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.

- 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

Load Image Data

In [1]:

```
import os

import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
from keras.callbacks import ModelCheckpoint, EarlyStopping

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

Out[1]: '2.8.0'

In [3]:

```
DATA_LIST = os.listdir('all/train')
DATASET_PATH  = 'all/train'
TEST_DIR =  'all/test'
IMAGE_SIZE    = (224, 224)
NUM_CLASSES   = len(DATA_LIST)
BATCH_SIZE    = 10  # try reducing batch size or freeze more layers if your GPU runs out of memory
NUM_EPOCHS    = 100
LEARNING_RATE = 0.0001 # start off with high rate first 0.001 and experiment with reducing it gradually
```

Generate Training and Validation Batches

In [5]:

```
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

Hint: Starting from a pre-trained model typically helps performance on a new task, e.g. starting with weights obtained by training on ImageNet.

In [7]:

```
#Model 1 using Xception pre-trained model -> 0.71-0.75 test acc
base_model = Xception(weights="imagenet", include_top=False, input_shape=train_batches.image_shape)

model = models.Sequential([
    tf.keras.Model(inputs=base_model.input, outputs=base_model.output, name="xception"),
    layers.GlobalAveragePooling2D(),
    layers.Dense(256, activation='relu', name="dense1"),
    layers.Dropout(0.2, name="dropout1"),
    layers.Dense(4, activation="softmax", name="pred_dense")
])

model.get_layer("xception").trainable=False

#summary of model architecture
model.summary()

#compile the model
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE),
    loss=tf.keras.losses.CategoricalCrossentropy(),
    metrics=['accuracy'],
)
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 xception (Functional)       (None, 7, 7, 2048)        20861480

 global_average_pooling2d_1   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense1 (Dense)              (None, 256)               524544

 dropout1 (Dropout)          (None, 256)               0

 pred_dense (Dense)          (None, 4)                 1028

=================================================================
Total params: 21,387,052
Trainable params: 525,572
Non-trainable params: 20,861,480
_________________________________________________________________
```

In [6]:

```
#Model 2 using MobileNet V2 pre-trained model

base_model2 = MobileNetV2(weights="imagenet", include_top=False, input_shape=train_batches.image_shape)

model2 = models.Sequential([
    tf.keras.Model(inputs=base_model2.input, outputs=base_model2.output, name="mobilenet"),
    layers.Flatten(),
    layers.Dense(128, activation='relu', name="dense1"),
    layers.Dropout(0.35, name="dropout1"),
    layers.Dense(4, activation="softmax", name="pred_dense")
])

model2.get_layer("mobilenet").trainable=False

#summary of model architecture
model2.summary()

#compile the model
model2.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE),
    loss=tf.keras.losses.CategoricalCrossentropy(),
    metrics=['accuracy'],
)
```

```
Model: "sequential"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 mobilenet (Functional)      (None, 7, 7, 1280)        2257984   
                                                                 
 flatten (Flatten)           (None, 62720)             0         
                                                                 
 dense1 (Dense)              (None, 128)               8028288   
                                                                 
 dropout1 (Dropout)          (None, 128)               0         
                                                                 
 pred_dense (Dense)          (None, 4)                 516       
                                                                 
=================================================================
Total params: 10,286,788
Trainable params: 8,028,804
Non-trainable params: 2,257,984
_________________________________________________________________
```

[5 points] Train Model

In [ ]:

```
#add in early stopping and model checkpoint
```

In [8]:

```
#FIT MODEL1 - Xception

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

history = model.fit(train_batches, steps_per_epoch=STEP_SIZE_TRAIN, validation_data=valid_batches,
                    validation_steps=STEP_SIZE_VALID, epochs=NUM_EPOCHS)
```

```
22
6
/share/pkg.7/tensorflow/2.8.0/install/lib/SCC/../python3.8/site-packages/keras_preprocessing/image/image_data_generator.p
y:720: UserWarning: This ImageDataGenerator specifies `featurewise_center`, but it hasn't been fit on any training data.
Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
/share/pkg.7/tensorflow/2.8.0/install/lib/SCC/../python3.8/site-packages/keras_preprocessing/image/image_data_generator.p
y:739: UserWarning: This ImageDataGenerator specifies `zca_whitening`, but it hasn't been fit on any training data. Fit i
t first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
Epoch 1/100
```

```
21/21 [==============================] - 65s 3s/step - loss: 1.3526 - accuracy: 0.3398 - val_loss: 1.1421 - val_accuracy: 
0.6600
Epoch 2/100
21/21 [==============================] - 61s 3s/step - loss: 1.1521 - accuracy: 0.5194 - val_loss: 1.0134 - val_accuracy: 
0.5600
Epoch 3/100
21/21 [==============================] - 61s 3s/step - loss: 0.9892 - accuracy: 0.6262 - val_loss: 0.9125 - val_accuracy: 
0.6400
Epoch 4/100
21/21 [==============================] - 62s 3s/step - loss: 0.9117 - accuracy: 0.6359 - val_loss: 0.9360 - val_accuracy: 
0.5800
Epoch 5/100
21/21 [==============================] - 61s 3s/step - loss: 0.8298 - accuracy: 0.6893 - val_loss: 0.8413 - val_accuracy: 
0.6400
Epoch 6/100
21/21 [==============================] - 61s 3s/step - loss: 0.8302 - accuracy: 0.6952 - val_loss: 0.8635 - val_accuracy: 
0.6000
Epoch 7/100
21/21 [==============================] - 61s 3s/step - loss: 0.7421 - accuracy: 0.7282 - val_loss: 0.8219 - val_accuracy: 
0.6200
Epoch 8/100
21/21 [==============================] - 61s 3s/step - loss: 0.8180 - accuracy: 0.6667 - val_loss: 0.7717 - val_accuracy: 
0.6400
Epoch 9/100
21/21 [==============================] - 61s 3s/step - loss: 0.7223 - accuracy: 0.7136 - val_loss: 0.8548 - val_accuracy: 
0.6800
Epoch 10/100
21/21 [==============================] - 61s 3s/step - loss: 0.7074 - accuracy: 0.7330 - val_loss: 0.7446 - val_accuracy: 
0.7000
Epoch 11/100
21/21 [==============================] - 61s 3s/step - loss: 0.7267 - accuracy: 0.7233 - val_loss: 0.6911 - val_accuracy: 
0.6800
Epoch 12/100
21/21 [==============================] - 61s 3s/step - loss: 0.6907 - accuracy: 0.7136 - val_loss: 0.6714 - val_accuracy: 
0.6200
Epoch 13/100
21/21 [==============================] - 61s 3s/step - loss: 0.6666 - accuracy: 0.7379 - val_loss: 0.7875 - val_accuracy: 
0.7000
Epoch 14/100
21/21 [==============================] - 61s 3s/step - loss: 0.6866 - accuracy: 0.6796 - val_loss: 0.6636 - val_accuracy: 
0.7000
Epoch 15/100
21/21 [==============================] - 61s 3s/step - loss: 0.6727 - accuracy: 0.7427 - val_loss: 0.7577 - val_accuracy: 
0.6400
Epoch 16/100
21/21 [==============================] - 61s 3s/step - loss: 0.6424 - accuracy: 0.7524 - val_loss: 0.6571 - val_accuracy: 
0.7200
Epoch 17/100
21/21 [==============================] - 61s 3s/step - loss: 0.6232 - accuracy: 0.7621 - val_loss: 0.7289 - val_accuracy: 
```

```
0.6600
Epoch 18/100
21/21 [==============================] - 61s 3s/step - loss: 0.6550 - accuracy: 0.6990 - val_loss: 0.8193 - val_accuracy:
0.6800
Epoch 19/100
21/21 [==============================] - 61s 3s/step - loss: 0.6322 - accuracy: 0.7379 - val_loss: 0.6521 - val_accuracy:
0.7200
Epoch 20/100
21/21 [==============================] - 61s 3s/step - loss: 0.5856 - accuracy: 0.7621 - val_loss: 0.6879 - val_accuracy:
0.7400
Epoch 21/100
21/21 [==============================] - 60s 3s/step - loss: 0.5983 - accuracy: 0.7718 - val_loss: 0.6075 - val_accuracy:
0.7400
Epoch 22/100
21/21 [==============================] - 61s 3s/step - loss: 0.5550 - accuracy: 0.7913 - val_loss: 0.7843 - val_accuracy:
0.6000
Epoch 23/100
21/21 [==============================] - 61s 3s/step - loss: 0.6556 - accuracy: 0.7233 - val_loss: 0.6613 - val_accuracy:
0.6600
Epoch 24/100
21/21 [==============================] - 61s 3s/step - loss: 0.5942 - accuracy: 0.7816 - val_loss: 0.6673 - val_accuracy:
0.6800
Epoch 25/100
21/21 [==============================] - 62s 3s/step - loss: 0.6042 - accuracy: 0.7524 - val_loss: 0.6686 - val_accuracy:
0.7800
Epoch 26/100
21/21 [==============================] - 61s 3s/step - loss: 0.6057 - accuracy: 0.7816 - val_loss: 0.6486 - val_accuracy:
0.7200
Epoch 27/100
21/21 [==============================] - 61s 3s/step - loss: 0.5447 - accuracy: 0.7864 - val_loss: 0.7335 - val_accuracy:
0.7000
Epoch 28/100
21/21 [==============================] - 61s 3s/step - loss: 0.6110 - accuracy: 0.6990 - val_loss: 0.7353 - val_accuracy:
0.6400
Epoch 29/100
21/21 [==============================] - 61s 3s/step - loss: 0.6508 - accuracy: 0.7524 - val_loss: 0.7340 - val_accuracy:
0.7000
Epoch 30/100
21/21 [==============================] - 60s 3s/step - loss: 0.5687 - accuracy: 0.7621 - val_loss: 0.6032 - val_accuracy:
0.7800
Epoch 31/100
21/21 [==============================] - 60s 3s/step - loss: 0.5228 - accuracy: 0.8058 - val_loss: 0.6893 - val_accuracy:
0.7400
Epoch 32/100
21/21 [==============================] - 61s 3s/step - loss: 0.6124 - accuracy: 0.7573 - val_loss: 0.7036 - val_accuracy:
0.6800
Epoch 33/100
21/21 [==============================] - 61s 3s/step - loss: 0.5942 - accuracy: 0.7379 - val_loss: 0.7043 - val_accuracy:
0.6600
```

```
Epoch 34/100
21/21 [==============================] - 60s 3s/step - loss: 0.5604 - accuracy: 0.7864 - val_loss: 0.6697 - val_accuracy:
0.7600
Epoch 35/100
21/21 [==============================] - 61s 3s/step - loss: 0.6246 - accuracy: 0.7330 - val_loss: 0.7918 - val_accuracy:
0.5800
Epoch 36/100
21/21 [==============================] - 61s 3s/step - loss: 0.5866 - accuracy: 0.7670 - val_loss: 0.6952 - val_accuracy:
0.7200
Epoch 37/100
21/21 [==============================] - 63s 3s/step - loss: 0.5280 - accuracy: 0.8010 - val_loss: 0.6916 - val_accuracy:
0.7200
Epoch 38/100
21/21 [==============================] - 63s 3s/step - loss: 0.5322 - accuracy: 0.7816 - val_loss: 0.6508 - val_accuracy:
0.6600
Epoch 39/100
21/21 [==============================] - 62s 3s/step - loss: 0.5222 - accuracy: 0.7816 - val_loss: 0.6160 - val_accuracy:
0.6600
Epoch 40/100
21/21 [==============================] - 63s 3s/step - loss: 0.5074 - accuracy: 0.7816 - val_loss: 0.8761 - val_accuracy:
0.5400
Epoch 41/100
21/21 [==============================] - 62s 3s/step - loss: 0.4977 - accuracy: 0.8107 - val_loss: 0.6851 - val_accuracy:
0.7000
Epoch 42/100
21/21 [==============================] - 62s 3s/step - loss: 0.4911 - accuracy: 0.7961 - val_loss: 0.7635 - val_accuracy:
0.6000
Epoch 43/100
21/21 [==============================] - 62s 3s/step - loss: 0.5363 - accuracy: 0.7767 - val_loss: 0.5931 - val_accuracy:
0.7400
Epoch 44/100
21/21 [==============================] - 62s 3s/step - loss: 0.5064 - accuracy: 0.7913 - val_loss: 0.5641 - val_accuracy:
0.7400
Epoch 45/100
21/21 [==============================] - 62s 3s/step - loss: 0.5226 - accuracy: 0.8058 - val_loss: 0.6501 - val_accuracy:
0.7200
Epoch 46/100
21/21 [==============================] - 64s 3s/step - loss: 0.4910 - accuracy: 0.7767 - val_loss: 0.7196 - val_accuracy:
0.7000
Epoch 47/100
21/21 [==============================] - 62s 3s/step - loss: 0.5630 - accuracy: 0.7427 - val_loss: 0.6744 - val_accuracy:
0.7600
Epoch 48/100
21/21 [==============================] - 62s 3s/step - loss: 0.4872 - accuracy: 0.7864 - val_loss: 0.7298 - val_accuracy:
0.6600
Epoch 49/100
21/21 [==============================] - 62s 3s/step - loss: 0.5342 - accuracy: 0.8058 - val_loss: 0.6986 - val_accuracy:
0.6800
Epoch 50/100
```

```
21/21 [==============================] - 62s 3s/step - loss: 0.5684 - accuracy: 0.7864 - val_loss: 0.7218 - val_accuracy:
0.6600
Epoch 51/100
21/21 [==============================] - 62s 3s/step - loss: 0.5600 - accuracy: 0.7718 - val_loss: 0.8378 - val_accuracy:
0.6400
Epoch 52/100
21/21 [==============================] - 62s 3s/step - loss: 0.5621 - accuracy: 0.7767 - val_loss: 0.7358 - val_accuracy:
0.7200
Epoch 53/100
21/21 [==============================] - 62s 3s/step - loss: 0.5310 - accuracy: 0.8000 - val_loss: 0.7367 - val_accuracy:
0.6400
Epoch 54/100
21/21 [==============================] - 61s 3s/step - loss: 0.4649 - accuracy: 0.8058 - val_loss: 0.7826 - val_accuracy:
0.6400
Epoch 55/100
21/21 [==============================] - 62s 3s/step - loss: 0.5424 - accuracy: 0.7816 - val_loss: 0.6756 - val_accuracy:
0.7000
Epoch 56/100
21/21 [==============================] - 62s 3s/step - loss: 0.4390 - accuracy: 0.8447 - val_loss: 0.7326 - val_accuracy:
0.6400
Epoch 57/100
21/21 [==============================] - 62s 3s/step - loss: 0.4992 - accuracy: 0.8010 - val_loss: 0.7567 - val_accuracy:
0.6400
Epoch 58/100
21/21 [==============================] - 62s 3s/step - loss: 0.5020 - accuracy: 0.8107 - val_loss: 0.7688 - val_accuracy:
0.6200
Epoch 59/100
21/21 [==============================] - 62s 3s/step - loss: 0.5322 - accuracy: 0.7913 - val_loss: 0.7087 - val_accuracy:
0.7000
Epoch 60/100
21/21 [==============================] - 62s 3s/step - loss: 0.4967 - accuracy: 0.7864 - val_loss: 0.7129 - val_accuracy:
0.7400
Epoch 61/100
21/21 [==============================] - 64s 3s/step - loss: 0.5179 - accuracy: 0.8048 - val_loss: 0.7799 - val_accuracy:
0.6800
Epoch 62/100
21/21 [==============================] - 61s 3s/step - loss: 0.4746 - accuracy: 0.8107 - val_loss: 0.4889 - val_accuracy:
0.7600
Epoch 63/100
21/21 [==============================] - 62s 3s/step - loss: 0.4484 - accuracy: 0.8155 - val_loss: 0.7426 - val_accuracy:
0.6800
Epoch 64/100
21/21 [==============================] - 61s 3s/step - loss: 0.4519 - accuracy: 0.7913 - val_loss: 0.7112 - val_accuracy:
0.6800
Epoch 65/100
21/21 [==============================] - 62s 3s/step - loss: 0.4640 - accuracy: 0.8107 - val_loss: 0.9041 - val_accuracy:
0.6400
Epoch 66/100
21/21 [==============================] - 63s 3s/step - loss: 0.4816 - accuracy: 0.8252 - val_loss: 0.6225 - val_accuracy:
```

```
0.6600
Epoch 67/100
21/21 [==============================] - 63s 3s/step - loss: 0.4952 - accuracy: 0.7913 - val_loss: 0.6824 - val_accuracy:
0.6600
Epoch 68/100
21/21 [==============================] - 62s 3s/step - loss: 0.3717 - accuracy: 0.8689 - val_loss: 0.5875 - val_accuracy:
0.7000
Epoch 69/100
21/21 [==============================] - 63s 3s/step - loss: 0.4931 - accuracy: 0.8107 - val_loss: 0.7239 - val_accuracy:
0.6600
Epoch 70/100
21/21 [==============================] - 62s 3s/step - loss: 0.4262 - accuracy: 0.8350 - val_loss: 0.6446 - val_accuracy:
0.7000
Epoch 71/100
21/21 [==============================] - 63s 3s/step - loss: 0.5286 - accuracy: 0.8107 - val_loss: 0.7738 - val_accuracy:
0.6200
Epoch 72/100
21/21 [==============================] - 62s 3s/step - loss: 0.4550 - accuracy: 0.8333 - val_loss: 0.7328 - val_accuracy:
0.7000
Epoch 73/100
21/21 [==============================] - 63s 3s/step - loss: 0.4541 - accuracy: 0.8107 - val_loss: 0.5657 - val_accuracy:
0.8000
Epoch 74/100
21/21 [==============================] - 63s 3s/step - loss: 0.5446 - accuracy: 0.7816 - val_loss: 0.5738 - val_accuracy:
0.7000
Epoch 75/100
21/21 [==============================] - 64s 3s/step - loss: 0.4689 - accuracy: 0.8252 - val_loss: 0.6138 - val_accuracy:
0.6600
Epoch 76/100
21/21 [==============================] - 62s 3s/step - loss: 0.4393 - accuracy: 0.8301 - val_loss: 0.6886 - val_accuracy:
0.7000
Epoch 77/100
21/21 [==============================] - 63s 3s/step - loss: 0.4631 - accuracy: 0.8107 - val_loss: 0.6436 - val_accuracy:
0.7000
Epoch 78/100
21/21 [==============================] - 62s 3s/step - loss: 0.4148 - accuracy: 0.8398 - val_loss: 0.6817 - val_accuracy:
0.6600
Epoch 79/100
21/21 [==============================] - 62s 3s/step - loss: 0.4778 - accuracy: 0.8333 - val_loss: 0.7577 - val_accuracy:
0.6800
Epoch 80/100
21/21 [==============================] - 63s 3s/step - loss: 0.4908 - accuracy: 0.7913 - val_loss: 0.6060 - val_accuracy:
0.7600
Epoch 81/100
21/21 [==============================] - 63s 3s/step - loss: 0.4357 - accuracy: 0.8252 - val_loss: 0.6610 - val_accuracy:
0.6600
Epoch 82/100
21/21 [==============================] - 62s 3s/step - loss: 0.4726 - accuracy: 0.8107 - val_loss: 0.5754 - val_accuracy:
0.7000
```

```
Epoch 83/100
21/21 [==============================] - 62s 3s/step - loss: 0.4734 - accuracy: 0.8155 - val_loss: 0.6404 - val_accuracy:
0.7000
Epoch 84/100
21/21 [==============================] - 62s 3s/step - loss: 0.4815 - accuracy: 0.7961 - val_loss: 0.7276 - val_accuracy:
0.6800
Epoch 85/100
21/21 [==============================] - 63s 3s/step - loss: 0.4086 - accuracy: 0.8398 - val_loss: 0.6204 - val_accuracy:
0.6800
Epoch 86/100
21/21 [==============================] - 63s 3s/step - loss: 0.4585 - accuracy: 0.8107 - val_loss: 0.6530 - val_accuracy:
0.6800
Epoch 87/100
21/21 [==============================] - 62s 3s/step - loss: 0.3987 - accuracy: 0.8495 - val_loss: 0.6715 - val_accuracy:
0.6800
Epoch 88/100
21/21 [==============================] - 63s 3s/step - loss: 0.4535 - accuracy: 0.8155 - val_loss: 0.7324 - val_accuracy:
0.6600
Epoch 89/100
21/21 [==============================] - 62s 3s/step - loss: 0.3982 - accuracy: 0.8010 - val_loss: 0.6595 - val_accuracy:
0.7200
Epoch 90/100
21/21 [==============================] - 62s 3s/step - loss: 0.4041 - accuracy: 0.8301 - val_loss: 0.5718 - val_accuracy:
0.6800
Epoch 91/100
21/21 [==============================] - 63s 3s/step - loss: 0.4674 - accuracy: 0.8107 - val_loss: 0.7876 - val_accuracy:
0.7000
Epoch 92/100
21/21 [==============================] - 62s 3s/step - loss: 0.4593 - accuracy: 0.8058 - val_loss: 0.6309 - val_accuracy:
0.7000
Epoch 93/100
21/21 [==============================] - 62s 3s/step - loss: 0.4462 - accuracy: 0.8107 - val_loss: 0.7142 - val_accuracy:
0.7200
Epoch 94/100
21/21 [==============================] - 62s 3s/step - loss: 0.4437 - accuracy: 0.8107 - val_loss: 0.7192 - val_accuracy:
0.6800
Epoch 95/100
21/21 [==============================] - 62s 3s/step - loss: 0.4481 - accuracy: 0.8447 - val_loss: 0.6550 - val_accuracy:
0.7000
Epoch 96/100
21/21 [==============================] - 61s 3s/step - loss: 0.5105 - accuracy: 0.8058 - val_loss: 0.7199 - val_accuracy:
0.6800
Epoch 97/100
21/21 [==============================] - 61s 3s/step - loss: 0.4073 - accuracy: 0.8447 - val_loss: 0.6854 - val_accuracy:
0.6600
Epoch 98/100
21/21 [==============================] - 63s 3s/step - loss: 0.3949 - accuracy: 0.8301 - val_loss: 0.6817 - val_accuracy:
0.7200
Epoch 99/100
```

```
21/21 [==============================] - 62s 3s/step - loss: 0.5036 - accuracy: 0.8058 - val_loss: 0.5611 - val_accuracy:
0.8000
Epoch 100/100
21/21 [==============================] - 63s 3s/step - loss: 0.4692 - accuracy: 0.7961 - val_loss: 0.6078 - val_accuracy:
0.7000
```

In [7]:

```
#FIT MODEL2 - MobileNet V2
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

NUM_EPOCHS=85

history2 = model2.fit(train_batches, steps_per_epoch=STEP_SIZE_TRAIN, validation_data=valid_batches,
                      validation_steps=STEP_SIZE_VALID, epochs=NUM_EPOCHS)
```

```
22
6
/share/pkg.7/tensorflow/2.8.0/install/lib/SCC/../python3.8/site-packages/keras_preprocessing/image/image_data_generator.p
y:720: UserWarning: This ImageDataGenerator specifies `featurewise_center`, but it hasn't been fit on any training data.
Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
/share/pkg.7/tensorflow/2.8.0/install/lib/SCC/../python3.8/site-packages/keras_preprocessing/image/image_data_generator.p
y:739: UserWarning: This ImageDataGenerator specifies `zca_whitening`, but it hasn't been fit on any training data. Fit i
t first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
Epoch 1/85
21/21 [==============================] - 10s 365ms/step - loss: 2.3634 - accuracy: 0.3204 - val_loss: 1.1026 - val_accura
cy: 0.5000
Epoch 2/85
21/21 [==============================] - 7s 316ms/step - loss: 1.4160 - accuracy: 0.4757 - val_loss: 1.1339 - val_accurac
y: 0.5000
Epoch 3/85
21/21 [==============================] - 7s 308ms/step - loss: 1.0484 - accuracy: 0.5728 - val_loss: 1.0165 - val_accurac
y: 0.5400
Epoch 4/85
21/21 [==============================] - 7s 309ms/step - loss: 0.9338 - accuracy: 0.5728 - val_loss: 0.8498 - val_accurac
y: 0.5600
Epoch 5/85
21/21 [==============================] - 7s 308ms/step - loss: 0.9067 - accuracy: 0.6262 - val_loss: 0.7902 - val_accurac
y: 0.7000
Epoch 6/85
21/21 [==============================] - 7s 308ms/step - loss: 0.9766 - accuracy: 0.6048 - val_loss: 0.7602 - val_accurac
y: 0.6600
Epoch 7/85
```

```
21/21 [==============================] - 7s 307ms/step - loss: 0.8574 - accuracy: 0.6117 - val_loss: 0.9741 - val_accurac
y: 0.5600
Epoch 8/85
21/21 [==============================] - 7s 308ms/step - loss: 0.7274 - accuracy: 0.6893 - val_loss: 0.6717 - val_accurac
y: 0.6800
Epoch 9/85
21/21 [==============================] - 6s 302ms/step - loss: 0.7429 - accuracy: 0.6650 - val_loss: 0.7631 - val_accurac
y: 0.6400
Epoch 10/85
21/21 [==============================] - 7s 305ms/step - loss: 0.8323 - accuracy: 0.6165 - val_loss: 0.6165 - val_accurac
y: 0.7000
Epoch 11/85
21/21 [==============================] - 7s 305ms/step - loss: 0.6612 - accuracy: 0.7136 - val_loss: 0.6489 - val_accurac
y: 0.7000
Epoch 12/85
21/21 [==============================] - 7s 309ms/step - loss: 0.8623 - accuracy: 0.6214 - val_loss: 0.6625 - val_accurac
y: 0.7000
Epoch 13/85
21/21 [==============================] - 6s 304ms/step - loss: 0.7511 - accuracy: 0.7039 - val_loss: 0.7487 - val_accurac
y: 0.6600
Epoch 14/85
21/21 [==============================] - 7s 319ms/step - loss: 0.7189 - accuracy: 0.7039 - val_loss: 0.6840 - val_accurac
y: 0.6400
Epoch 15/85
21/21 [==============================] - 6s 302ms/step - loss: 0.6588 - accuracy: 0.7039 - val_loss: 0.5924 - val_accurac
y: 0.6800
Epoch 16/85
21/21 [==============================] - 7s 306ms/step - loss: 0.6578 - accuracy: 0.7184 - val_loss: 0.7401 - val_accurac
y: 0.6600
Epoch 17/85
21/21 [==============================] - 6s 299ms/step - loss: 0.7129 - accuracy: 0.7330 - val_loss: 0.6827 - val_accurac
y: 0.6600
Epoch 18/85
21/21 [==============================] - 7s 307ms/step - loss: 0.6509 - accuracy: 0.7330 - val_loss: 0.7907 - val_accurac
y: 0.7200
Epoch 19/85
21/21 [==============================] - 7s 301ms/step - loss: 0.6610 - accuracy: 0.7427 - val_loss: 0.7616 - val_accurac
y: 0.7400
Epoch 20/85
21/21 [==============================] - 7s 298ms/step - loss: 0.6176 - accuracy: 0.7573 - val_loss: 0.7331 - val_accurac
y: 0.6600
Epoch 21/85
21/21 [==============================] - 6s 294ms/step - loss: 0.7350 - accuracy: 0.6845 - val_loss: 0.5877 - val_accurac
y: 0.7800
Epoch 22/85
21/21 [==============================] - 6s 300ms/step - loss: 0.6032 - accuracy: 0.7379 - val_loss: 0.7586 - val_accurac
y: 0.6200
Epoch 23/85
21/21 [==============================] - 6s 299ms/step - loss: 0.6448 - accuracy: 0.7427 - val_loss: 0.5922 - val_accurac
```

```
y: 0.6800
Epoch 24/85
21/21 [==============================] - 6s 304ms/step - loss: 0.5874 - accuracy: 0.7670 - val_loss: 0.6742 - val_accurac
y: 0.7000
Epoch 25/85
21/21 [==============================] - 7s 306ms/step - loss: 0.6517 - accuracy: 0.6942 - val_loss: 0.7059 - val_accurac
y: 0.6800
Epoch 26/85
21/21 [==============================] - 7s 303ms/step - loss: 0.6305 - accuracy: 0.7524 - val_loss: 0.6063 - val_accurac
y: 0.6600
Epoch 27/85
21/21 [==============================] - 6s 305ms/step - loss: 0.6145 - accuracy: 0.7087 - val_loss: 0.7288 - val_accurac
y: 0.6200
Epoch 28/85
21/21 [==============================] - 6s 307ms/step - loss: 0.5566 - accuracy: 0.7816 - val_loss: 0.5973 - val_accurac
y: 0.7400
Epoch 29/85
21/21 [==============================] - 7s 304ms/step - loss: 0.6100 - accuracy: 0.7379 - val_loss: 0.4926 - val_accurac
y: 0.7800
Epoch 30/85
21/21 [==============================] - 6s 298ms/step - loss: 0.6032 - accuracy: 0.7184 - val_loss: 0.5960 - val_accurac
y: 0.6800
Epoch 31/85
21/21 [==============================] - 7s 305ms/step - loss: 0.5921 - accuracy: 0.7573 - val_loss: 0.6551 - val_accurac
y: 0.6600
Epoch 32/85
21/21 [==============================] - 6s 305ms/step - loss: 0.5165 - accuracy: 0.7476 - val_loss: 0.6845 - val_accurac
y: 0.6600
Epoch 33/85
21/21 [==============================] - 7s 315ms/step - loss: 0.4734 - accuracy: 0.8058 - val_loss: 0.7960 - val_accurac
y: 0.6200
Epoch 34/85
21/21 [==============================] - 7s 305ms/step - loss: 0.5406 - accuracy: 0.7864 - val_loss: 0.6726 - val_accurac
y: 0.6400
Epoch 35/85
21/21 [==============================] - 6s 304ms/step - loss: 0.5950 - accuracy: 0.7718 - val_loss: 0.6446 - val_accurac
y: 0.7000
Epoch 36/85
21/21 [==============================] - 6s 292ms/step - loss: 0.5625 - accuracy: 0.7573 - val_loss: 0.4894 - val_accurac
y: 0.6800
Epoch 37/85
21/21 [==============================] - 6s 302ms/step - loss: 0.5994 - accuracy: 0.7573 - val_loss: 0.6936 - val_accurac
y: 0.7000
Epoch 38/85
21/21 [==============================] - 6s 305ms/step - loss: 0.5778 - accuracy: 0.7816 - val_loss: 0.6554 - val_accurac
y: 0.6800
Epoch 39/85
21/21 [==============================] - 6s 301ms/step - loss: 0.5135 - accuracy: 0.8058 - val_loss: 0.6277 - val_accurac
y: 0.7000
```

```
Epoch 40/85
21/21 [==============================] - 6s 301ms/step - loss: 0.5796 - accuracy: 0.7621 - val_loss: 0.7803 - val_accurac
y: 0.5800
Epoch 41/85
21/21 [==============================] - 6s 303ms/step - loss: 0.4766 - accuracy: 0.8010 - val_loss: 0.6655 - val_accurac
y: 0.6600
Epoch 42/85
21/21 [==============================] - 6s 300ms/step - loss: 0.5617 - accuracy: 0.8107 - val_loss: 0.6964 - val_accurac
y: 0.6200
Epoch 43/85
21/21 [==============================] - 7s 308ms/step - loss: 0.5128 - accuracy: 0.7816 - val_loss: 0.4918 - val_accurac
y: 0.7600
Epoch 44/85
21/21 [==============================] - 7s 296ms/step - loss: 0.5793 - accuracy: 0.7767 - val_loss: 0.6160 - val_accurac
y: 0.7000
Epoch 45/85
21/21 [==============================] - 7s 308ms/step - loss: 0.4738 - accuracy: 0.8107 - val_loss: 0.6720 - val_accurac
y: 0.7200
Epoch 46/85
21/21 [==============================] - 7s 303ms/step - loss: 0.4414 - accuracy: 0.8155 - val_loss: 0.6609 - val_accurac
y: 0.7200
Epoch 47/85
21/21 [==============================] - 6s 297ms/step - loss: 0.4564 - accuracy: 0.8058 - val_loss: 0.7807 - val_accurac
y: 0.6400
Epoch 48/85
21/21 [==============================] - 7s 311ms/step - loss: 0.5845 - accuracy: 0.7718 - val_loss: 0.6363 - val_accurac
y: 0.7400
Epoch 49/85
21/21 [==============================] - 6s 304ms/step - loss: 0.5100 - accuracy: 0.7670 - val_loss: 0.5365 - val_accurac
y: 0.7400
Epoch 50/85
21/21 [==============================] - 7s 307ms/step - loss: 0.4662 - accuracy: 0.8204 - val_loss: 0.6745 - val_accurac
y: 0.6800
Epoch 51/85
21/21 [==============================] - 6s 301ms/step - loss: 0.4212 - accuracy: 0.8155 - val_loss: 0.6799 - val_accurac
y: 0.7000
Epoch 52/85
21/21 [==============================] - 6s 298ms/step - loss: 0.4754 - accuracy: 0.7961 - val_loss: 0.5898 - val_accurac
y: 0.7200
Epoch 53/85
21/21 [==============================] - 7s 308ms/step - loss: 0.4551 - accuracy: 0.7913 - val_loss: 0.7522 - val_accurac
y: 0.6800
Epoch 54/85
21/21 [==============================] - 6s 299ms/step - loss: 0.3787 - accuracy: 0.8155 - val_loss: 0.6327 - val_accurac
y: 0.6400
Epoch 55/85
21/21 [==============================] - 7s 310ms/step - loss: 0.6574 - accuracy: 0.7573 - val_loss: 0.6646 - val_accurac
y: 0.7200
Epoch 56/85
```

```
21/21 [==============================] - 6s 303ms/step - loss: 0.5172 - accuracy: 0.7864 - val_loss: 0.5486 - val_accurac
y: 0.7400
Epoch 57/85
21/21 [==============================] - 6s 299ms/step - loss: 0.4721 - accuracy: 0.7913 - val_loss: 0.5730 - val_accurac
y: 0.6200
Epoch 58/85
21/21 [==============================] - 6s 304ms/step - loss: 0.4614 - accuracy: 0.8286 - val_loss: 0.5730 - val_accurac
y: 0.7000
Epoch 59/85
21/21 [==============================] - 6s 295ms/step - loss: 0.4230 - accuracy: 0.8155 - val_loss: 0.6952 - val_accurac
y: 0.7000
Epoch 60/85
21/21 [==============================] - 6s 295ms/step - loss: 0.4917 - accuracy: 0.7816 - val_loss: 0.6543 - val_accurac
y: 0.7600
Epoch 61/85
21/21 [==============================] - 6s 304ms/step - loss: 0.5154 - accuracy: 0.7864 - val_loss: 0.6671 - val_accurac
y: 0.6600
Epoch 62/85
21/21 [==============================] - 6s 303ms/step - loss: 0.4848 - accuracy: 0.7718 - val_loss: 0.5780 - val_accurac
y: 0.7000
Epoch 63/85
21/21 [==============================] - 6s 296ms/step - loss: 0.5009 - accuracy: 0.8058 - val_loss: 0.8596 - val_accurac
y: 0.6800
Epoch 64/85
21/21 [==============================] - 7s 296ms/step - loss: 0.4953 - accuracy: 0.7961 - val_loss: 0.5970 - val_accurac
y: 0.6400
Epoch 65/85
21/21 [==============================] - 7s 311ms/step - loss: 0.4830 - accuracy: 0.7961 - val_loss: 0.6104 - val_accurac
y: 0.7400
Epoch 66/85
21/21 [==============================] - 7s 307ms/step - loss: 0.4547 - accuracy: 0.8010 - val_loss: 0.6281 - val_accurac
y: 0.7200
Epoch 67/85
21/21 [==============================] - 6s 301ms/step - loss: 0.4907 - accuracy: 0.8010 - val_loss: 0.6958 - val_accurac
y: 0.6400
Epoch 68/85
21/21 [==============================] - 7s 303ms/step - loss: 0.5088 - accuracy: 0.7621 - val_loss: 0.5237 - val_accurac
y: 0.7400
Epoch 69/85
21/21 [==============================] - 7s 309ms/step - loss: 0.5625 - accuracy: 0.7913 - val_loss: 0.6556 - val_accurac
y: 0.6600
Epoch 70/85
21/21 [==============================] - 7s 310ms/step - loss: 0.4485 - accuracy: 0.7961 - val_loss: 0.5324 - val_accurac
y: 0.7000
Epoch 71/85
21/21 [==============================] - 6s 303ms/step - loss: 0.4623 - accuracy: 0.8107 - val_loss: 0.5399 - val_accurac
y: 0.7400
Epoch 72/85
21/21 [==============================] - 6s 312ms/step - loss: 0.4863 - accuracy: 0.7961 - val_loss: 0.6547 - val_accurac
```

```
y: 0.6600
Epoch 73/85
21/21 [==============================] - 6s 297ms/step - loss: 0.4846 - accuracy: 0.8107 - val_loss: 0.4284 - val_accurac
y: 0.7800
Epoch 74/85
21/21 [==============================] - 6s 300ms/step - loss: 0.4253 - accuracy: 0.7864 - val_loss: 0.6919 - val_accurac
y: 0.6800
Epoch 75/85
21/21 [==============================] - 6s 296ms/step - loss: 0.4270 - accuracy: 0.8252 - val_loss: 0.6943 - val_accurac
y: 0.6800
Epoch 76/85
21/21 [==============================] - 6s 299ms/step - loss: 0.5281 - accuracy: 0.7913 - val_loss: 0.6511 - val_accurac
y: 0.7200
Epoch 77/85
21/21 [==============================] - 7s 307ms/step - loss: 0.4645 - accuracy: 0.8447 - val_loss: 0.5591 - val_accurac
y: 0.7800
Epoch 78/85
21/21 [==============================] - 6s 297ms/step - loss: 0.4625 - accuracy: 0.8204 - val_loss: 0.5118 - val_accurac
y: 0.7400
Epoch 79/85
21/21 [==============================] - 6s 300ms/step - loss: 0.4804 - accuracy: 0.8058 - val_loss: 0.7075 - val_accurac
y: 0.7200
Epoch 80/85
21/21 [==============================] - 6s 302ms/step - loss: 0.4646 - accuracy: 0.7864 - val_loss: 0.6573 - val_accurac
y: 0.7000
Epoch 81/85
21/21 [==============================] - 7s 306ms/step - loss: 0.5263 - accuracy: 0.7864 - val_loss: 0.6306 - val_accurac
y: 0.7000
Epoch 82/85
21/21 [==============================] - 6s 299ms/step - loss: 0.5177 - accuracy: 0.7670 - val_loss: 0.7737 - val_accurac
y: 0.7000
Epoch 83/85
21/21 [==============================] - 7s 305ms/step - loss: 0.4247 - accuracy: 0.8204 - val_loss: 0.5353 - val_accurac
y: 0.7800
Epoch 84/85
21/21 [==============================] - 7s 312ms/step - loss: 0.5542 - accuracy: 0.7718 - val_loss: 0.7064 - val_accurac
y: 0.7000
Epoch 85/85
21/21 [==============================] - 6s 300ms/step - loss: 0.4208 - accuracy: 0.7961 - val_loss: 0.5657 - val_accurac
y: 0.7600
```

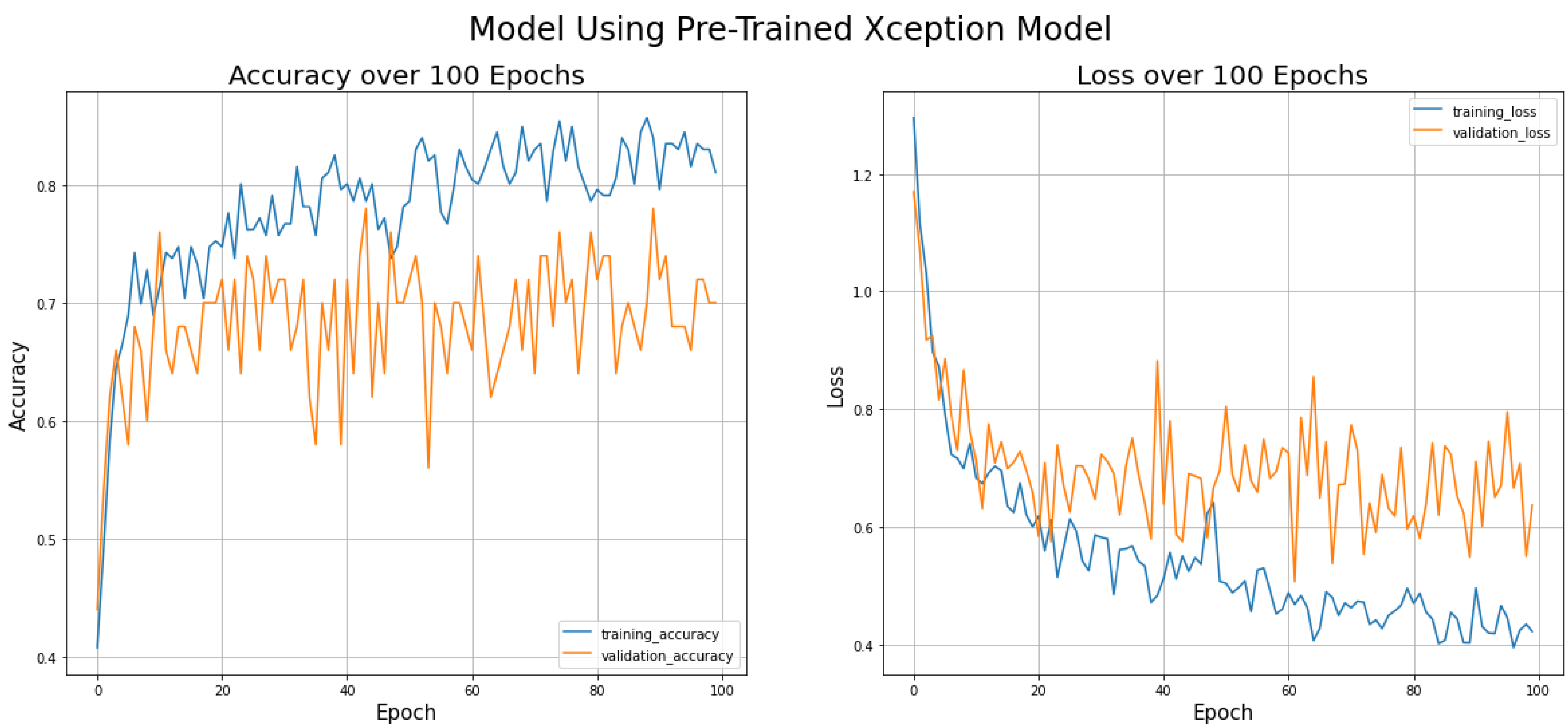
[5 points] Plot Accuracy and Loss During Training

In [10]:

```python
plt.figure(figsize=(20,8))
plt.suptitle("Model Using Pre-Trained Xception Model", fontsize=24)

#plot the accuracies for the training and validation sets
plt.subplot(1, 2, 1)
```

```python
plt.plot(history.history['accuracy'], label='training_accuracy')
plt.plot(history.history['val_accuracy'], label = 'validation_accuracy')
plt.title('Accuracy over %s Epochs' % NUM_EPOCHS, fontsize=20)
plt.xlabel('Epoch', fontsize=15)
plt.ylabel('Accuracy', fontsize=15)
plt.grid()
plt.legend(loc='lower right')

#plot the loss for the training and validation sets
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='training_loss')
plt.plot(history.history['val_loss'], label = 'validation_loss')
plt.title('Loss over %s Epochs' % NUM_EPOCHS, fontsize=20)
plt.xlabel('Epoch', fontsize=15)
plt.ylabel('Loss', fontsize=15)
plt.grid()
plt.legend(loc='upper right')
```

Out[10]: <matplotlib.legend.Legend at 0x2b9ac7d62670>

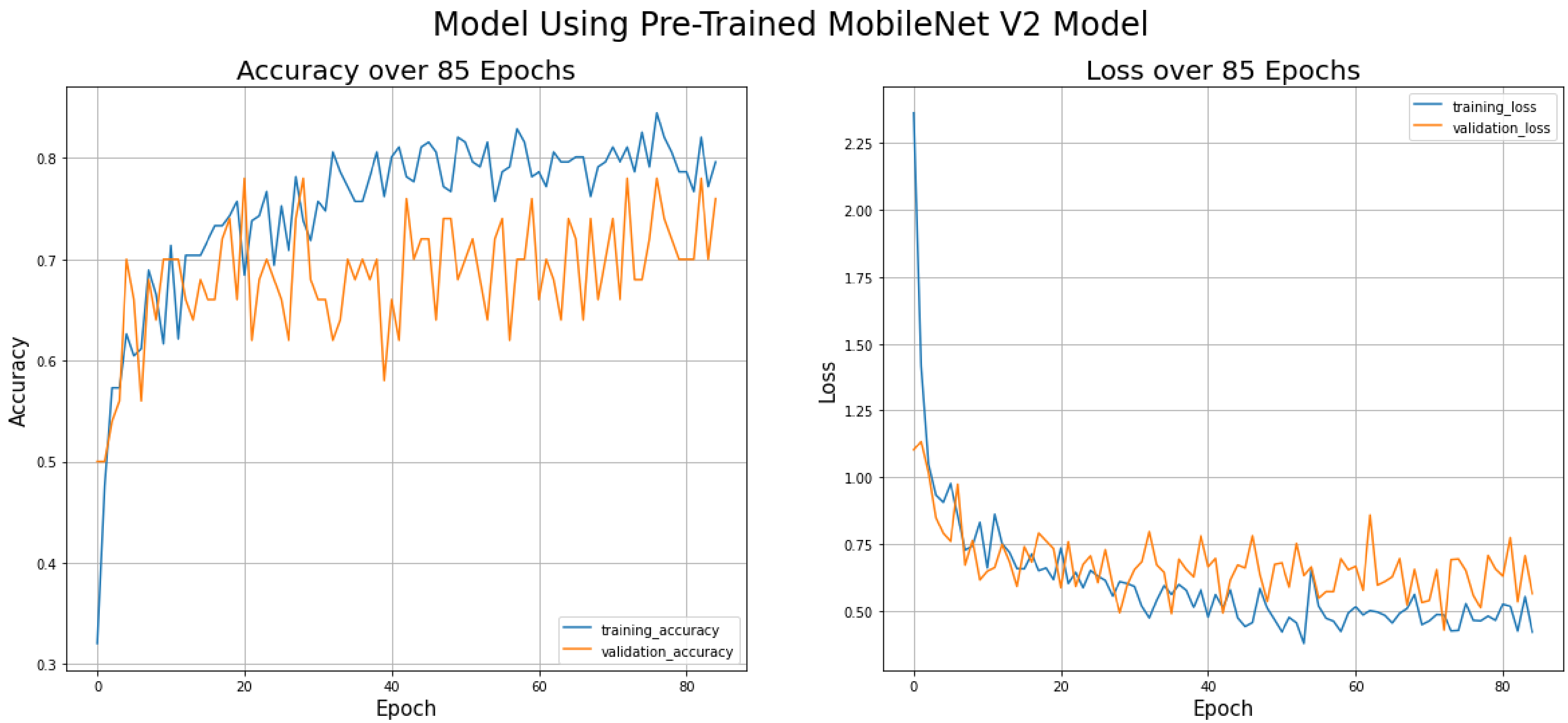


In [8]:

```
plt.figure(figsize=(20,8))
plt.suptitle("Model Using Pre-Trained MobileNet V2 Model", fontsize=24)

#plot the accuracies for the training and validation sets
plt.subplot(1, 2, 1)
plt.plot(history2.history['accuracy'], label='training_accuracy')
plt.plot(history2.history['val_accuracy'], label = 'validation_accuracy')
plt.title('Accuracy over %s Epochs' % NUM_EPOCHS, fontsize=20)
plt.xlabel('Epoch', fontsize=15)
plt.ylabel('Accuracy', fontsize=15)
plt.grid()
plt.legend(loc='lower right')

#plot the loss for the training and validation sets
plt.subplot(1, 2, 2)
plt.plot(history2.history['loss'], label='training_loss')
plt.plot(history2.history['val_loss'], label = 'validation_loss')
plt.title('Loss over %s Epochs' % NUM_EPOCHS, fontsize=20)
plt.xlabel('Epoch', fontsize=15)
plt.ylabel('Loss', fontsize=15)
plt.grid()
plt.legend(loc='upper right')
```

Out[8]:

```
<matplotlib.legend.Legend at 0x2b927739ab80>
```



Testing Model

In [12]:

```
#Accuracy for Model 1 - Xception Pre-Trained Model
print("Model 1: Xception Pre-Trained Model")
test_datagen = ImageDataGenerator(rescale=1. / 255)

eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=IMAGE_SIZE,
                                                  batch_size=1,shuffle=True,seed=42,class_mode="categorical")
eval_generator.reset()
print(len(eval_generator))
x = model.evaluate(eval_generator,steps = np.ceil(len(eval_generator)),
                   use_multiprocessing = False,verbose = 1,workers=1)
print('Test loss:' , x[0])
print('Test accuracy:',x[1])
```

```
Model 1: Xception Pre-Trained Model
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 4s 111ms/step - loss: 0.5150 - accuracy: 0.7778
```

```
Test loss: 0.514977753162384
Test accuracy: 0.7777777910232544
```

In [10]:

```python
#Accuracy for Model 2 - MobileNet V2 Pre-Trained Model
print("Model 2: MobileNet V2 Pre-Trained Model")
test_datagen = ImageDataGenerator(rescale=1. / 255)

eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=IMAGE_SIZE,
                                                  batch_size=1,shuffle=True,seed=42,class_mode="categorical")
eval_generator.reset()
print(len(eval_generator))
x = model2.evaluate(eval_generator,steps = np.ceil(len(eval_generator)),
                    use_multiprocessing = False,verbose = 1,workers=1)
print('Test loss:' , x[0])
print('Test accuracy:',x[1])
```

```
Model 2: MobileNet V2 Pre-Trained Model
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 1s 26ms/step - loss: 0.8866 - accuracy: 0.6389
Test loss: 0.8865880966186523
Test accuracy: 0.6388888955116272
```

[10 points] TSNE Plot

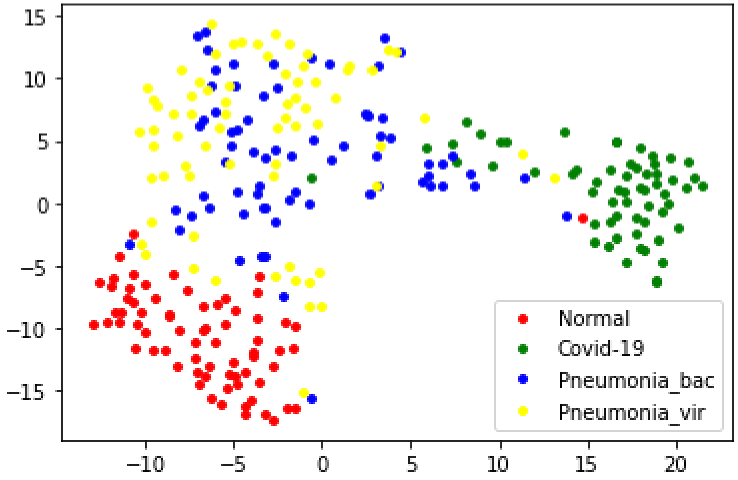
t-Distributed Stochastic Neighbor Embedding (t-SNE) is a widely used technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. After training is complete, extract features from a specific deep layer of your choice, use t-SNE to reduce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

In [13]:

```python
#Model 1 - Xception
from sklearn.manifold import TSNE

intermediate_layer_model = models.Model(inputs=model.input,
                                        outputs=model.get_layer('dense1').output)

tsne_eval_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                  batch_size=1,shuffle=False,seed=42,class_mode="categorical")

# raise NotImplementedError("Extract features from the tsne_data_generator and fit a t-SNE model for the features,"
#                           "and plot the resulting 2D features of the four classes.")

features = intermediate_layer_model.predict(tsne_eval_generator)
```

```
label = tsne_eval_generator.class_indices

classes = tsne_eval_generator.classes
fea_tsne = TSNE(learning_rate=50).fit_transform(features)
X,Y = zip(*fea_tsne)

X_Nor=[]
Y_Nor=[]

X_Cov=[]
Y_Cov=[]

X_Bac=[]
Y_Bac=[]

X_Vir=[]
Y_Vir=[]

for x,y,c in zip(X,Y,classes):
    if(label['covid']==c):
        X_Cov.append(x)
        Y_Cov.append(y)
    elif(label['normal']==c):
        X_Nor.append(x)
        Y_Nor.append(y)
    elif(label['pneumonia_bac']==c):
        X_Bac.append(x)
        Y_Bac.append(y)
    else:
        X_Vir.append(x)
        Y_Vir.append(y)

plt.scatter(X_Nor, Y_Nor, c='red', label='Normal', s=15)
plt.scatter(X_Cov, Y_Cov, c='green', label='Covid-19', s=15)
plt.scatter(X_Bac, Y_Bac, c='blue', label='Pneumonia_bac', s=15)
plt.scatter(X_Vir, Y_Vir, c='yellow', label='Pneumonia_vir', s=15)

plt.legend()
plt.show()
```

```
Found 270 images belonging to 4 classes.
```



In [11]:

```
#Model 2 - MobileNet V2
from sklearn.manifold import TSNE

intermediate_layer_model = models.Model(inputs=model2.input,
                                        outputs=model2.get_layer('dense1').output)

tsne_eval_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                       batch_size=1,shuffle=False,seed=42,class_mode="categorical")

# raise NotImplementedError("Extract features from the tsne_data_generator and fit a t-SNE model for the features,"
#                           "and plot the resulting 2D features of the four classes.")

features = intermediate_layer_model.predict(tsne_eval_generator)
label = tsne_eval_generator.class_indices

classes = tsne_eval_generator.classes
fea_tsne = TSNE(learning_rate=50).fit_transform(features)
X,Y = zip(*fea_tsne)

X_Nor=[]
Y_Nor=[]

X_Cov=[]
Y_Cov=[]

X_Bac=[]
Y_Bac=[]
```

```
X_Vir=[]
Y_Vir=[]

for x,y,c in zip(X,Y,classes):
    if(label['covid']==c):
        X_Cov.append(x)
        Y_Cov.append(y)
    elif(label['normal']==c):
        X_Nor.append(x)
        Y_Nor.append(y)
    elif(label['pneumonia_bac']==c):
        X_Bac.append(x)
        Y_Bac.append(y)
    else:
        X_Vir.append(x)
        Y_Vir.append(y)

plt.scatter(X_Nor, Y_Nor, c='red', label='Normal', s=15)
plt.scatter(X_Cov, Y_Cov, c='green', label='Covid-19', s=15)
plt.scatter(X_Bac, Y_Bac, c='blue', label='Pneumonia_bac', s=15)
plt.scatter(X_Vir, Y_Vir, c='yellow', label='Pneumonia_vir', s=15)

plt.legend()
plt.show()
```

```
Found 270 images belonging to 4 classes.
```