Class Challenge: Image Classification of COVID-19 X-rays

Task 2 [Total points: 30]

Setup

- This assignment involves the following packages: 'matplotlib', 'numpy', and 'sklearn'.
- If you are using conda, use the following commands to install the above packages:

```
conda install matplotlib
conda install numpy
conda install -c anaconda scikit-learn
```

- If you are using pip, use use the following commands to install the above packages:

```
pip install matplotlib
pip install numpy
pip install sklearn
```

Data

Please download the data using the following link: COVID-19 (https://drive.google.com/file/d/1Y88tgqpQ1Pjko_7rntcPowOJs_QNOrJ-/view).

- After downloading 'Covid_Data_GradientCrescent.zip', unzip the file and you should see the following data structure:

|--all
|--------train
|--------test
|--two
|--------train
|--------test

- Put the 'all' folder, the 'two' folder and this python notebook in the **same directory** so that the following code can correctly locate the data.

[20 points] Multi-class Classification

Xception Architecture Begins Here

In [11]:
```
import os

import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator

os.environ['OMP_NUM_THREADS'] = '1'
os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
tf.__version__
```

Out[11]: '2.3.0'

Load Image Data

In [12]:
```
DATA_LIST = os.listdir('Covid_Data_GradientCrescent/all/train')
DATASET_PATH  = 'Covid_Data_GradientCrescent/all/train'
TEST_DIR =  'Covid_Data_GradientCrescent/all/test'
IMAGE_SIZE    = (224, 224)
NUM_CLASSES   = len(DATA_LIST)
BATCH_SIZE    = 32  # try reducing batch size or freeze more layers if y
our GPU runs out of memory
NUM_EPOCHS    = 50
LEARNING_RATE = 0.0001 # start off with high rate first 0.001 and experi
ment with reducing it gradually
```

Generate Training and Validation Batches

In [13]:
```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_center = True,
                                   featurewise_std_normalization = True,width_shift_range=0.2,
                                   height_shift_range=0.2,shear_range=0.25,zoom_range=0.1,
                                   zca_whitening = True,channel_shift_range = 20,
                                   horizontal_flip = True,vertical_flip = True,
                                   validation_split = 0.2,fill_mode='constant')


train_batches = train_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                  shuffle=True,batch_size=BATCH_SIZE,
                                                  subset = "training",seed=42,
                                                  class_mode="categorical")

valid_batches = train_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                  shuffle=True,batch_size=BATCH_SIZE,
                                                  subset = "validation",
                                                  seed=42,class_mode="categorical")
```

```
Found 216 images belonging to 4 classes.
Found 54 images belonging to 4 classes.
```

[10 points] Build Model

In [14]:
```
from tensorflow.keras import models, layers, optimizers

xception = tf.keras.applications.Xception(weights="imagenet", include_top=False, input_shape=(224,224,3))

model_xception = tf.keras.models.Sequential([
    xception,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(256, activation="relu", name='dense_feature'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(NUM_CLASSES, activation="softmax")
])

model_xception.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
xception (Functional)        (None, 7, 7, 2048)        20861480
_________________________________________________________________
flatten_1 (Flatten)          (None, 100352)            0
_________________________________________________________________
dropout_2 (Dropout)          (None, 100352)            0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               25690368
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 1028
=================================================================
Total params: 46,552,876
Trainable params: 46,498,348
Non-trainable params: 54,528
_________________________________________________________________
```

[5 points] Train Model

In [15]:
```
#FIT MODEL
print(len(train_batches))
print(len(valid_batches))

STEP_SIZE_TRAIN=train_batches.n//train_batches.batch_size
STEP_SIZE_VALID=valid_batches.n//valid_batches.batch_size

from keras import optimizers

model_xception.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
              optimizer=optimizers.Adam(lr=LEARNING_RATE),
              metrics=['accuracy'])

output = model_xception.fit(train_batches,
                 validation_data = valid_batches,
                 validation_steps = STEP_SIZE_VALID,
                 batch_size = BATCH_SIZE,
                 steps_per_epoch =STEP_SIZE_TRAIN,
                 epochs= NUM_EPOCHS)
```

```
7
2

/Applications/anaconda3/lib/python3.8/site-packages/keras_preprocessin
g/image/image_data_generator.py:720: UserWarning: This ImageDataGenerat
or specifies `featurewise_center`, but it hasn't been fit on any traini
ng data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
/Applications/anaconda3/lib/python3.8/site-packages/keras_preprocessin
g/image/image_data_generator.py:739: UserWarning: This ImageDataGenerat
or specifies `zca_whitening`, but it hasn't been fit on any training da
ta. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
```

```
Epoch 1/50
6/6 [==============================] - 132s 22s/step - loss: 1.6369 - a
ccuracy: 0.2826 - val_loss: 1.7080 - val_accuracy: 0.3438
Epoch 2/50
6/6 [==============================] - 127s 21s/step - loss: 1.1329 - a
ccuracy: 0.5272 - val_loss: 1.2803 - val_accuracy: 0.4688
Epoch 3/50
6/6 [==============================] - 132s 22s/step - loss: 1.1097 - a
ccuracy: 0.5978 - val_loss: 2.2748 - val_accuracy: 0.2188
Epoch 4/50
6/6 [==============================] - 111s 19s/step - loss: 0.8181 - a
ccuracy: 0.6667 - val_loss: 1.4605 - val_accuracy: 0.4375
Epoch 5/50
6/6 [==============================] - 114s 19s/step - loss: 0.7053 - a
ccuracy: 0.6576 - val_loss: 2.0104 - val_accuracy: 0.4062
Epoch 6/50
6/6 [==============================] - 109s 18s/step - loss: 0.6057 - a
ccuracy: 0.7391 - val_loss: 1.6936 - val_accuracy: 0.3750
Epoch 7/50
6/6 [==============================] - 97s 16s/step - loss: 0.5459 - ac
curacy: 0.7880 - val_loss: 2.0396 - val_accuracy: 0.2812
Epoch 8/50
6/6 [==============================] - 104s 17s/step - loss: 0.5382 - a
ccuracy: 0.7708 - val_loss: 0.7253 - val_accuracy: 0.5938
Epoch 9/50
6/6 [==============================] - 79s 13s/step - loss: 0.5297 - ac
curacy: 0.7935 - val_loss: 0.7413 - val_accuracy: 0.5625
Epoch 10/50
6/6 [==============================] - 84s 14s/step - loss: 0.4334 - ac
curacy: 0.8098 - val_loss: 0.9103 - val_accuracy: 0.6250
Epoch 11/50
6/6 [==============================] - 93s 16s/step - loss: 0.4472 - ac
curacy: 0.8152 - val_loss: 0.9750 - val_accuracy: 0.5625
Epoch 12/50
6/6 [==============================] - 88s 15s/step - loss: 0.3528 - ac
curacy: 0.8478 - val_loss: 0.8866 - val_accuracy: 0.5938
Epoch 13/50
6/6 [==============================] - 84s 14s/step - loss: 0.4338 - ac
curacy: 0.8424 - val_loss: 0.5809 - val_accuracy: 0.6875
Epoch 14/50
6/6 [==============================] - 93s 15s/step - loss: 0.2850 - ac
curacy: 0.8696 - val_loss: 0.7478 - val_accuracy: 0.6250
Epoch 15/50
6/6 [==============================] - 96s 16s/step - loss: 0.3855 - ac
curacy: 0.8315 - val_loss: 0.6172 - val_accuracy: 0.7188
Epoch 16/50
6/6 [==============================] - 83s 14s/step - loss: 0.3753 - ac
curacy: 0.8261 - val_loss: 0.5162 - val_accuracy: 0.6250
Epoch 17/50
6/6 [==============================] - 87s 15s/step - loss: 0.3851 - ac
curacy: 0.8696 - val_loss: 0.8151 - val_accuracy: 0.6562
Epoch 18/50
6/6 [==============================] - 86s 14s/step - loss: 0.2917 - ac
curacy: 0.8750 - val_loss: 0.7713 - val_accuracy: 0.5625
Epoch 19/50
6/6 [==============================] - 101s 17s/step - loss: 0.2765 - a
ccuracy: 0.8906 - val_loss: 0.7533 - val_accuracy: 0.6562
```

```
Epoch 20/50
6/6 [==============================] - 86s 14s/step - loss: 0.3141 - ac
curacy: 0.8641 - val_loss: 0.5897 - val_accuracy: 0.7188
Epoch 21/50
6/6 [==============================] - 89s 15s/step - loss: 0.3380 - ac
curacy: 0.8696 - val_loss: 0.5689 - val_accuracy: 0.7500
Epoch 22/50
6/6 [==============================] - 89s 15s/step - loss: 0.2746 - ac
curacy: 0.8859 - val_loss: 0.6452 - val_accuracy: 0.7500
Epoch 23/50
6/6 [==============================] - 86s 14s/step - loss: 0.2848 - ac
curacy: 0.8696 - val_loss: 0.5618 - val_accuracy: 0.6562
Epoch 24/50
6/6 [==============================] - 94s 16s/step - loss: 0.3147 - ac
curacy: 0.8587 - val_loss: 0.8509 - val_accuracy: 0.6562
Epoch 25/50
6/6 [==============================] - 87s 15s/step - loss: 0.2749 - ac
curacy: 0.9010 - val_loss: 0.5800 - val_accuracy: 0.6875
Epoch 26/50
6/6 [==============================] - 100s 17s/step - loss: 0.2102 - a
ccuracy: 0.9076 - val_loss: 0.7614 - val_accuracy: 0.6562
Epoch 27/50
6/6 [==============================] - 98s 16s/step - loss: 0.2881 - ac
curacy: 0.9022 - val_loss: 0.9663 - val_accuracy: 0.7188
Epoch 28/50
6/6 [==============================] - 111s 18s/step - loss: 0.2690 - a
ccuracy: 0.8967 - val_loss: 0.6608 - val_accuracy: 0.6875
Epoch 29/50
6/6 [==============================] - 89s 15s/step - loss: 0.2871 - ac
curacy: 0.8913 - val_loss: 0.3777 - val_accuracy: 0.7500
Epoch 30/50
6/6 [==============================] - 90s 15s/step - loss: 0.2953 - ac
curacy: 0.9185 - val_loss: 0.8372 - val_accuracy: 0.7188
Epoch 31/50
6/6 [==============================] - 97s 16s/step - loss: 0.2328 - ac
curacy: 0.9062 - val_loss: 0.5755 - val_accuracy: 0.7188
Epoch 32/50
6/6 [==============================] - 93s 16s/step - loss: 0.2027 - ac
curacy: 0.8958 - val_loss: 0.9792 - val_accuracy: 0.6875
Epoch 33/50
6/6 [==============================] - 95s 16s/step - loss: 0.1685 - ac
curacy: 0.9293 - val_loss: 0.8338 - val_accuracy: 0.7188
Epoch 34/50
6/6 [==============================] - 94s 16s/step - loss: 0.1690 - ac
curacy: 0.9115 - val_loss: 0.5440 - val_accuracy: 0.7188
Epoch 35/50
6/6 [==============================] - 87s 15s/step - loss: 0.1580 - ac
curacy: 0.9293 - val_loss: 0.7717 - val_accuracy: 0.6875
Epoch 36/50
6/6 [==============================] - 97s 16s/step - loss: 0.1373 - ac
curacy: 0.9511 - val_loss: 0.6642 - val_accuracy: 0.7188
Epoch 37/50
6/6 [==============================] - 92s 15s/step - loss: 0.1456 - ac
curacy: 0.9674 - val_loss: 1.0976 - val_accuracy: 0.6562
Epoch 38/50
6/6 [==============================] - 94s 16s/step - loss: 0.1187 - ac
curacy: 0.9457 - val_loss: 1.2064 - val_accuracy: 0.7188
```

```
Epoch 39/50
6/6 [==============================] - 86s 14s/step - loss: 0.1056 - ac
curacy: 0.9620 - val_loss: 0.7825 - val_accuracy: 0.7812
Epoch 40/50
6/6 [==============================] - 85s 14s/step - loss: 0.1369 - ac
curacy: 0.9511 - val_loss: 0.6103 - val_accuracy: 0.7812
Epoch 41/50
6/6 [==============================] - 81s 14s/step - loss: 0.1503 - ac
curacy: 0.9348 - val_loss: 0.8662 - val_accuracy: 0.7500
Epoch 42/50
6/6 [==============================] - 106s 18s/step - loss: 0.1560 - a
ccuracy: 0.9375 - val_loss: 0.9623 - val_accuracy: 0.5938
Epoch 43/50
6/6 [==============================] - 85s 14s/step - loss: 0.1050 - ac
curacy: 0.9565 - val_loss: 1.1463 - val_accuracy: 0.6875
Epoch 44/50
6/6 [==============================] - 96s 16s/step - loss: 0.1966 - ac
curacy: 0.9130 - val_loss: 1.1675 - val_accuracy: 0.6875
Epoch 45/50
6/6 [==============================] - 91s 15s/step - loss: 0.1221 - ac
curacy: 0.9511 - val_loss: 0.7169 - val_accuracy: 0.7188
Epoch 46/50
6/6 [==============================] - 107s 18s/step - loss: 0.0922 - a
ccuracy: 0.9783 - val_loss: 0.5179 - val_accuracy: 0.7812
Epoch 47/50
6/6 [==============================] - 85s 14s/step - loss: 0.2048 - ac
curacy: 0.9185 - val_loss: 0.9012 - val_accuracy: 0.7188
Epoch 48/50
6/6 [==============================] - 81s 13s/step - loss: 0.0698 - ac
curacy: 0.9728 - val_loss: 0.8724 - val_accuracy: 0.7500
Epoch 49/50
6/6 [==============================] - 80s 13s/step - loss: 0.1118 - ac
curacy: 0.9674 - val_loss: 1.0622 - val_accuracy: 0.6562
Epoch 50/50
6/6 [==============================] - 85s 14s/step - loss: 0.1207 - ac
curacy: 0.9531 - val_loss: 1.2855 - val_accuracy: 0.6562
```

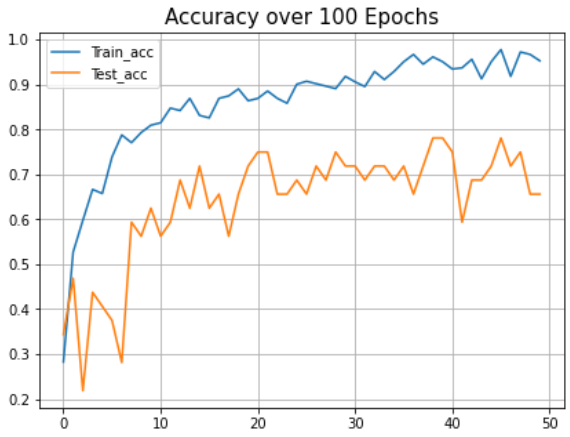
In [16]:
```
model_xception.save('Xception_Multiclass.h5')
```

[5 points] Plot Accuracy and Loss During Training

In [17]:
```
import matplotlib.pyplot as plt

def plot_accuracy_loss_training(output, epochs):
    training_accuracy = output.history['accuracy']
    training_loss = output.history['loss']
    validation_accuracy = output.history['val_accuracy']
    validation_loss = output.history['val_loss']
    plt.figure(figsize=(15, 5))
    plt.subplot(121)
    plt.plot(range(0,NUM_EPOCHS), training_accuracy[:], label='Train_ac
c')
    plt.plot(range(0,NUM_EPOCHS), validation_accuracy[:], label='Test_ac
c')
    plt.title('Accuracy over ' + str(epochs) + ' Epochs', size=15)
    plt.legend()
    plt.grid(True)
    plt.subplot(122)
    plt.plot(range(0,NUM_EPOCHS), training_loss[:], label='Train_loss')
    plt.plot(range(0,NUM_EPOCHS), validation_loss[:], label='Test_loss')
    plt.title('Loss over ' + str(epochs) + ' Epochs', size=15)
    plt.legend()
    plt.grid(True)
    plt.show()

plot_accuracy_loss_training(output, 100)
```



Testing Model

In [18]:
```
test_datagen = ImageDataGenerator(rescale=1. / 255)

eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=I
MAGE_SIZE,
                                                  batch_size=1,shuffle=F
alse,seed=42,class_mode="categorical")
eval_generator.reset()
print(len(eval_generator))
model_xception = keras.models.load_model('Xception_Multiclass.h5')
x = model_xception.evaluate_generator(eval_generator,steps = np.ceil(len
(eval_generator)),
                            use_multiprocessing = False,verbose = 1,worke
rs=1)
print('Test loss:' , x[0])
print('Test accuracy:',x[1])
```

```
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 5s 138ms/step - loss: 1.0454 -
accuracy: 0.7500
Test loss: 1.0453734397888184
Test accuracy: 0.75
```

[10 points] TSNE Plot

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a widely used technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. After training is complete, extract features from a specific deep layer of your choice, use t-SNE to reduce the dimensionality of your extracted features to 2 dimensions and plot the resulting 2D features.

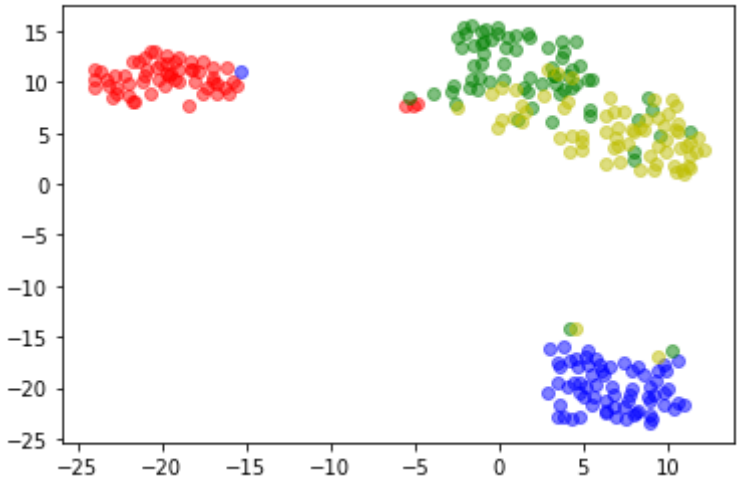
In [20]:
```
from sklearn.manifold import TSNE
import seaborn as sns
import pandas as pd
from tensorflow import keras

intermediate_layer_model = tf.keras.models.Model(inputs=model_xception.input,
                                                 outputs=model_xception.get_layer('dense_feature').output)
tsne_data_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                       batch_size=1,shuffle=False,seed=42,class_mode="binary")

tsne_data_generator.reset()
activations = intermediate_layer_model.predict_generator(tsne_data_generator, verbose=1)
tsne = TSNE(random_state=42, n_components=2).fit_transform(activations)

for index, tsne in enumerate(tsne):
    if tsne_data_generator.labels[index] == 0:
        plt.scatter(tsne[0], tsne[1], color = 'r', alpha=0.5)
    elif tsne_data_generator.labels[index] == 1:
        plt.scatter(tsne[0], tsne[1], color = 'b', alpha=0.5)
    elif tsne_data_generator.labels[index] == 2:
        plt.scatter(tsne[0], tsne[1], color = 'g', alpha=0.5)
    else:
        plt.scatter(tsne[0], tsne[1], color = 'y', alpha=0.5)
plt.show()
```

```
Found 270 images belonging to 4 classes.
WARNING:tensorflow:From <ipython-input-20-870959db74b5>:12: Model.predict_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.predict, which supports generators.
270/270 [==============================] - 44s 162ms/step
```



----x----x----x----x----x----x----x----x----x----x----x----x----x----x----x----x----x----x----x----x----x----x---

Architecture 2 (ResNetV2) Begins Here

In [21]:
```
import os

import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator

os.environ['OMP_NUM_THREADS'] = '1'
os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
tf.__version__
```

Out[21]:
```
'2.3.0'
```

In [22]:
```
DATA_LIST = os.listdir('Covid_Data_GradientCrescent/all/train')
DATASET_PATH  = 'Covid_Data_GradientCrescent/all/train'
TEST_DIR =  'Covid_Data_GradientCrescent/all/test'
IMAGE_SIZE    = (224, 224)
NUM_CLASSES   = len(DATA_LIST)
BATCH_SIZE    = 32  # try reducing batch size or freeze more layers if y
our GPU runs out of memory
NUM_EPOCHS    = 50
LEARNING_RATE = 0.0001 # start off with high rate first 0.001 and experi
ment with reducing it gradually
```

In [23]:
```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_center = True,
                                   featurewise_std_normalization = True,width_shift_range=0.2,
                                   height_shift_range=0.2,shear_range=0.25,zoom_range=0.1,
                                   zca_whitening = True,channel_shift_range = 20,
                                   horizontal_flip = True,vertical_flip = True,
                                   validation_split = 0.2,fill_mode='constant')


train_batches = train_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                  shuffle=True,batch_size=BATCH_SIZE,
                                                  subset = "training",seed=42,
                                                  class_mode="categorical")

valid_batches = train_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                  shuffle=True,batch_size=BATCH_SIZE,
                                                  subset = "validation",
                                                  seed=42,class_mode="categorical")
```

```
Found 216 images belonging to 4 classes.
Found 54 images belonging to 4 classes.

/Applications/anaconda3/lib/python3.8/site-packages/keras_preprocessing/image/image_data_generator.py:342: UserWarning: This ImageDataGenerator specifies `zca_whitening` which overrides setting of`featurewise_std_normalization`.
  warnings.warn('This ImageDataGenerator specifies '
```

In [24]:
```
from tensorflow.keras import models, layers, optimizers

resnet50v2 = tf.keras.applications.ResNet50V2(weights="imagenet", includ
e_top=False, input_shape=(224,224,3))

model_resnet50v2 = tf.keras.models.Sequential([
    resnet50v2,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation="relu", name='dense_feature'),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(NUM_CLASSES, activation="softmax")
])

model_resnet50v2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50v2 (Functional)      (None, 7, 7, 2048)        23564800
_________________________________________________________________
flatten_2 (Flatten)          (None, 100352)            0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               25690368
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 1028
=================================================================
Total params: 49,256,196
Trainable params: 49,210,756
Non-trainable params: 45,440
_________________________________________________________________
```

In [25]:
```
#FIT MODEL
print(len(train_batches))
print(len(valid_batches))

STEP_SIZE_TRAIN=train_batches.n//train_batches.batch_size
STEP_SIZE_VALID=valid_batches.n//valid_batches.batch_size

from keras import optimizers

model_resnet50v2.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
              optimizer=optimizers.Adam(lr=LEARNING_RATE),
              metrics=['accuracy'])

output = model_resnet50v2.fit(train_batches,
                 validation_data = valid_batches,
                 validation_steps = STEP_SIZE_VALID,
                 batch_size = BATCH_SIZE,
                 steps_per_epoch =STEP_SIZE_TRAIN,
                 epochs= NUM_EPOCHS)
```

```
7
2

/Applications/anaconda3/lib/python3.8/site-packages/keras_preprocessin
g/image/image_data_generator.py:720: UserWarning: This ImageDataGenerat
or specifies `featurewise_center`, but it hasn't been fit on any traini
ng data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
/Applications/anaconda3/lib/python3.8/site-packages/keras_preprocessin
g/image/image_data_generator.py:739: UserWarning: This ImageDataGenerat
or specifies `zca_whitening`, but it hasn't been fit on any training da
ta. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
```

```
Epoch 1/50
6/6 [==============================] - 61s 10s/step - loss: 2.6517 - ac
curacy: 0.3315 - val_loss: 2.1742 - val_accuracy: 0.3750
Epoch 2/50
6/6 [==============================] - 60s 10s/step - loss: 1.5166 - ac
curacy: 0.5761 - val_loss: 3.0785 - val_accuracy: 0.3750
Epoch 3/50
6/6 [==============================] - 57s 9s/step - loss: 1.3592 - acc
uracy: 0.5815 - val_loss: 2.0084 - val_accuracy: 0.4688
Epoch 4/50
6/6 [==============================] - 59s 10s/step - loss: 1.0066 - ac
curacy: 0.6630 - val_loss: 2.8739 - val_accuracy: 0.3438
Epoch 5/50
6/6 [==============================] - 57s 9s/step - loss: 0.8916 - acc
uracy: 0.6576 - val_loss: 1.9295 - val_accuracy: 0.3750
Epoch 6/50
6/6 [==============================] - 56s 9s/step - loss: 0.6571 - acc
uracy: 0.7120 - val_loss: 1.8892 - val_accuracy: 0.3438
Epoch 7/50
6/6 [==============================] - 56s 9s/step - loss: 0.6693 - acc
uracy: 0.7337 - val_loss: 1.8156 - val_accuracy: 0.4688
Epoch 8/50
6/6 [==============================] - 63s 11s/step - loss: 0.5914 - ac
curacy: 0.7554 - val_loss: 1.4264 - val_accuracy: 0.4062
Epoch 9/50
6/6 [==============================] - 56s 9s/step - loss: 0.4779 - acc
uracy: 0.7989 - val_loss: 1.3585 - val_accuracy: 0.5000
Epoch 10/50
6/6 [==============================] - 56s 9s/step - loss: 0.5169 - acc
uracy: 0.7663 - val_loss: 1.6259 - val_accuracy: 0.5000
Epoch 11/50
6/6 [==============================] - 56s 9s/step - loss: 0.4795 - acc
uracy: 0.7880 - val_loss: 0.9419 - val_accuracy: 0.6875
Epoch 12/50
6/6 [==============================] - 59s 10s/step - loss: 0.5047 - ac
curacy: 0.7717 - val_loss: 1.4937 - val_accuracy: 0.5312
Epoch 13/50
6/6 [==============================] - 56s 9s/step - loss: 0.6244 - acc
uracy: 0.7554 - val_loss: 0.8166 - val_accuracy: 0.6250
Epoch 14/50
6/6 [==============================] - 56s 9s/step - loss: 0.5021 - acc
uracy: 0.7663 - val_loss: 1.0644 - val_accuracy: 0.5938
Epoch 15/50
6/6 [==============================] - 57s 9s/step - loss: 0.4646 - acc
uracy: 0.7609 - val_loss: 1.4203 - val_accuracy: 0.5312
Epoch 16/50
6/6 [==============================] - 56s 9s/step - loss: 0.5768 - acc
uracy: 0.7989 - val_loss: 1.1689 - val_accuracy: 0.4688
Epoch 17/50
6/6 [==============================] - 61s 10s/step - loss: 0.4512 - ac
curacy: 0.8370 - val_loss: 1.0449 - val_accuracy: 0.5938
Epoch 18/50
6/6 [==============================] - 59s 10s/step - loss: 0.4130 - ac
curacy: 0.8177 - val_loss: 0.9427 - val_accuracy: 0.5312
Epoch 19/50
6/6 [==============================] - 59s 10s/step - loss: 0.3792 - ac
curacy: 0.8424 - val_loss: 1.0455 - val_accuracy: 0.5625
```

```
Epoch 20/50
6/6 [==============================] - 59s 10s/step - loss: 0.4527 - ac
curacy: 0.8043 - val_loss: 0.5960 - val_accuracy: 0.6875
Epoch 21/50
6/6 [==============================] - 58s 10s/step - loss: 0.3437 - ac
curacy: 0.8587 - val_loss: 1.0546 - val_accuracy: 0.6250
Epoch 22/50
6/6 [==============================] - 58s 10s/step - loss: 0.3572 - ac
curacy: 0.8385 - val_loss: 0.9655 - val_accuracy: 0.6562
Epoch 23/50
6/6 [==============================] - 59s 10s/step - loss: 0.4258 - ac
curacy: 0.8333 - val_loss: 0.9572 - val_accuracy: 0.5938
Epoch 24/50
6/6 [==============================] - 56s 9s/step - loss: 0.4221 - acc
uracy: 0.8098 - val_loss: 0.7832 - val_accuracy: 0.7500
Epoch 25/50
6/6 [==============================] - 62s 10s/step - loss: 0.3241 - ac
curacy: 0.8315 - val_loss: 1.0236 - val_accuracy: 0.6562
Epoch 26/50
6/6 [==============================] - 56s 9s/step - loss: 0.3021 - acc
uracy: 0.8859 - val_loss: 0.9236 - val_accuracy: 0.5938
Epoch 27/50
6/6 [==============================] - 59s 10s/step - loss: 0.2863 - ac
curacy: 0.8646 - val_loss: 0.5950 - val_accuracy: 0.7500
Epoch 28/50
6/6 [==============================] - 56s 9s/step - loss: 0.3050 - acc
uracy: 0.8587 - val_loss: 1.1840 - val_accuracy: 0.5312
Epoch 29/50
6/6 [==============================] - 56s 9s/step - loss: 0.3441 - acc
uracy: 0.8533 - val_loss: 0.7849 - val_accuracy: 0.6562
Epoch 30/50
6/6 [==============================] - 56s 9s/step - loss: 0.3668 - acc
uracy: 0.8207 - val_loss: 1.0620 - val_accuracy: 0.6875
Epoch 31/50
6/6 [==============================] - 56s 9s/step - loss: 0.3253 - acc
uracy: 0.8804 - val_loss: 1.3939 - val_accuracy: 0.5312
Epoch 32/50
6/6 [==============================] - 59s 10s/step - loss: 0.4103 - ac
curacy: 0.8281 - val_loss: 1.1655 - val_accuracy: 0.5312
Epoch 33/50
6/6 [==============================] - 57s 9s/step - loss: 0.2916 - acc
uracy: 0.8859 - val_loss: 0.7165 - val_accuracy: 0.7188
Epoch 34/50
6/6 [==============================] - 65s 11s/step - loss: 0.3411 - ac
curacy: 0.8587 - val_loss: 0.9267 - val_accuracy: 0.7188
Epoch 35/50
6/6 [==============================] - 56s 9s/step - loss: 0.2570 - acc
uracy: 0.8913 - val_loss: 1.0970 - val_accuracy: 0.5312
Epoch 36/50
6/6 [==============================] - 56s 9s/step - loss: 0.3236 - acc
uracy: 0.8859 - val_loss: 0.8009 - val_accuracy: 0.6875
Epoch 37/50
6/6 [==============================] - 55s 9s/step - loss: 0.3295 - acc
uracy: 0.8641 - val_loss: 1.1366 - val_accuracy: 0.5938
Epoch 38/50
6/6 [==============================] - 57s 9s/step - loss: 0.2966 - acc
uracy: 0.8424 - val_loss: 1.3536 - val_accuracy: 0.6562
```

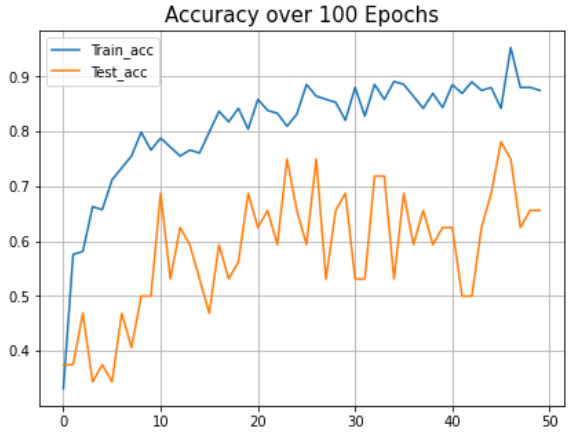
```
Epoch 39/50
6/6 [==============================] - 55s 9s/step - loss: 0.3973 - acc
uracy: 0.8696 - val_loss: 0.8186 - val_accuracy: 0.5938
Epoch 40/50
6/6 [==============================] - 58s 10s/step - loss: 0.3631 - ac
curacy: 0.8438 - val_loss: 0.8418 - val_accuracy: 0.6250
Epoch 41/50
6/6 [==============================] - 58s 10s/step - loss: 0.2584 - ac
curacy: 0.8854 - val_loss: 0.9155 - val_accuracy: 0.6250
Epoch 42/50
6/6 [==============================] - 56s 9s/step - loss: 0.2938 - acc
uracy: 0.8696 - val_loss: 1.4083 - val_accuracy: 0.5000
Epoch 43/50
6/6 [==============================] - 60s 10s/step - loss: 0.2665 - ac
curacy: 0.8906 - val_loss: 1.4082 - val_accuracy: 0.5000
Epoch 44/50
6/6 [==============================] - 57s 10s/step - loss: 0.2459 - ac
curacy: 0.8750 - val_loss: 0.8367 - val_accuracy: 0.6250
Epoch 45/50
6/6 [==============================] - 58s 10s/step - loss: 0.3234 - ac
curacy: 0.8802 - val_loss: 1.1925 - val_accuracy: 0.6875
Epoch 46/50
6/6 [==============================] - 58s 10s/step - loss: 0.3655 - ac
curacy: 0.8424 - val_loss: 0.8013 - val_accuracy: 0.7812
Epoch 47/50
6/6 [==============================] - 61s 10s/step - loss: 0.1563 - ac
curacy: 0.9531 - val_loss: 0.6466 - val_accuracy: 0.7500
Epoch 48/50
6/6 [==============================] - 58s 10s/step - loss: 0.3269 - ac
curacy: 0.8802 - val_loss: 1.2061 - val_accuracy: 0.6250
Epoch 49/50
6/6 [==============================] - 56s 9s/step - loss: 0.2481 - acc
uracy: 0.8804 - val_loss: 0.8723 - val_accuracy: 0.6562
Epoch 50/50
6/6 [==============================] - 56s 9s/step - loss: 0.3419 - acc
uracy: 0.8750 - val_loss: 1.1181 - val_accuracy: 0.6562
```

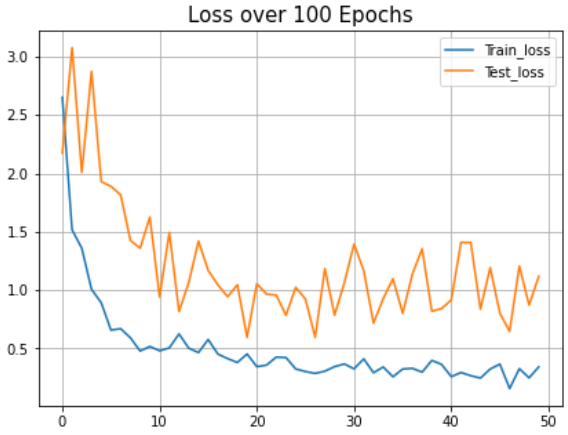
In [26]:
```
model_resnet50v2.save('Resnet50v2_Multiclass.h5')
```

In [27]:
```
import matplotlib.pyplot as plt

def plot_accuracy_loss_training(output, epochs):
    training_accuracy = output.history['accuracy']
    training_loss = output.history['loss']
    validation_accuracy = output.history['val_accuracy']
    validation_loss = output.history['val_loss']
    plt.figure(figsize=(15, 5))
    plt.subplot(121)
    plt.plot(range(0,NUM_EPOCHS), training_accuracy[:], label='Train_ac
c')
    plt.plot(range(0,NUM_EPOCHS), validation_accuracy[:], label='Test_ac
c')
    plt.title('Accuracy over ' + str(epochs) + ' Epochs', size=15)
    plt.legend()
    plt.grid(True)
    plt.subplot(122)
    plt.plot(range(0,NUM_EPOCHS), training_loss[:], label='Train_loss')
    plt.plot(range(0,NUM_EPOCHS), validation_loss[:], label='Test_loss')
    plt.title('Loss over ' + str(epochs) + ' Epochs', size=15)
    plt.legend()
    plt.grid(True)
    plt.show()

plot_accuracy_loss_training(output, 100)
```





In [28]:
```
test_datagen = ImageDataGenerator(rescale=1. / 255)

eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=I
MAGE_SIZE,
                                                  batch_size=1,shuffle=T
rue,seed=42,class_mode="categorical")
eval_generator.reset()
print(len(eval_generator))
model_resnet50v2 = keras.models.load_model('Resnet50v2_Multiclass.h5')
x = model_resnet50v2.evaluate_generator(eval_generator,steps = np.ceil(l
en(eval_generator)),
                            use_multiprocessing = False,verbose = 1,worke
rs=1)
print('Test loss:' , x[0])
print('Test accuracy:',x[1])
```

```
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 4s 115ms/step - loss: 1.2000 -
accuracy: 0.5000
Test loss: 1.2000336647033691
Test accuracy: 0.5
```

In [29]:
```
from sklearn.manifold import TSNE
import seaborn as sns
import pandas as pd
from tensorflow import keras

intermediate_layer_model = tf.keras.models.Model(inputs=model_resnet50v2
.input,
                                                 outputs=model_resnet50v2.get_lay
er('dense_feature').output)
tsne_data_generator = test_datagen.flow_from_directory(DATASET_PATH,targ
et_size=IMAGE_SIZE,
                                                       batch_size=1,shuffle=F
alse,seed=42,class_mode="binary")

tsne_data_generator.reset()
activations = intermediate_layer_model.predict_generator(tsne_data_gener
ator, verbose=1)
tsne = TSNE(random_state=42, n_components=2).fit_transform(activations)

for index, tsne in enumerate(tsne):
    if tsne_data_generator.labels[index] == 0:
        plt.scatter(tsne[0], tsne[1], color = 'r', alpha=0.5)
    elif tsne_data_generator.labels[index] == 1:
        plt.scatter(tsne[0], tsne[1], color = 'b', alpha=0.5)
    elif tsne_data_generator.labels[index] == 2:
        plt.scatter(tsne[0], tsne[1], color = 'g', alpha=0.5)
    else:
        plt.scatter(tsne[0], tsne[1], color = 'y', alpha=0.5)
plt.show()
```

```
Found 270 images belonging to 4 classes.
270/270 [==============================] - 35s 131ms/step
```