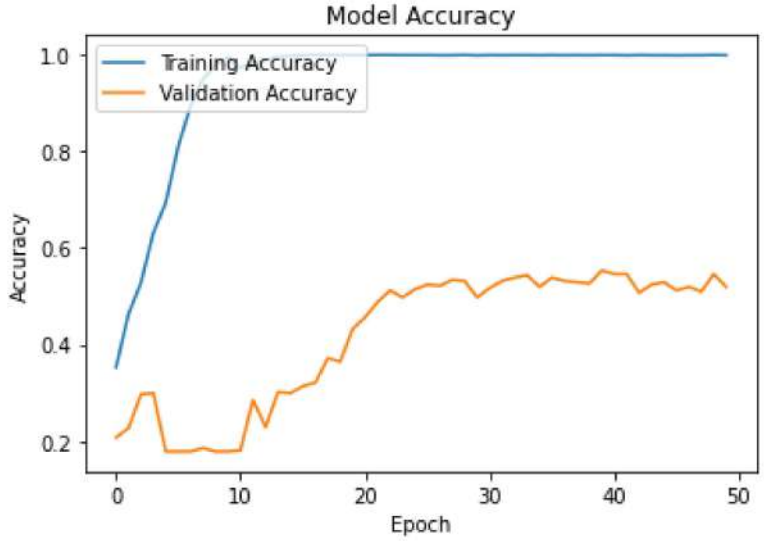
In [ ]:
```
leakyrelu = tf.keras.layers.LeakyReLU(alpha=0.01)
model12 = Sequential()

model12.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation = leakyrelu, input_shape = (80,80,1)))
model12.add(MaxPooling2D(pool_size=(2, 2)))

model12.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model12.add(MaxPooling2D(pool_size=(2, 2)))
model12.add(BatchNormalization())

model12.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
```

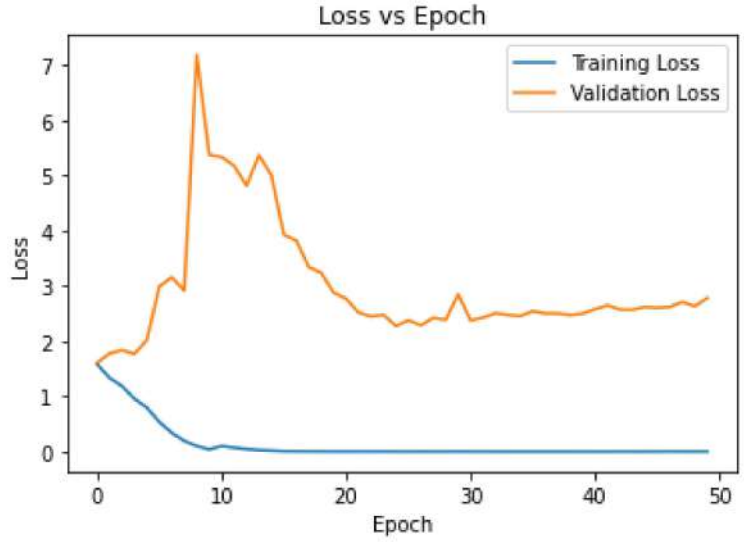
```
model12.add(MaxPooling2D(pool_size=(2, 2)))
model12.add(BatchNormalization())

model12.add(Flatten())
model12.add(Dense(128, activation = leakyrelu))
model12.add(Dense(256, activation = leakyrelu))
model12.add(Dense(5, activation = 'softmax'))

model12.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model12.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_33 (Conv2D)           (None, 80, 80, 16)        160
_________________________________________________________________
max_pooling2d_30 (MaxPooling (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 40, 40, 32)        12832
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 20, 20, 32)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 20, 20, 32)        128
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 10, 10, 64)        256
_________________________________________________________________
flatten_11 (Flatten)         (None, 6400)              0
_________________________________________________________________
dense_24 (Dense)             (None, 128)               819328
_________________________________________________________________
dense_25 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_26 (Dense)             (None, 5)                 1285
=================================================================
Total params: 918,277
Trainable params: 918,085
Non-trainable params: 192
_________________________________________________________________
```

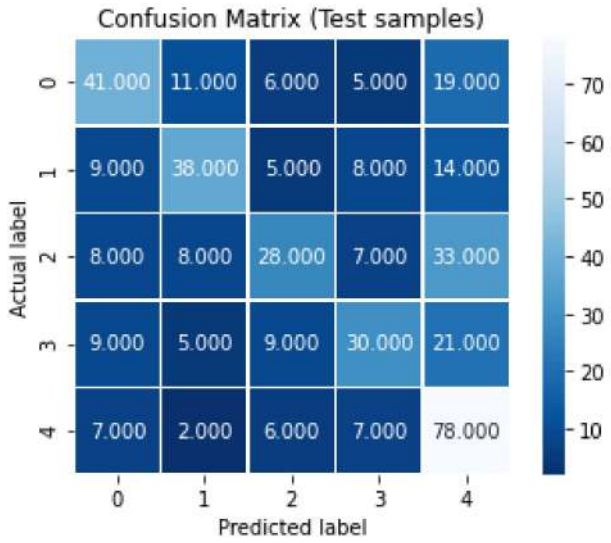
In [ ]:
```
History12 = model12.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History12, model12, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.998924732208252
Maximum validation accuracy:  0.5531401038169861
```



	precision	recall	f1-score	support
0	0.55	0.50	0.53	82
1	0.59	0.51	0.55	74
2	0.52	0.33	0.41	84
3	0.53	0.41	0.46	74
4	0.47	0.78	0.59	100
accuracy			0.52	414
macro avg	0.53	0.51	0.51	414
weighted avg	0.53	0.52	0.51	414

Model 13

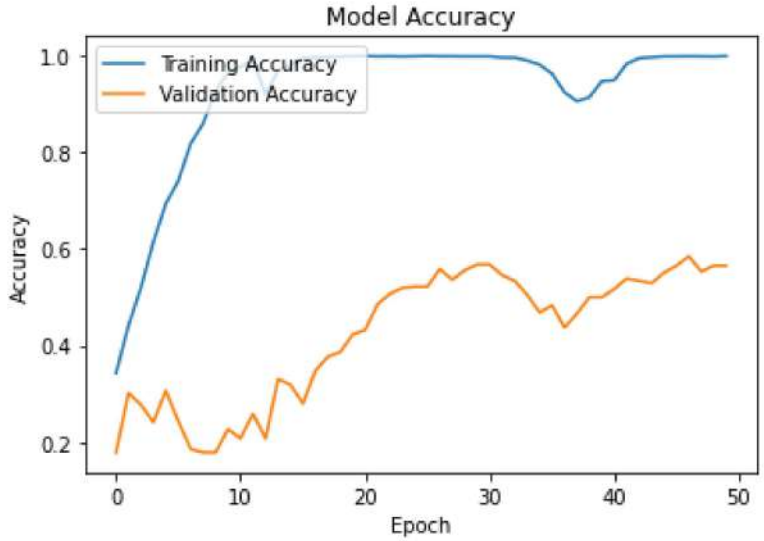
In [ ]:
```
leakyrelu = tf.keras.layers.LeakyReLU(alpha=0.01)
model13 = Sequential()

model13.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation = leakyrelu, input_shape = (80,80,1)))
model13.add(MaxPooling2D(pool_size=(2, 2)))
model13.add(Dropout(0.1))

model13.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model13.add(MaxPooling2D(pool_size=(2, 2)))
model13.add(BatchNormalization())
model13.add(Dropout(0.1))
```

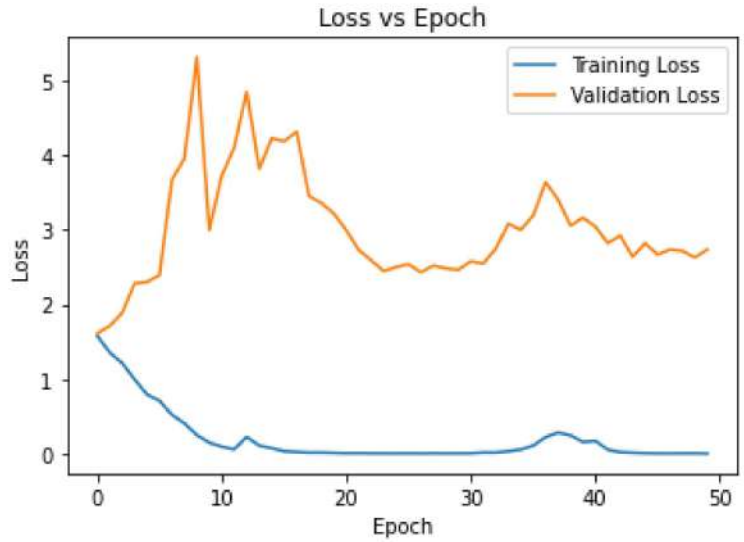
```
model13.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model13.add(MaxPooling2D(pool_size=(2, 2)))
model13.add(BatchNormalization())
model13.add(Dropout(0.1))

model13.add(Flatten())
model13.add(Dense(128, activation = leakyrelu))
model13.add(Dense(256, activation = leakyrelu))
model13.add(Dense(5, activation = 'softmax'))

model13.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model13.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_36 (Conv2D)           (None, 80, 80, 16)        160
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 40, 40, 16)        0
_________________________________________________________________
dropout_33 (Dropout)         (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 40, 40, 32)        12832
_________________________________________________________________
max_pooling2d_34 (MaxPooling (None, 20, 20, 32)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 20, 20, 32)        128
_________________________________________________________________
dropout_34 (Dropout)         (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_35 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 10, 10, 64)        256
_________________________________________________________________
dropout_35 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
flatten_12 (Flatten)         (None, 6400)              0
_________________________________________________________________
dense_27 (Dense)             (None, 128)               819328
_________________________________________________________________
dense_28 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_29 (Dense)             (None, 5)                 1285
=================================================================
Total params: 918,277
Trainable params: 918,085
Non-trainable params: 192
_________________________________________________________________
```

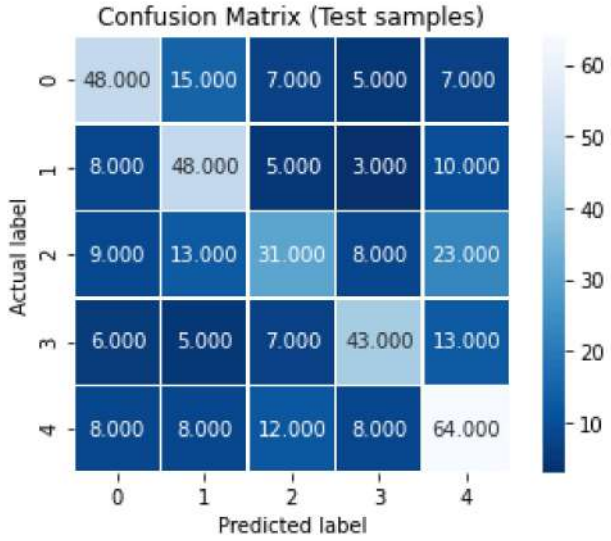
In [ ]:
```
History13 = model13.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History13, model13, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.998924732208252
Maximum validation accuracy:  0.5845410823822021
```



	precision	recall	f1-score	support
0	0.61	0.59	0.60	82
1	0.54	0.65	0.59	74
2	0.50	0.37	0.42	84
3	0.64	0.58	0.61	74
4	0.55	0.64	0.59	100
accuracy			0.57	414
macro avg	0.57	0.56	0.56	414
weighted avg	0.57	0.57	0.56	414

Model 10 still performs the best. Now, we will be adding more convolution layers to it and check if it performs better.

Model 14

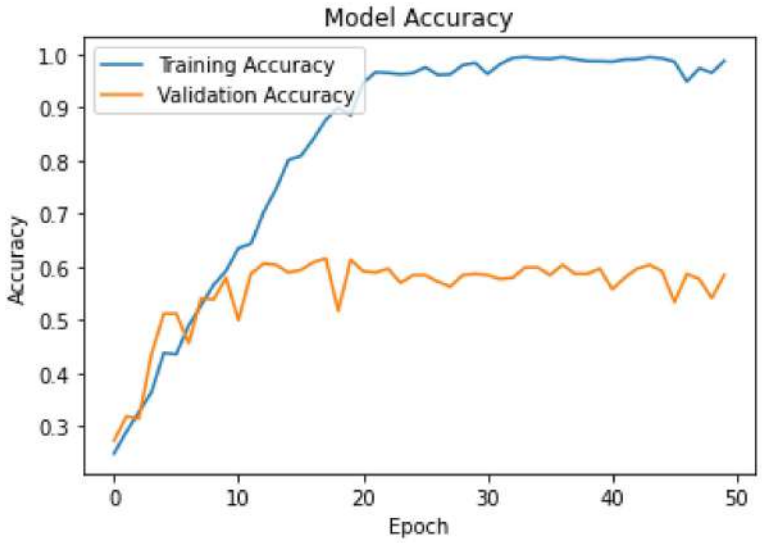
In [ ]:
```
leakyrelu = tf.keras.layers.LeakyReLU(alpha=0.01)
model14 = Sequential()

model14.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation = leakyrelu, input_shape = (80,80,1)))
model14.add(MaxPooling2D(pool_size=(2, 2)))
```

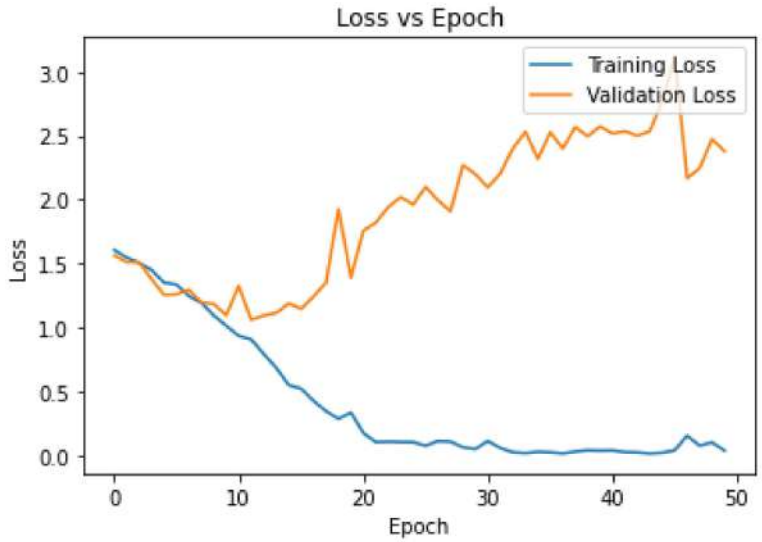
```
model14.add(Dropout(0.1))

model14.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model14.add(MaxPooling2D(pool_size=(2, 2)))
model14.add(Dropout(0.1))

model14.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model14.add(MaxPooling2D(pool_size=(2, 2)))
model14.add(Dropout(0.1))

model14.add(Conv2D(filters = 128, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model14.add(MaxPooling2D(pool_size=(2, 2)))
model14.add(Dropout(0.1))

model14.add(Flatten())
model14.add(Dense(128, activation = leakyrelu))
model14.add(Dense(256, activation = leakyrelu))
model14.add(Dense(5, activation = 'softmax'))

model14.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model14.summary()
```

```
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_39 (Conv2D)           (None, 80, 80, 16)        160
_________________________________________________________________
max_pooling2d_36 (MaxPooling (None, 40, 40, 16)        0
_________________________________________________________________
dropout_36 (Dropout)         (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 40, 40, 32)        12832
_________________________________________________________________
max_pooling2d_37 (MaxPooling (None, 20, 20, 32)        0
_________________________________________________________________
dropout_37 (Dropout)         (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_38 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_38 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 10, 10, 128)       204928
_________________________________________________________________
max_pooling2d_39 (MaxPooling (None, 5, 5, 128)         0
_________________________________________________________________
dropout_39 (Dropout)         (None, 5, 5, 128)         0
_________________________________________________________________
flatten_13 (Flatten)         (None, 3200)              0
_________________________________________________________________
dense_30 (Dense)             (None, 128)               409728
_________________________________________________________________
dense_31 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_32 (Dense)             (None, 5)                 1285
=================================================================
Total params: 713,221
Trainable params: 713,221
Non-trainable params: 0
_________________________________________________________________
```

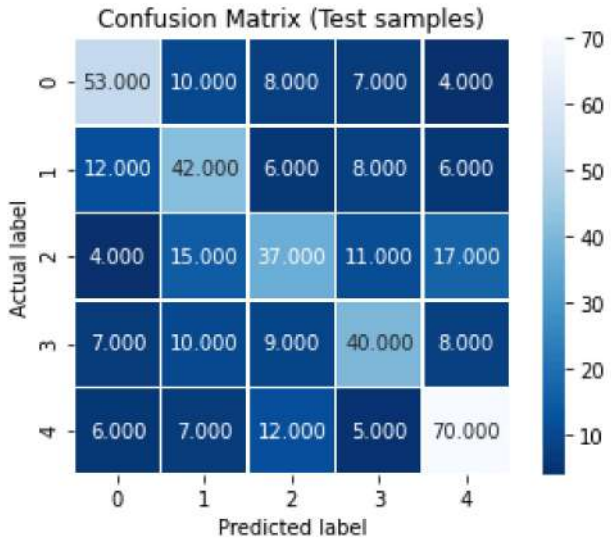
In [ ]:
```
History14 = model14.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History14, model14, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.9951612949371338
Maximum validation accuracy:  0.6159420013427734
```



	precision	recall	f1-score	support
0	0.65	0.65	0.65	82
1	0.50	0.57	0.53	74
2	0.51	0.44	0.47	84
3	0.56	0.54	0.55	74
4	0.67	0.70	0.68	100
accuracy			0.58	414
macro avg	0.58	0.58	0.58	414
weighted avg	0.58	0.58	0.58	414

Model 15

In [ ]:
```
leakyrelu = tf.keras.layers.LeakyReLU(alpha=0.01)
model15 = Sequential()

model15.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation = leakyrelu, input_shape = (80,80,1)))
model15.add(MaxPooling2D(pool_size=(2, 2)))
model15.add(Dropout(0.1))

model15.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model15.add(MaxPooling2D(pool_size=(2, 2)))
model15.add(Dropout(0.1))
```

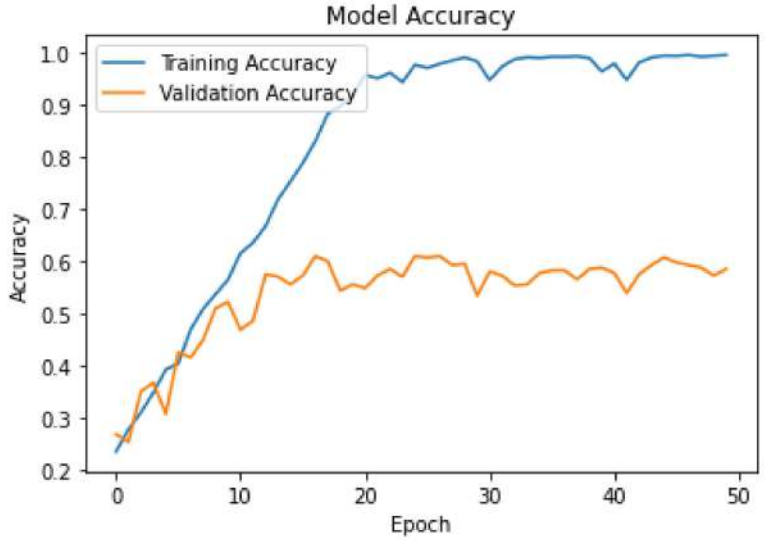
```
model15.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model15.add(MaxPooling2D(pool_size=(2, 2)))
model15.add(Dropout(0.1))

model15.add(Conv2D(filters = 128, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model15.add(MaxPooling2D(pool_size=(2, 2)))
model15.add(Dropout(0.1))

model15.add(Conv2D(filters = 256, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model15.add(MaxPooling2D(pool_size=(2, 2)))
model15.add(Dropout(0.1))

model15.add(Flatten())
model15.add(Dense(128, activation = leakyrelu))
model15.add(Dense(256, activation = leakyrelu))
model15.add(Dense(5, activation = 'softmax'))

model15.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model15.summary()
```

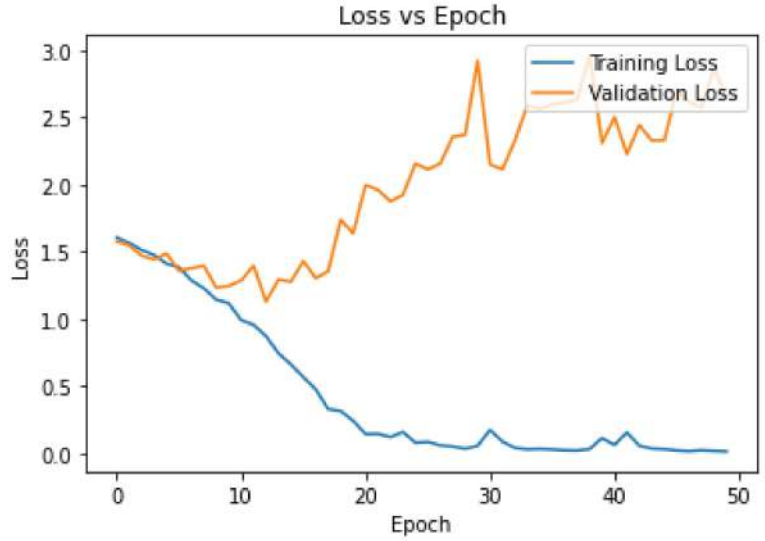
```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_43 (Conv2D)           (None, 80, 80, 16)        160
_________________________________________________________________
max_pooling2d_40 (MaxPooling (None, 40, 40, 16)        0
_________________________________________________________________
dropout_40 (Dropout)         (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_44 (Conv2D)           (None, 40, 40, 32)        12832
_________________________________________________________________
max_pooling2d_41 (MaxPooling (None, 20, 20, 32)        0
_________________________________________________________________
dropout_41 (Dropout)         (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_45 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_42 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_42 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 10, 10, 128)       204928
_________________________________________________________________
max_pooling2d_43 (MaxPooling (None, 5, 5, 128)         0
_________________________________________________________________
dropout_43 (Dropout)         (None, 5, 5, 128)         0
_________________________________________________________________
conv2d_47 (Conv2D)           (None, 5, 5, 256)         819456
_________________________________________________________________
max_pooling2d_44 (MaxPooling (None, 2, 2, 256)         0
_________________________________________________________________
dropout_44 (Dropout)         (None, 2, 2, 256)         0
_________________________________________________________________
flatten_14 (Flatten)         (None, 1024)              0
_________________________________________________________________
dense_33 (Dense)             (None, 128)               131200
_________________________________________________________________
dense_34 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_35 (Dense)             (None, 5)                 1285
=================================================================
Total params: 1,254,149
```

```
Trainable params: 1,254,149
Non-trainable params: 0
_________________________________________________________________
```

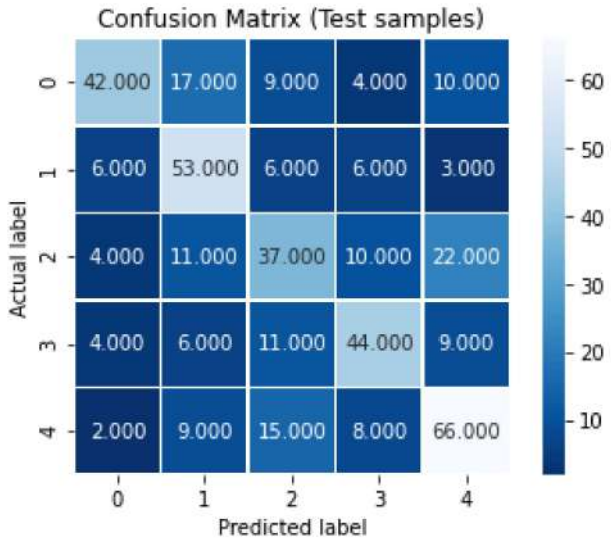
In [ ]:
```
History15 = model15.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History15, model15, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.9943548440933228
Maximum validation accuracy:  0.6086956262588501
```



	precision	recall	f1-score	support
0	0.72	0.51	0.60	82
1	0.55	0.72	0.62	74
2	0.47	0.44	0.46	84
3	0.61	0.59	0.60	74
4	0.60	0.66	0.63	100
accuracy			0.58	414
macro avg	0.59	0.58	0.58	414
weighted avg	0.59	0.58	0.58	414

Model 16

In [26]:
```
leakyrelu = tf.keras.layers.LeakyReLU(alpha=0.01)
model16 = Sequential()

model16.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation = leakyrelu, input_shape = (80,80,1)))
model16.add(MaxPooling2D(pool_size=(2, 2)))
model16.add(Dropout(0.1))

model16.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model16.add(MaxPooling2D(pool_size=(2, 2)))
model16.add(Dropout(0.1))
```

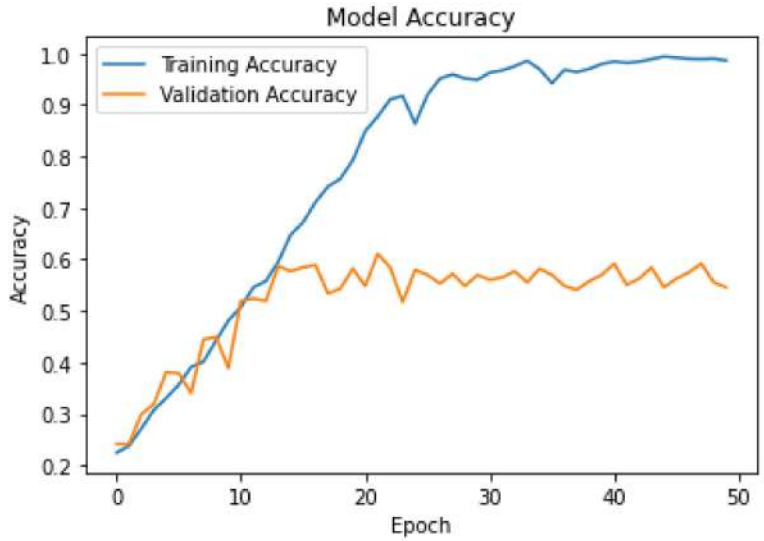
```
model16.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model16.add(MaxPooling2D(pool_size=(2, 2)))
model16.add(Dropout(0.1))

model16.add(Conv2D(filters = 128, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model16.add(MaxPooling2D(pool_size=(2, 2)))
model16.add(Dropout(0.1))

model16.add(Conv2D(filters = 256, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model16.add(MaxPooling2D(pool_size=(2, 2)))
model16.add(Dropout(0.1))

model16.add(Conv2D(filters = 512, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model16.add(MaxPooling2D(pool_size=(2, 2)))
model16.add(Dropout(0.1))

model16.add(Flatten())
model16.add(Dense(128, activation = leakyrelu))
model16.add(Dense(256, activation = leakyrelu))
model16.add(Dense(5, activation = 'softmax'))

model16.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model16.summary()
```

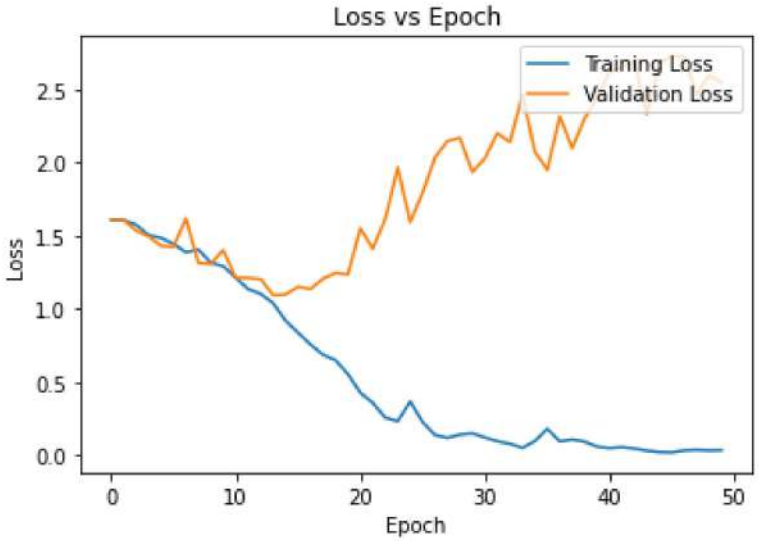
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 80, 80, 16)        160

 max_pooling2d_3 (MaxPooling  (None, 40, 40, 16)       0
 2D)

 dropout_3 (Dropout)         (None, 40, 40, 16)        0

 conv2d_4 (Conv2D)           (None, 40, 40, 32)        12832

 max_pooling2d_4 (MaxPooling  (None, 20, 20, 32)       0
 2D)

 dropout_4 (Dropout)         (None, 20, 20, 32)        0

 conv2d_5 (Conv2D)           (None, 20, 20, 64)        51264

 max_pooling2d_5 (MaxPooling  (None, 10, 10, 64)       0
 2D)

 dropout_5 (Dropout)         (None, 10, 10, 64)        0

 conv2d_6 (Conv2D)           (None, 10, 10, 128)       204928

 max_pooling2d_6 (MaxPooling  (None, 5, 5, 128)        0
 2D)

 dropout_6 (Dropout)         (None, 5, 5, 128)         0

 conv2d_7 (Conv2D)           (None, 5, 5, 256)         819456

 max_pooling2d_7 (MaxPooling  (None, 2, 2, 256)        0
 2D)

 dropout_7 (Dropout)         (None, 2, 2, 256)         0

 conv2d_8 (Conv2D)           (None, 2, 2, 512)         3277312

 max_pooling2d_8 (MaxPooling  (None, 1, 1, 512)        0
 2D)
```

```
dropout_8 (Dropout)          (None, 1, 1, 512)         0

flatten_1 (Flatten)          (None, 512)               0

dense_1 (Dense)              (None, 128)               65664

dense_2 (Dense)              (None, 256)               33024

dense_3 (Dense)              (None, 5)                 1285

=================================================================
Total params: 4,465,925
Trainable params: 4,465,925
Non-trainable params: 0
_________________________________________________________________
```

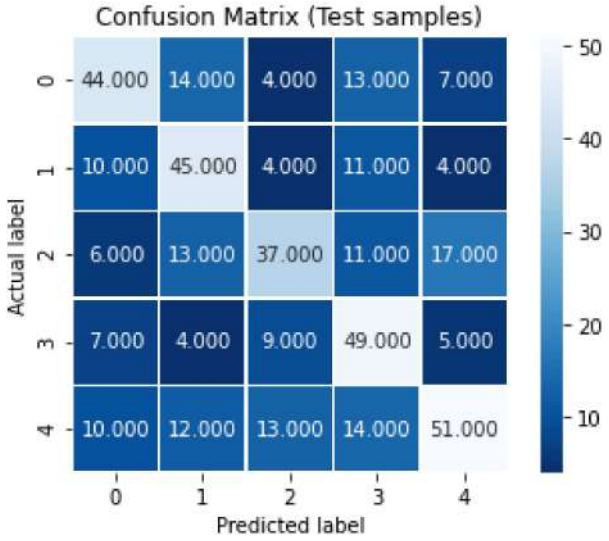
In [ ]:
```
History16 = model16.fit(train,train_labels, batch_size=128, epochs = 50, validation_data = (test,test_labels))
```

In [ ]:
```
draw(History16, model16, train, test, train_labels, test_labels)
```






```
Maximum training accuracy:  0.9943548440933228
Maximum validation accuracy:  0.6111111044883728
```



	precision	recall	f1-score	support
0	0.57	0.54	0.55	82
1	0.51	0.61	0.56	74
2	0.55	0.44	0.49	84
3	0.50	0.66	0.57	74
4	0.61	0.51	0.55	100
accuracy			0.55	414
macro avg	0.55	0.55	0.54	414
weighted avg	0.55	0.55	0.54	414

Model 16 is hence the best model so far with an accuracy of 63.87% on the validation dataset.

```
leakyrelu = tf.keras.layers.LeakyReLU(alpha=0.01)
model16 = Sequential()

model16.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation = leakyrelu, input_shape =
(80,80,1)))
model16.add(MaxPooling2D(pool_size=(2, 2)))
model16.add(Dropout(0.1))

model16.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model16.add(MaxPooling2D(pool_size=(2, 2)))
model16.add(Dropout(0.1))

model16.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model16.add(MaxPooling2D(pool_size=(2, 2)))
model16.add(Dropout(0.1))

model16.add(Conv2D(filters = 128, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model16.add(MaxPooling2D(pool_size=(2, 2)))
model16.add(Dropout(0.1))

model16.add(Conv2D(filters = 256, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model16.add(MaxPooling2D(pool_size=(2, 2)))
model16.add(Dropout(0.1))
```

```
model16.add(Conv2D(filters = 512, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model16.add(MaxPooling2D(pool_size=(2, 2)))
model16.add(Dropout(0.1))

model16.add(Flatten())
model16.add(Dense(128, activation = leakyrelu))
model16.add(Dense(256, activation = leakyrelu))
model16.add(Dense(5, activation = 'softmax'))

model16.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model16.summary()
```

Now, we will use this model and run the experiment on color images.

Load the colour dataset

In [29]:
```
X = []
Z = []
IMG_SIZE = 80
BASE_PATH = "/content/drive/My Drive/ML_DRIVE/Assign_6/flowers_dataset/"
FLOWER_DAISY_DIR     = BASE_PATH + "daisy"
FLOWER_SUNFLOWER_DIR = BASE_PATH + "sunflower"
FLOWER_TULIP_DIR     = BASE_PATH + "tulip"
FLOWER_DANDI_DIR     = BASE_PATH + "dandelion"
FLOWER_ROSE_DIR      = BASE_PATH + "rose"
```

In [ ]:
```
make_train_data('Daisy',FLOWER_DAISY_DIR, 'color')
```

In [ ]:
```
make_train_data('Sunflower',FLOWER_SUNFLOWER_DIR, 'color')
```

In [ ]:
```
make_train_data('Tulip',FLOWER_TULIP_DIR, 'color')
```

In [ ]:
```
make_train_data('Dandelion',FLOWER_DANDI_DIR, 'color')
```

In [ ]:
```
make_train_data('Rose',FLOWER_ROSE_DIR, 'color')
```

In [ ]:
```
def draw_matrix(model, train, test, train_labels, test_labels):
  predictions_test = []
  for item in test:
    predictions_test.append(np.argmax(model.predict(item.reshape(-1,80,80,3))))

  cm_test = sklearn.metrics.confusion_matrix(np.argmax(test_labels, axis = 1), predictions_test)
  sns.heatmap(cm_test, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_r');
  plt.ylabel('Actual label');
  plt.xlabel('Predicted label');
  all_sample_title = 'Confusion Matrix (Test samples)'
  plt.title(all_sample_title)
  plt.show()
  print(classification_report(np.argmax(test_labels, axis = 1), predictions_test))

def draw(History, model, train, test, train_labels, test_labels):
  draw_graph(History)
  draw_matrix(model, train, test, train_labels, test_labels)
```

Pre-processing

In [ ]:
```
le = LabelEncoder()
Y  = le.fit_transform(Z)
Y  = to_categorical(Y,5)
X  = np.array(X)
X  = X/255
```

Train : Test split

In [ ]:
```
train, test, train_labels, test_labels = train_test_split(X, Y, test_size=0.1, random_state=100)
```

Training the model

the model is same as model 16(best) but in that input_shape was (80,80,1) as it was not colour. But since we require colour (80,80,3), hence we are creating the same model with change only in input_shape

In [ ]:
```
leakyrelu = tf.keras.layers.LeakyReLU(alpha=0.01)
model17 = Sequential()
```

```
model17.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation = leakyrelu, input_shape = (80,80,3)))
model17.add(MaxPooling2D(pool_size=(2, 2)))
model17.add(Dropout(0.1))

model17.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model17.add(MaxPooling2D(pool_size=(2, 2)))
model17.add(Dropout(0.1))

model17.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model17.add(MaxPooling2D(pool_size=(2, 2)))
model17.add(Dropout(0.1))

model17.add(Conv2D(filters = 128, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model17.add(MaxPooling2D(pool_size=(2, 2)))
model17.add(Dropout(0.1))

model17.add(Conv2D(filters = 256, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model17.add(MaxPooling2D(pool_size=(2, 2)))
model17.add(Dropout(0.1))

model17.add(Conv2D(filters = 512, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model17.add(MaxPooling2D(pool_size=(2, 2)))
model17.add(Dropout(0.1))

model17.add(Flatten())
model17.add(Dense(128, activation = leakyrelu))
model17.add(Dense(256, activation = leakyrelu))
model17.add(Dense(5, activation = 'softmax'))

model17.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model17.summary()
```

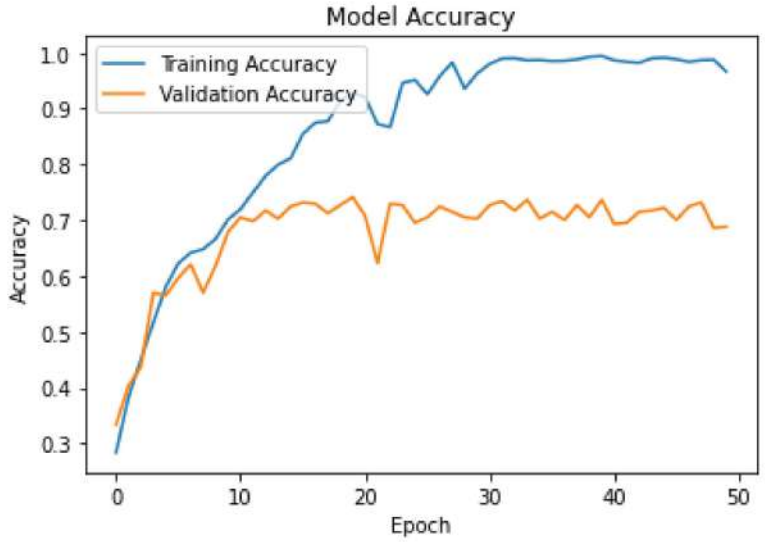
Model: "sequential_16"

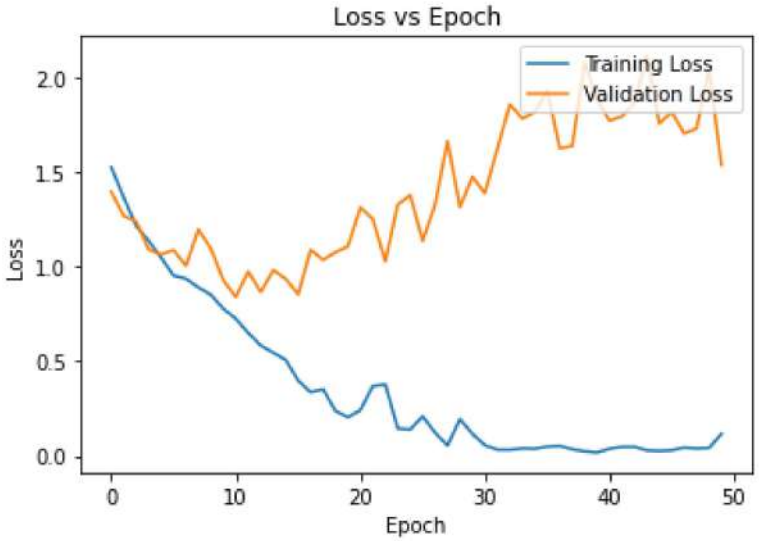
Layer (type)	Output Shape	Param #
conv2d_54 (Conv2D)	(None, 80, 80, 16)	448
max_pooling2d_51 (MaxPooling	(None, 40, 40, 16)	0
dropout_51 (Dropout)	(None, 40, 40, 16)	0
conv2d_55 (Conv2D)	(None, 40, 40, 32)	12832
max_pooling2d_52 (MaxPooling	(None, 20, 20, 32)	0
dropout_52 (Dropout)	(None, 20, 20, 32)	0
conv2d_56 (Conv2D)	(None, 20, 20, 64)	51264
max_pooling2d_53 (MaxPooling	(None, 10, 10, 64)	0
dropout_53 (Dropout)	(None, 10, 10, 64)	0
conv2d_57 (Conv2D)	(None, 10, 10, 128)	204928
max_pooling2d_54 (MaxPooling	(None, 5, 5, 128)	0
dropout_54 (Dropout)	(None, 5, 5, 128)	0
conv2d_58 (Conv2D)	(None, 5, 5, 256)	819456
max_pooling2d_55 (MaxPooling	(None, 2, 2, 256)	0
dropout_55 (Dropout)	(None, 2, 2, 256)	0
conv2d_59 (Conv2D)	(None, 2, 2, 512)	3277312
max_pooling2d_56 (MaxPooling	(None, 1, 1, 512)	0
dropout_56 (Dropout)	(None, 1, 1, 512)	0
flatten_16 (Flatten)	(None, 512)	0
dense_39 (Dense)	(None, 128)	65664

```
dense_40 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_41 (Dense)             (None, 5)                 1285
=================================================================
Total params: 4,466,213
Trainable params: 4,466,213
Non-trainable params: 0
_________________________________________________________________
```

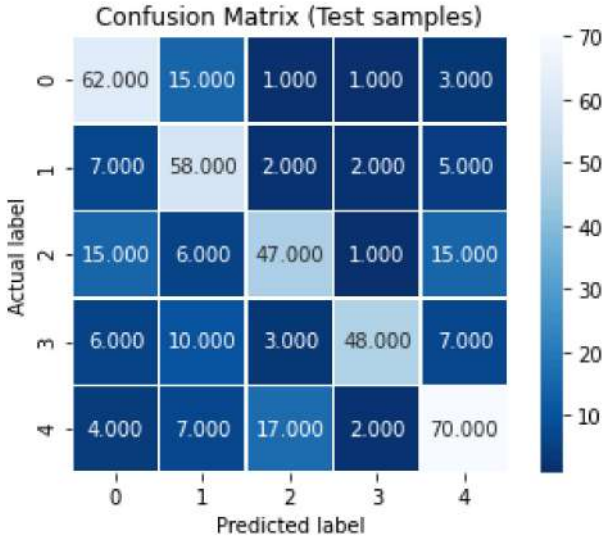
In [ ]:
```
History17 = model17.fit(train, train_labels, batch_size=128, epochs=50, validation_data=(test,test_labels))
```

In [ ]:
```
draw(History17, model17, train, test, train_labels, test_labels)
```





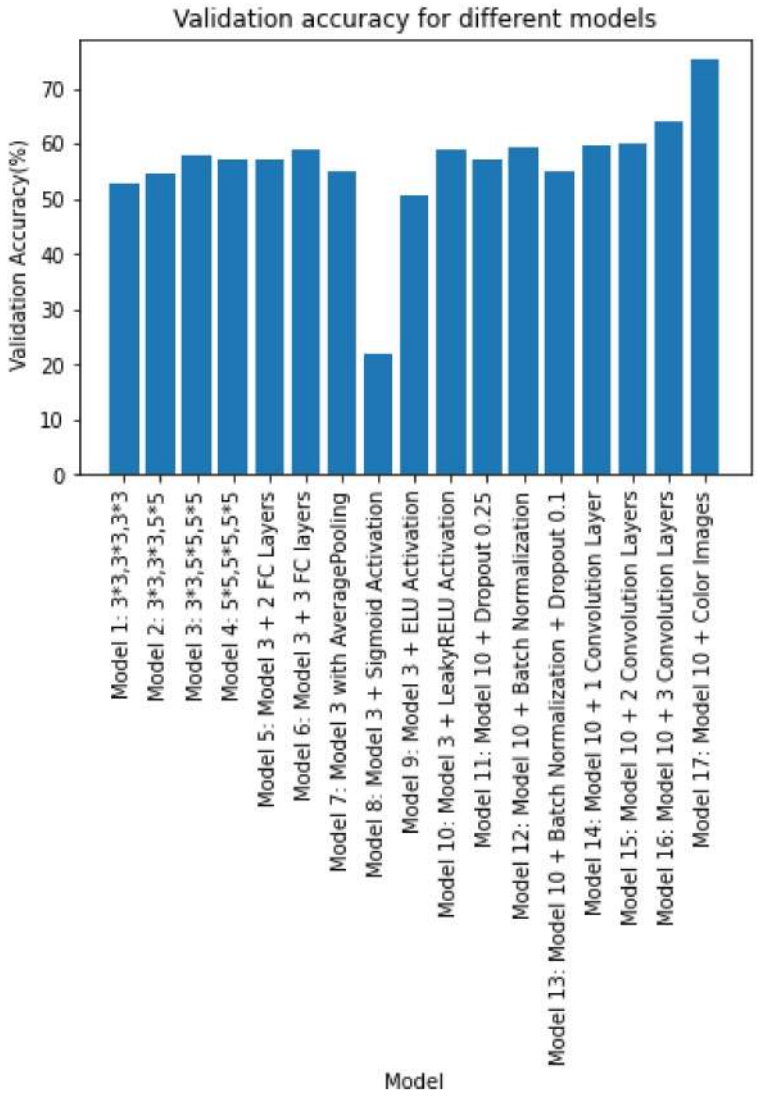

```
Maximum training accuracy:  0.9943548440933228
Maximum validation accuracy:  0.7415459156036377
```



	precision	recall	f1-score	support
0	0.66	0.76	0.70	82
1	0.60	0.78	0.68	74
2	0.67	0.56	0.61	84
3	0.89	0.65	0.75	74
4	0.70	0.70	0.70	100
accuracy			0.69	414
macro avg	0.70	0.69	0.69	414
weighted avg	0.70	0.69	0.69	414

Plot bar-graph for comparing validation accuracy for all the models

In [ ]:
```
validation = [52.94, 54.66, 57.84, 57.10, 57.11, 58.82, 55.14, 22.06, 50.49, 58.81, 57.11, 59.31, 54.90, 59.80, 60.05, (
serial = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]
labels = ['Model 1: 3*3,3*3,3*3', 'Model 2: 3*3,3*3,5*5', 'Model 3: 3*3,5*5,5*5', 'Model 4: 5*5,5*5,5*5', 'Model 5: Mode
          'Model 6: Model 3 + 3 FC layers', 'Model 7: Model 3 with AveragePooling', 'Model 8: Model 3 + Sigmoid Activat
          'Model 9: Model 3 + ELU Activation', 'Model 10: Model 3 + LeakyRELU Activation', 'Model 11: Model 10 + Dropou
          'Model 12: Model 10 + Batch Normalization', 'Model 13: Model 10 + Batch Normalization + Dropout 0.1',
          'Model 14: Model 10 + 1 Convolution Layer', 'Model 15: Model 10 + 2 Convolution Layers', 'Model 16: Model 10 +
          'Model 17: Model 10 + Color Images']

plt.bar(serial,validation)
plt.title("Validation accuracy for different models")
plt.xticks(serial, labels, rotation = 90)
plt.xlabel("Model")
plt.ylabel(" Validation Accuracy(%)")
plt.show()
```



Trying out the best model on MNIST Dataset

Loading Dataset

In [21]:
```
(train, train_labels), (test, test_labels) = tf.keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

In [22]:
```
def draw_matrix(model, train, test, train_labels, test_labels):
  predictions_test = []
  for item in test:
    predictions_test.append(np.argmax(model.predict(item.reshape(-1,28,28,1))))

  cm_test = sklearn.metrics.confusion_matrix(np.argmax(test_labels, axis = 1), predictions_test)
  sns.heatmap(cm_test, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_r');
  plt.ylabel('Actual label');
  plt.xlabel('Predicted label');
  all_sample_title = 'Confusion Matrix (Test samples)'
  plt.title(all_sample_title)
  plt.show()
  print(classification_report(np.argmax(test_labels, axis = 1), predictions_test))

def draw(History, model, train, test, train_labels, test_labels):
  draw_graph(History)
  draw_matrix(model, train, test, train_labels, test_labels)
```

Preparing the images for feeding into CNN

In [23]:
```
train = train.reshape((-1,28,28,1))
test = test.reshape((-1,28,28,1))
```

Categorical (One-hot) Encoding

In [24]:
```
from keras.utils import np_utils
train_labels = np_utils.to_categorical(train_labels, 10)
test_labels = np_utils.to_categorical(test_labels, 10)
```

Model

In [25]:
```
leakyrelu = tf.keras.layers.LeakyReLU(alpha=0.01)
model18 = Sequential()

model18.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation = leakyrelu, input_shape = (28,28,1)))
```

```
model18.add(MaxPooling2D(pool_size=(2, 2)))
model18.add(Dropout(0.1))

model18.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model18.add(MaxPooling2D(pool_size=(2, 2)))
model18.add(Dropout(0.1))

model18.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model18.add(MaxPooling2D(pool_size=(2, 2)))
model18.add(Dropout(0.1))

model18.add(Conv2D(filters = 128, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model18.add(MaxPooling2D(pool_size=(2, 2)))
model18.add(Dropout(0.1))

model18.add(Conv2D(filters = 256, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model18.add(Dropout(0.1))

model18.add(Conv2D(filters = 512, kernel_size = (5,5),padding = 'Same',activation = leakyrelu))
model18.add(Dropout(0.1))

model18.add(Flatten())
model18.add(Dense(128, activation = leakyrelu))
model18.add(Dense(256, activation = leakyrelu))
model18.add(Dense(10, activation = 'softmax'))

model18.compile(optimizer=Adam(),loss='categorical_crossentropy',metrics=['accuracy'])
model18.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 16)        160

 max_pooling2d (MaxPooling2D  (None, 14, 14, 16)       0
 )

 dropout (Dropout)           (None, 14, 14, 16)        0

 conv2d_1 (Conv2D)           (None, 14, 14, 32)        12832

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 32)         0
 2D)

 dropout_1 (Dropout)         (None, 7, 7, 32)          0

 conv2d_2 (Conv2D)           (None, 7, 7, 64)          51264

 max_pooling2d_2 (MaxPooling  (None, 3, 3, 64)         0
 2D)

 dropout_2 (Dropout)         (None, 3, 3, 64)          0

 conv2d_3 (Conv2D)           (None, 3, 3, 128)         204928

 max_pooling2d_3 (MaxPooling  (None, 1, 1, 128)        0
 2D)

 dropout_3 (Dropout)         (None, 1, 1, 128)         0

 conv2d_4 (Conv2D)           (None, 1, 1, 256)         819456

 dropout_4 (Dropout)         (None, 1, 1, 256)         0

 conv2d_5 (Conv2D)           (None, 1, 1, 512)         3277312

 dropout_5 (Dropout)         (None, 1, 1, 512)         0

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 128)               65664
```

```
 dense_1 (Dense)             (None, 256)               33024

 dense_2 (Dense)             (None, 10)                2570

=================================================================
Total params: 4,467,210
Trainable params: 4,467,210
Non-trainable params: 0
_________________________________________________________________
```

In [26]:
```
History18 = model18.fit(train,train_labels, batch_size=128, epochs = 10, validation_split = 0.1)
```

```
Epoch 1/10
422/422 [==============================] - 16s 15ms/step - loss: 0.3383 - accuracy: 0.8872 - val_loss: 0.0690 - val_acc
uracy: 0.9810
Epoch 2/10
422/422 [==============================] - 6s 15ms/step - loss: 0.0909 - accuracy: 0.9754 - val_loss: 0.0508 - val_accu
racy: 0.9862
Epoch 3/10
422/422 [==============================] - 6s 14ms/step - loss: 0.0670 - accuracy: 0.9819 - val_loss: 0.0464 - val_accu
racy: 0.9878
Epoch 4/10
422/422 [==============================] - 7s 16ms/step - loss: 0.0563 - accuracy: 0.9854 - val_loss: 0.0369 - val_accu
racy: 0.9913
Epoch 5/10
422/422 [==============================] - 6s 14ms/step - loss: 0.0514 - accuracy: 0.9865 - val_loss: 0.0449 - val_accu
racy: 0.9890
Epoch 6/10
422/422 [==============================] - 6s 14ms/step - loss: 0.0452 - accuracy: 0.9883 - val_loss: 0.0499 - val_accu
racy: 0.9897
Epoch 7/10
422/422 [==============================] - 6s 14ms/step - loss: 0.0405 - accuracy: 0.9896 - val_loss: 0.0392 - val_accu
racy: 0.9923
Epoch 8/10
422/422 [==============================] - 6s 15ms/step - loss: 0.0396 - accuracy: 0.9895 - val_loss: 0.0451 - val_accu
racy: 0.9902
Epoch 9/10
422/422 [==============================] - 6s 14ms/step - loss: 0.0348 - accuracy: 0.9906 - val_loss: 0.0499 - val_accu
racy: 0.9902
Epoch 10/10
422/422 [==============================] - 6s 14ms/step - loss: 0.0330 - accuracy: 0.9911 - val_loss: 0.0378 - val_accu
racy: 0.9912
```

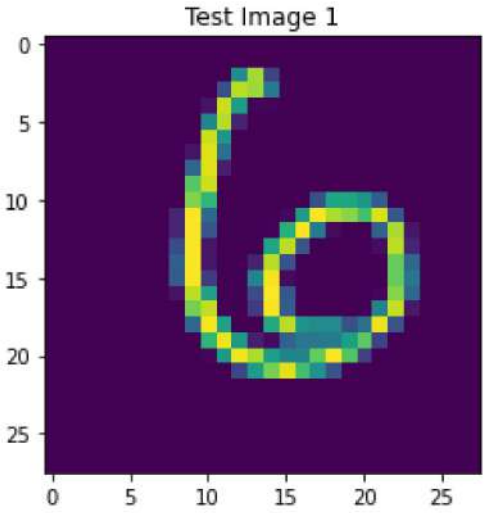
Making predictions on 5 test images

In [27]:
```
results = model18.predict(test)

def test_inference(id, results):
  img = test[id]
  img = img.reshape((28,28))
  plt.imshow(img)
  plt.title("Test Image 1")
  print("True class:", np.argmax(test_labels[id]))
  print("Predicted class:", np.argmax(results[id]))
  plt.show()
```

```
313/313 [==============================] - 2s 4ms/step
```

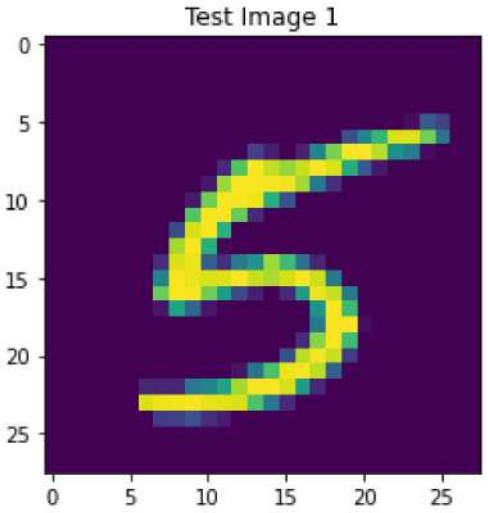
In [28]:
```
test_inference(100, results)
```

```
True class: 6
Predicted class: 6
```



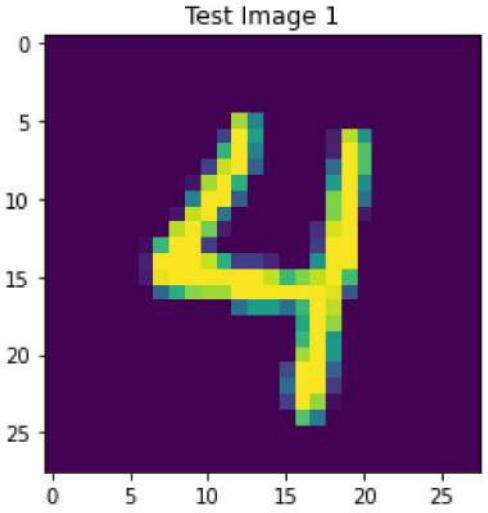
In [29]:
```
test_inference(52, results)
```

```
True class: 5
Predicted class: 5
```



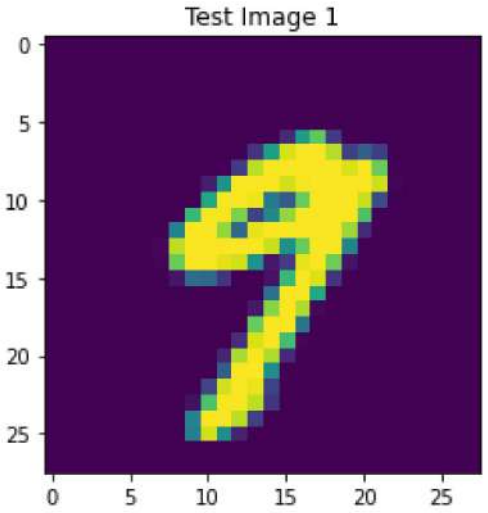
In [30]:
```
test_inference(27, results)
```

```
True class: 4
Predicted class: 4
```



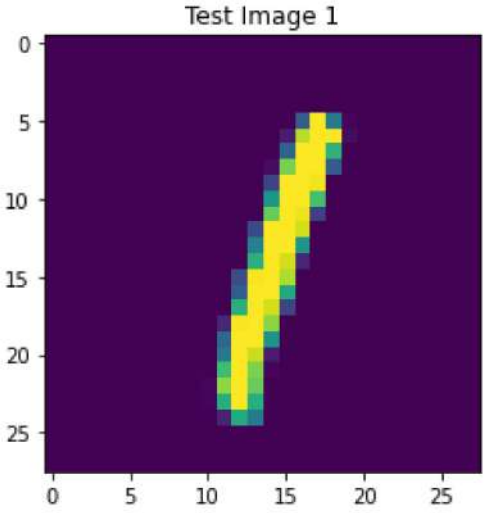
In [31]:
```
test_inference(20, results)
```

```
True class: 9
Predicted class: 9
```



In [32]:
```
test_inference(5, results)
```

```
True class: 1
Predicted class: 1
```



Testing on 5 handwritten images

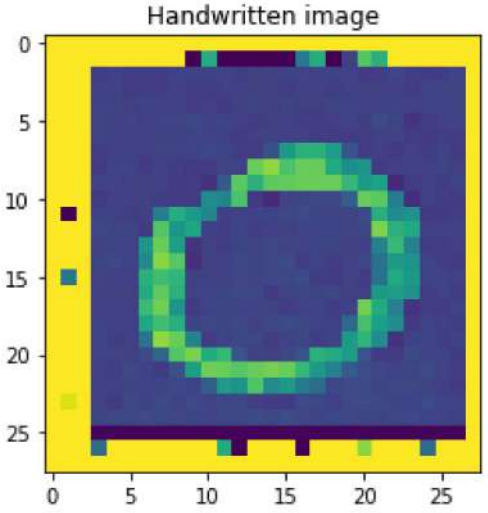
In [100…
```
def load_image(path):
    img = cv2.imread(path)
    return img
```

```
def inference(img):
  img_size = (28,28)
  img = np.array(img)
  img = cv2.resize(img, img_size)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  test_img = img.reshape((-1,28,28,1))
  img_class = np.argmax(model18.predict(test_img), axis=-1)
  prediction = img_class[0]
  classname = img_class[0]
  print("Predicted Class: " + (str)(classname))
  img = img.reshape((28,28))
  plt.imshow(img)
  plt.title("Handwritten image")
  plt.show()
```

In [101…
```
BASE_PATH = "/content/drive/My Drive/ML_DRIVE/Assign_6/"
```

In [104…
```
path = BASE_PATH + "0.png"
img1 = load_image(path)
inference(img1)
```

```
1/1 [==============================] - 0s 15ms/step
Predicted Class: 0
```



In [ ]:
```
path = BASE_PATH + "1.png"
img1 = load_image(path)
```

```
inference(img1)
```

In [ ]:
```
path = BASE_PATH + "2.png"
img1 = load_image(path)
inference(img1)
```

In [ ]:
```
path = BASE_PATH + "3.png"
img1 = load_image(path)
inference(img1)
```

In [ ]:
```
path = BASE_PATH + "6.png"
img1 = load_image(path)
inference(img1)
```

In [ ]:
```

```