task2_template

December 3, 2020

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

2 Task 2 [Total points: 30]

2.1 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
```

2.2 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.

2.3 [20 points] Multi-class Classification

[13]:
```
import os

import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import InceptionV3, VGG16
```

```
import warnings
warnings.filterwarnings('ignore')

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

[13]: '2.0.0'

Load Image Data

[2]:
```
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

[3]:
```
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.
```

2.4 Model 1 - InceptionV3

[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.

[4]:
```
inception_V3 = InceptionV3(weights = 'imagenet', include_top = False, \
              input_shape = (224,224,3), classes = NUM_CLASSES)

for layer in inception_V3 .layers:
    if "BatchNormalization" in layer.__class__.__name__:
        layer.trainable = True

model = tf.keras.models.Sequential()

model.add(inception_V3)

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dropout(.5))

model.add(tf.keras.layers.Dense(256, activation = 'relu'))

model.add(tf.keras.layers.Dropout(.5))

model.add(tf.keras.layers.Dense(4, activation = 'softmax'))

model.summary()
```

```
Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Model)         (None, 5, 5, 2048)        21802784
_________________________________________________________________
flatten (Flatten)            (None, 51200)             0
_________________________________________________________________
dropout (Dropout)            (None, 51200)             0
_________________________________________________________________
dense (Dense)                (None, 256)               13107456
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
```

```
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 1028
=================================================================
Total params: 34,911,268
Trainable params: 34,876,836
Non-trainable params: 34,432
_________________________________________________________________
```

[5 points] Train Model

[5]:
```
#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

OPTIMIZER = tf.keras.optimizers.Adam(learning_rate = LEARNING_RATE)

model.compile(optimizer = OPTIMIZER, \
                loss=tf.keras.losses.CategoricalCrossentropy(), \
                metrics=['accuracy'])


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

```
22
6
Train for 21 steps, validate for 5 steps
Epoch 1/100
21/21 [==============================] - 110s 5s/step - loss: 2.5969 - accuracy:
0.3107 - val_loss: 1.8354 - val_accuracy: 0.3000
Epoch 2/100
21/21 [==============================] - 87s 4s/step - loss: 1.7381 - accuracy:
0.3495 - val_loss: 1.7264 - val_accuracy: 0.4600
Epoch 3/100
21/21 [==============================] - 92s 4s/step - loss: 1.3877 - accuracy:
0.4175 - val_loss: 1.6633 - val_accuracy: 0.2600
Epoch 4/100
21/21 [==============================] - 89s 4s/step - loss: 1.1198 - accuracy:
0.5049 - val_loss: 1.2339 - val_accuracy: 0.5200
Epoch 5/100
21/21 [==============================] - 86s 4s/step - loss: 1.0680 - accuracy:
0.5291 - val_loss: 0.8587 - val_accuracy: 0.6400
Epoch 6/100
21/21 [==============================] - 88s 4s/step - loss: 1.0551 - accuracy:
```

```
0.5680 - val_loss: 0.9382 - val_accuracy: 0.6400
Epoch 7/100
21/21 [==============================] - 89s 4s/step - loss: 0.8846 - accuracy:
0.6505 - val_loss: 0.8075 - val_accuracy: 0.6600
Epoch 8/100
21/21 [==============================] - 87s 4s/step - loss: 0.7842 - accuracy:
0.6893 - val_loss: 1.1124 - val_accuracy: 0.5400
Epoch 9/100
21/21 [==============================] - 90s 4s/step - loss: 0.7043 - accuracy:
0.6893 - val_loss: 0.9652 - val_accuracy: 0.5400
Epoch 10/100
21/21 [==============================] - 88s 4s/step - loss: 0.6951 - accuracy:
0.6845 - val_loss: 0.7848 - val_accuracy: 0.6600
Epoch 11/100
21/21 [==============================] - 86s 4s/step - loss: 0.7480 - accuracy:
0.6942 - val_loss: 0.7255 - val_accuracy: 0.7200
Epoch 12/100
21/21 [==============================] - 86s 4s/step - loss: 0.6912 - accuracy:
0.6699 - val_loss: 0.7683 - val_accuracy: 0.6400
Epoch 13/100
21/21 [==============================] - 83s 4s/step - loss: 0.6389 - accuracy:
0.7136 - val_loss: 0.6122 - val_accuracy: 0.6800
Epoch 14/100
21/21 [==============================] - 81s 4s/step - loss: 0.5519 - accuracy:
0.7233 - val_loss: 0.5855 - val_accuracy: 0.7200
Epoch 15/100
21/21 [==============================] - 84s 4s/step - loss: 0.6912 - accuracy:
0.6893 - val_loss: 0.6278 - val_accuracy: 0.6800
Epoch 16/100
21/21 [==============================] - 88s 4s/step - loss: 0.6462 - accuracy:
0.7282 - val_loss: 0.9123 - val_accuracy: 0.6000
Epoch 17/100
21/21 [==============================] - 81s 4s/step - loss: 0.6318 - accuracy:
0.7330 - val_loss: 1.1805 - val_accuracy: 0.5800
Epoch 18/100
21/21 [==============================] - 82s 4s/step - loss: 0.4871 - accuracy:
0.7913 - val_loss: 0.5881 - val_accuracy: 0.7200
Epoch 19/100
21/21 [==============================] - 81s 4s/step - loss: 0.5241 - accuracy:
0.7670 - val_loss: 0.5504 - val_accuracy: 0.7600
Epoch 20/100
21/21 [==============================] - 83s 4s/step - loss: 0.5212 - accuracy:
0.7816 - val_loss: 0.4078 - val_accuracy: 0.8400
Epoch 21/100
21/21 [==============================] - 83s 4s/step - loss: 0.4955 - accuracy:
0.7476 - val_loss: 0.4067 - val_accuracy: 0.8000
Epoch 22/100
21/21 [==============================] - 83s 4s/step - loss: 0.5598 - accuracy:
```

```
0.7767 - val_loss: 0.4074 - val_accuracy: 0.8200
Epoch 23/100
21/21 [==============================] - 79s 4s/step - loss: 0.5403 - accuracy:
0.7913 - val_loss: 0.4498 - val_accuracy: 0.7600
Epoch 24/100
21/21 [==============================] - 81s 4s/step - loss: 0.4767 - accuracy:
0.7767 - val_loss: 0.4756 - val_accuracy: 0.7600
Epoch 25/100
21/21 [==============================] - 82s 4s/step - loss: 0.4437 - accuracy:
0.8155 - val_loss: 0.4241 - val_accuracy: 0.7600
Epoch 26/100
21/21 [==============================] - 83s 4s/step - loss: 0.4682 - accuracy:
0.8107 - val_loss: 0.7865 - val_accuracy: 0.6800
Epoch 27/100
21/21 [==============================] - 84s 4s/step - loss: 0.4625 - accuracy:
0.7816 - val_loss: 0.9283 - val_accuracy: 0.6600
Epoch 28/100
21/21 [==============================] - 83s 4s/step - loss: 0.3792 - accuracy:
0.8398 - val_loss: 0.6404 - val_accuracy: 0.8000
Epoch 29/100
21/21 [==============================] - 80s 4s/step - loss: 0.5467 - accuracy:
0.7524 - val_loss: 0.4037 - val_accuracy: 0.7800
Epoch 30/100
21/21 [==============================] - 82s 4s/step - loss: 0.4761 - accuracy:
0.7767 - val_loss: 0.6112 - val_accuracy: 0.8000
Epoch 31/100
21/21 [==============================] - 82s 4s/step - loss: 0.5103 - accuracy:
0.7913 - val_loss: 0.6073 - val_accuracy: 0.7000
Epoch 32/100
21/21 [==============================] - 83s 4s/step - loss: 0.4013 - accuracy:
0.8204 - val_loss: 0.6973 - val_accuracy: 0.7200
Epoch 33/100
21/21 [==============================] - 85s 4s/step - loss: 0.3539 - accuracy:
0.8495 - val_loss: 0.5133 - val_accuracy: 0.7600
Epoch 34/100
21/21 [==============================] - 83s 4s/step - loss: 0.3225 - accuracy:
0.8398 - val_loss: 0.5445 - val_accuracy: 0.7800
Epoch 35/100
21/21 [==============================] - 80s 4s/step - loss: 0.3235 - accuracy:
0.8667 - val_loss: 0.6564 - val_accuracy: 0.7200
Epoch 36/100
21/21 [==============================] - 82s 4s/step - loss: 0.3564 - accuracy:
0.8495 - val_loss: 0.6579 - val_accuracy: 0.6600
Epoch 37/100
21/21 [==============================] - 81s 4s/step - loss: 0.3865 - accuracy:
0.8641 - val_loss: 0.4584 - val_accuracy: 0.7800
Epoch 38/100
21/21 [==============================] - 83s 4s/step - loss: 0.3759 - accuracy:
```

```
0.8641 - val_loss: 0.5324 - val_accuracy: 0.8200
Epoch 39/100
21/21 [==============================] - 83s 4s/step - loss: 0.3093 - accuracy:
0.8786 - val_loss: 0.5150 - val_accuracy: 0.8000
Epoch 40/100
21/21 [==============================] - 83s 4s/step - loss: 0.3049 - accuracy:
0.8835 - val_loss: 0.6403 - val_accuracy: 0.7800
Epoch 41/100
21/21 [==============================] - 80s 4s/step - loss: 0.3007 - accuracy:
0.8883 - val_loss: 0.5046 - val_accuracy: 0.7800
Epoch 42/100
21/21 [==============================] - 81s 4s/step - loss: 0.3070 - accuracy:
0.8786 - val_loss: 0.8840 - val_accuracy: 0.7600
Epoch 43/100
21/21 [==============================] - 83s 4s/step - loss: 0.2559 - accuracy:
0.9078 - val_loss: 0.6013 - val_accuracy: 0.7600
Epoch 44/100
21/21 [==============================] - 83s 4s/step - loss: 0.3167 - accuracy:
0.8932 - val_loss: 0.4911 - val_accuracy: 0.8200
Epoch 45/100
21/21 [==============================] - 83s 4s/step - loss: 0.2905 - accuracy:
0.8932 - val_loss: 0.8152 - val_accuracy: 0.7000
Epoch 46/100
21/21 [==============================] - 84s 4s/step - loss: 0.3022 - accuracy:
0.8932 - val_loss: 0.7685 - val_accuracy: 0.7400
Epoch 47/100
21/21 [==============================] - 82s 4s/step - loss: 0.2077 - accuracy:
0.9272 - val_loss: 0.9132 - val_accuracy: 0.7000
Epoch 48/100
21/21 [==============================] - 81s 4s/step - loss: 0.3445 - accuracy:
0.8495 - val_loss: 0.7784 - val_accuracy: 0.7200
Epoch 49/100
21/21 [==============================] - 82s 4s/step - loss: 0.1559 - accuracy:
0.9466 - val_loss: 0.5442 - val_accuracy: 0.8000
Epoch 50/100
21/21 [==============================] - 82s 4s/step - loss: 0.1706 - accuracy:
0.9272 - val_loss: 0.7064 - val_accuracy: 0.8400
Epoch 51/100
21/21 [==============================] - 81s 4s/step - loss: 0.3524 - accuracy:
0.9029 - val_loss: 0.7628 - val_accuracy: 0.8200
Epoch 52/100
21/21 [==============================] - 84s 4s/step - loss: 0.2273 - accuracy:
0.9126 - val_loss: 0.7873 - val_accuracy: 0.7400
Epoch 53/100
21/21 [==============================] - 82s 4s/step - loss: 0.1908 - accuracy:
0.9515 - val_loss: 0.9048 - val_accuracy: 0.7400
Epoch 54/100
21/21 [==============================] - 80s 4s/step - loss: 0.2347 - accuracy:
```

```
0.9223 - val_loss: 0.5191 - val_accuracy: 0.8000
Epoch 55/100
21/21 [==============================] - 81s 4s/step - loss: 0.1904 - accuracy:
0.9223 - val_loss: 0.5471 - val_accuracy: 0.8000
Epoch 56/100
21/21 [==============================] - 82s 4s/step - loss: 0.1621 - accuracy:
0.9515 - val_loss: 0.7438 - val_accuracy: 0.7800
Epoch 57/100
21/21 [==============================] - 82s 4s/step - loss: 0.1238 - accuracy:
0.9563 - val_loss: 0.8533 - val_accuracy: 0.7800
Epoch 58/100
21/21 [==============================] - 82s 4s/step - loss: 0.1512 - accuracy:
0.9417 - val_loss: 0.6527 - val_accuracy: 0.8000
Epoch 59/100
21/21 [==============================] - 83s 4s/step - loss: 0.2557 - accuracy:
0.9223 - val_loss: 1.5476 - val_accuracy: 0.6600
Epoch 60/100
21/21 [==============================] - 79s 4s/step - loss: 0.2019 - accuracy:
0.9369 - val_loss: 0.9874 - val_accuracy: 0.7200
Epoch 61/100
21/21 [==============================] - 82s 4s/step - loss: 0.2151 - accuracy:
0.9286 - val_loss: 0.8549 - val_accuracy: 0.7400
Epoch 62/100
21/21 [==============================] - 82s 4s/step - loss: 0.2152 - accuracy:
0.9417 - val_loss: 0.8409 - val_accuracy: 0.7200
Epoch 63/100
21/21 [==============================] - 81s 4s/step - loss: 0.2758 - accuracy:
0.9078 - val_loss: 0.9333 - val_accuracy: 0.7600
Epoch 64/100
21/21 [==============================] - 83s 4s/step - loss: 0.1679 - accuracy:
0.9515 - val_loss: 0.8490 - val_accuracy: 0.7800
Epoch 65/100
21/21 [==============================] - 81s 4s/step - loss: 0.1073 - accuracy:
0.9709 - val_loss: 1.1228 - val_accuracy: 0.7600
Epoch 66/100
21/21 [==============================] - 81s 4s/step - loss: 0.1763 - accuracy:
0.9466 - val_loss: 1.2014 - val_accuracy: 0.7200
Epoch 67/100
21/21 [==============================] - 80s 4s/step - loss: 0.1911 - accuracy:
0.9320 - val_loss: 0.8821 - val_accuracy: 0.7800
Epoch 68/100
21/21 [==============================] - 82s 4s/step - loss: 0.1512 - accuracy:
0.9369 - val_loss: 0.9682 - val_accuracy: 0.7400
Epoch 69/100
21/21 [==============================] - 82s 4s/step - loss: 0.1211 - accuracy:
0.9563 - val_loss: 1.2713 - val_accuracy: 0.7200
Epoch 70/100
21/21 [==============================] - 83s 4s/step - loss: 0.1871 - accuracy:
```

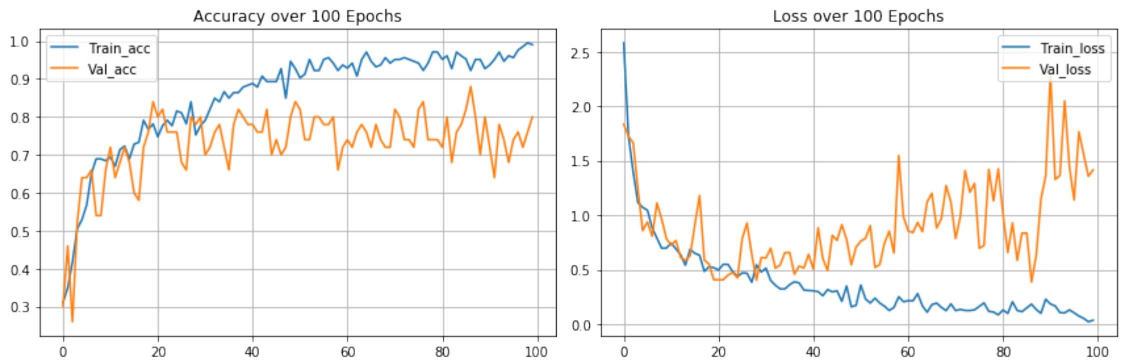
```
0.9417 - val_loss: 1.1213 - val_accuracy: 0.7200
Epoch 71/100
21/21 [==============================] - 84s 4s/step - loss: 0.1226 - accuracy:
0.9515 - val_loss: 0.7859 - val_accuracy: 0.8200
Epoch 72/100
21/21 [==============================] - 81s 4s/step - loss: 0.1323 - accuracy:
0.9515 - val_loss: 0.9874 - val_accuracy: 0.8000
Epoch 73/100
21/21 [==============================] - 80s 4s/step - loss: 0.1229 - accuracy:
0.9563 - val_loss: 1.4103 - val_accuracy: 0.7400
Epoch 74/100
21/21 [==============================] - 82s 4s/step - loss: 0.1224 - accuracy:
0.9515 - val_loss: 1.2101 - val_accuracy: 0.7400
Epoch 75/100
21/21 [==============================] - 82s 4s/step - loss: 0.1315 - accuracy:
0.9466 - val_loss: 1.2926 - val_accuracy: 0.7200
Epoch 76/100
21/21 [==============================] - 82s 4s/step - loss: 0.1618 - accuracy:
0.9417 - val_loss: 0.6963 - val_accuracy: 0.8200
Epoch 77/100
21/21 [==============================] - 83s 4s/step - loss: 0.2068 - accuracy:
0.9223 - val_loss: 0.7221 - val_accuracy: 0.8400
Epoch 78/100
21/21 [==============================] - 82s 4s/step - loss: 0.1194 - accuracy:
0.9417 - val_loss: 1.4209 - val_accuracy: 0.7400
Epoch 79/100
21/21 [==============================] - 80s 4s/step - loss: 0.1101 - accuracy:
0.9714 - val_loss: 1.1314 - val_accuracy: 0.7400
Epoch 80/100
21/21 [==============================] - 82s 4s/step - loss: 0.0837 - accuracy:
0.9709 - val_loss: 1.4255 - val_accuracy: 0.7400
Epoch 81/100
21/21 [==============================] - 83s 4s/step - loss: 0.1283 - accuracy:
0.9515 - val_loss: 1.0120 - val_accuracy: 0.7200
Epoch 82/100
21/21 [==============================] - 82s 4s/step - loss: 0.0987 - accuracy:
0.9612 - val_loss: 0.6565 - val_accuracy: 0.8000
Epoch 83/100
21/21 [==============================] - 83s 4s/step - loss: 0.2015 - accuracy:
0.9272 - val_loss: 0.9290 - val_accuracy: 0.6800
Epoch 84/100
21/21 [==============================] - 90s 4s/step - loss: 0.1232 - accuracy:
0.9709 - val_loss: 0.5848 - val_accuracy: 0.7600
Epoch 85/100
21/21 [==============================] - 50s 2s/step - loss: 0.1146 - accuracy:
0.9612 - val_loss: 0.8353 - val_accuracy: 0.7800
Epoch 86/100
21/21 [==============================] - 50s 2s/step - loss: 0.1486 - accuracy:
```

```
0.9524 - val_loss: 0.8358 - val_accuracy: 0.8200
Epoch 87/100
21/21 [==============================] - 49s 2s/step - loss: 0.1814 - accuracy:
0.9223 - val_loss: 0.3860 - val_accuracy: 0.8800
Epoch 88/100
21/21 [==============================] - 49s 2s/step - loss: 0.1320 - accuracy:
0.9515 - val_loss: 0.6307 - val_accuracy: 0.8000
Epoch 89/100
21/21 [==============================] - 50s 2s/step - loss: 0.0973 - accuracy:
0.9515 - val_loss: 1.1490 - val_accuracy: 0.7000
Epoch 90/100
21/21 [==============================] - 49s 2s/step - loss: 0.2248 - accuracy:
0.9272 - val_loss: 1.3603 - val_accuracy: 0.8000
Epoch 91/100
21/21 [==============================] - 50s 2s/step - loss: 0.1830 - accuracy:
0.9369 - val_loss: 2.2939 - val_accuracy: 0.7200
Epoch 92/100
21/21 [==============================] - 50s 2s/step - loss: 0.1752 - accuracy:
0.9515 - val_loss: 1.3285 - val_accuracy: 0.6400
Epoch 93/100
21/21 [==============================] - 49s 2s/step - loss: 0.1101 - accuracy:
0.9709 - val_loss: 1.3633 - val_accuracy: 0.7800
Epoch 94/100
21/21 [==============================] - 49s 2s/step - loss: 0.1009 - accuracy:
0.9466 - val_loss: 2.0496 - val_accuracy: 0.7400
Epoch 95/100
21/21 [==============================] - 49s 2s/step - loss: 0.1320 - accuracy:
0.9612 - val_loss: 1.4467 - val_accuracy: 0.6800
Epoch 96/100
21/21 [==============================] - 49s 2s/step - loss: 0.1012 - accuracy:
0.9563 - val_loss: 1.1392 - val_accuracy: 0.7400
Epoch 97/100
21/21 [==============================] - 50s 2s/step - loss: 0.0742 - accuracy:
0.9762 - val_loss: 1.7666 - val_accuracy: 0.7600
Epoch 98/100
21/21 [==============================] - 56s 3s/step - loss: 0.0520 - accuracy:
0.9854 - val_loss: 1.5605 - val_accuracy: 0.7200
Epoch 99/100
21/21 [==============================] - 55s 3s/step - loss: 0.0212 - accuracy:
0.9951 - val_loss: 1.3573 - val_accuracy: 0.7600
Epoch 100/100
21/21 [==============================] - 59s 3s/step - loss: 0.0354 - accuracy:
0.9905 - val_loss: 1.4161 - val_accuracy: 0.8000
```

[5 points] Plot Accuracy and Loss During Training

[6]:
```python
import matplotlib.pyplot as plt
```

```
plt.figure(figsize = (12,4))

plt.subplot(1, 2, 1)
plt.grid()
plt.plot(history.history['accuracy'], label='Train_acc')
plt.plot(history.history['val_accuracy'], label = 'Val_acc')
plt.title("Accuracy over 100 Epochs")
plt.legend(loc='upper left')

plt.subplot(1, 2, 2)
plt.grid()
plt.plot(history.history['loss'], label='Train_loss')
plt.plot(history.history['val_loss'], label = 'Val_loss')
plt.title("Loss over 100 Epochs")
plt.legend(loc='upper right')

plt.tight_layout()
plt.show()
```



Testing Model

[11]:
```
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_generator(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])
```

```
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 15s 429ms/step - loss: 1.2045 -
accuracy: 0.8056
Test loss: 1.2045319996486077
Test accuracy: 0.8055556
```

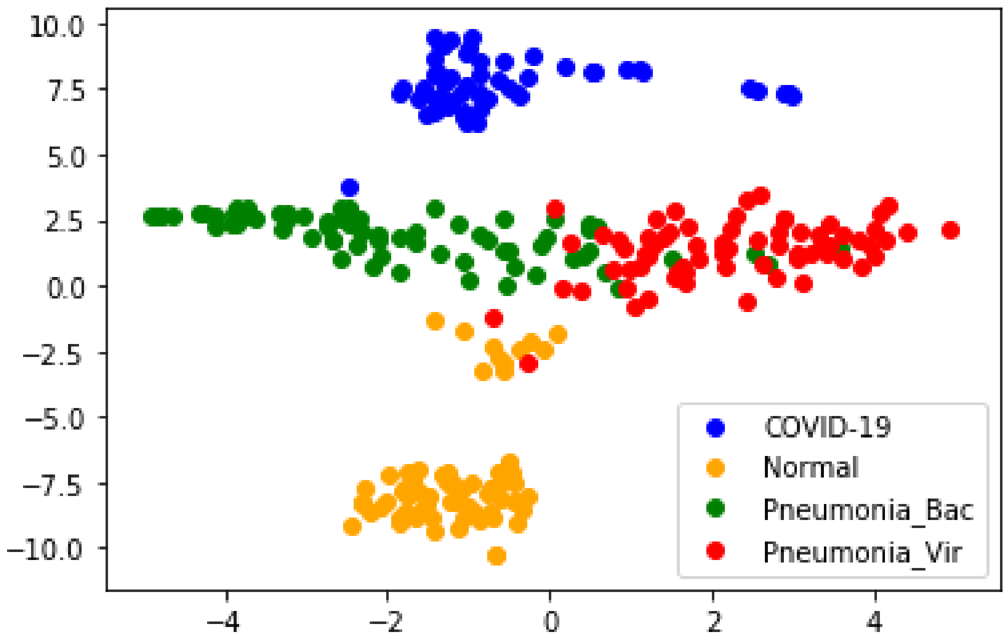
2.5 [10 points] TSNE Plot

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.

[9]:

```python
from sklearn.manifold import TSNE

intermediate_layer_model = tf.keras.models.Model(inputs=model.input,
                                        outputs=model.get_layer('dense').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")

features = intermediate_layer_model.predict_generator(tsne_eval_generator)

labels = tsne_eval_generator.labels

tsne = TSNE(perplexity = 75, learning_rate = 80).fit_transform(features)

for i in range(len(tsne)):
    if labels[i] == 0:
        # covid
        c = plt.scatter(tsne[:,0][i], tsne[:,1][i], color = 'blue')
    elif labels[i] == 1:
        # normal
        n = plt.scatter(tsne[:,0][i], tsne[:,1][i], color = 'orange')
    elif labels[i] == 2:
        # pneumonia_bac
        pb = plt.scatter(tsne[:,0][i], tsne[:,1][i], color = 'green')
    else:
        # pneumonia_vir
        pv = plt.scatter(tsne[:,0][i], tsne[:,1][i], color = 'red')

plt.legend((c, n, pb, pv), ("COVID-19", "Normal", "Pneumonia_Bac", \
                            "Pneumonia_Vir"))

plt.show()
```

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



2.6 Model 2 - VGG16

[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.

[15]:
```
vgg_16 = VGG16(weights = 'imagenet', include_top = False, \
               input_shape = (224,224,3), classes = NUM_CLASSES)

for layer in vgg_16.layers:
    layer.trainable = False

model2 = tf.keras.models.Sequential()

model2.add(vgg_16)

model2.add(tf.keras.layers.Flatten())

model2.add(tf.keras.layers.Dropout(.15))

model2.add(tf.keras.layers.Dense(256, activation = 'relu'))

model2.add(tf.keras.layers.Dropout(.15))

model2.add(tf.keras.layers.Dense(4, activation = 'softmax'))
```

```
model2.summary()
```

```
Model: "sequential_2"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_2 (Flatten)          (None, 25088)             0
_________________________________________________________________
dropout_4 (Dropout)          (None, 25088)             0
_________________________________________________________________
dense_4 (Dense)              (None, 256)               6422784
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_5 (Dense)              (None, 4)                 1028
=================================================================
Total params: 21,138,500
Trainable params: 6,423,812
Non-trainable params: 14,714,688
_________________________________________________________________
```

[5 points] Train Model

[16]:
```
#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

OPTIMIZER = tf.keras.optimizers.Adam(learning_rate = LEARNING_RATE)

model2.compile(optimizer = OPTIMIZER, \
                loss=tf.keras.losses.CategoricalCrossentropy(), \
                metrics=['accuracy'])


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

```
22
6
Train for 21 steps, validate for 5 steps
Epoch 1/100
```

```
21/21 [==============================] - 67s 3s/step - loss: 1.4669 - accuracy:
0.3932 - val_loss: 1.3730 - val_accuracy: 0.4400
Epoch 2/100
21/21 [==============================] - 57s 3s/step - loss: 1.2117 - accuracy:
0.4612 - val_loss: 0.9771 - val_accuracy: 0.5000
Epoch 3/100
21/21 [==============================] - 40s 2s/step - loss: 1.0722 - accuracy:
0.5291 - val_loss: 1.0043 - val_accuracy: 0.6000
Epoch 4/100
21/21 [==============================] - 40s 2s/step - loss: 1.0843 - accuracy:
0.5437 - val_loss: 1.1655 - val_accuracy: 0.4000
Epoch 5/100
21/21 [==============================] - 41s 2s/step - loss: 0.9862 - accuracy:
0.5680 - val_loss: 0.9938 - val_accuracy: 0.4600
Epoch 6/100
21/21 [==============================] - 43s 2s/step - loss: 0.9835 - accuracy:
0.5291 - val_loss: 0.9200 - val_accuracy: 0.5400
Epoch 7/100
21/21 [==============================] - 41s 2s/step - loss: 0.9227 - accuracy:
0.5437 - val_loss: 0.8223 - val_accuracy: 0.6400
Epoch 8/100
21/21 [==============================] - 41s 2s/step - loss: 0.9652 - accuracy:
0.5971 - val_loss: 0.8462 - val_accuracy: 0.6000
Epoch 9/100
21/21 [==============================] - 42s 2s/step - loss: 0.8416 - accuracy:
0.6262 - val_loss: 0.9351 - val_accuracy: 0.4800
Epoch 10/100
21/21 [==============================] - 42s 2s/step - loss: 0.8879 - accuracy:
0.6262 - val_loss: 0.8435 - val_accuracy: 0.5600
Epoch 11/100
21/21 [==============================] - 42s 2s/step - loss: 0.8228 - accuracy:
0.6117 - val_loss: 0.8565 - val_accuracy: 0.6400
Epoch 12/100
21/21 [==============================] - 43s 2s/step - loss: 0.8518 - accuracy:
0.6000 - val_loss: 0.7840 - val_accuracy: 0.6800
Epoch 13/100
21/21 [==============================] - 41s 2s/step - loss: 0.7883 - accuracy:
0.6408 - val_loss: 0.7579 - val_accuracy: 0.6800
Epoch 14/100
21/21 [==============================] - 41s 2s/step - loss: 0.8324 - accuracy:
0.6311 - val_loss: 0.7016 - val_accuracy: 0.6400
Epoch 15/100
21/21 [==============================] - 41s 2s/step - loss: 0.8351 - accuracy:
0.6408 - val_loss: 0.8454 - val_accuracy: 0.6400
Epoch 16/100
21/21 [==============================] - 41s 2s/step - loss: 0.8050 - accuracy:
0.6408 - val_loss: 0.8872 - val_accuracy: 0.6200
Epoch 17/100
```

```
21/21 [==============================] - 41s 2s/step - loss: 0.6914 - accuracy:
0.6845 - val_loss: 0.8136 - val_accuracy: 0.5800
Epoch 18/100
21/21 [==============================] - 41s 2s/step - loss: 0.7798 - accuracy:
0.7233 - val_loss: 0.9131 - val_accuracy: 0.5200
Epoch 19/100
21/21 [==============================] - 44s 2s/step - loss: 0.8211 - accuracy:
0.6845 - val_loss: 0.8092 - val_accuracy: 0.6200
Epoch 20/100
21/21 [==============================] - 42s 2s/step - loss: 0.7647 - accuracy:
0.6748 - val_loss: 0.9460 - val_accuracy: 0.5600
Epoch 21/100
21/21 [==============================] - 41s 2s/step - loss: 0.6935 - accuracy:
0.7087 - val_loss: 0.7922 - val_accuracy: 0.5600
Epoch 22/100
21/21 [==============================] - 42s 2s/step - loss: 0.7626 - accuracy:
0.6714 - val_loss: 0.8534 - val_accuracy: 0.5800
Epoch 23/100
21/21 [==============================] - 43s 2s/step - loss: 0.7376 - accuracy:
0.6796 - val_loss: 0.6829 - val_accuracy: 0.6600
Epoch 24/100
21/21 [==============================] - 41s 2s/step - loss: 0.7629 - accuracy:
0.6165 - val_loss: 0.6310 - val_accuracy: 0.6800
Epoch 25/100
21/21 [==============================] - 42s 2s/step - loss: 0.7023 - accuracy:
0.6456 - val_loss: 0.7674 - val_accuracy: 0.6200
Epoch 26/100
21/21 [==============================] - 42s 2s/step - loss: 0.7456 - accuracy:
0.6553 - val_loss: 0.7602 - val_accuracy: 0.6600
Epoch 27/100
21/21 [==============================] - 41s 2s/step - loss: 0.7095 - accuracy:
0.6699 - val_loss: 0.8742 - val_accuracy: 0.5800
Epoch 28/100
21/21 [==============================] - 41s 2s/step - loss: 0.7369 - accuracy:
0.6748 - val_loss: 0.7878 - val_accuracy: 0.6400
Epoch 29/100
21/21 [==============================] - 41s 2s/step - loss: 0.7360 - accuracy:
0.6699 - val_loss: 0.7833 - val_accuracy: 0.6600
Epoch 30/100
21/21 [==============================] - 41s 2s/step - loss: 0.7045 - accuracy:
0.6748 - val_loss: 0.9846 - val_accuracy: 0.5400
Epoch 31/100
21/21 [==============================] - 41s 2s/step - loss: 0.6883 - accuracy:
0.6893 - val_loss: 0.7226 - val_accuracy: 0.6400
Epoch 32/100
21/21 [==============================] - 41s 2s/step - loss: 0.7150 - accuracy:
0.6796 - val_loss: 0.8466 - val_accuracy: 0.5200
Epoch 33/100
```

```
21/21 [==============================] - 41s 2s/step - loss: 0.6662 - accuracy:
0.6748 - val_loss: 0.6116 - val_accuracy: 0.6800
Epoch 34/100
21/21 [==============================] - 42s 2s/step - loss: 0.6253 - accuracy:
0.6845 - val_loss: 0.6315 - val_accuracy: 0.7000
Epoch 35/100
21/21 [==============================] - 42s 2s/step - loss: 0.6268 - accuracy:
0.7573 - val_loss: 0.6350 - val_accuracy: 0.6600
Epoch 36/100
21/21 [==============================] - 45s 2s/step - loss: 0.5871 - accuracy:
0.7427 - val_loss: 0.6916 - val_accuracy: 0.6800
Epoch 37/100
21/21 [==============================] - 42s 2s/step - loss: 0.5817 - accuracy:
0.7864 - val_loss: 0.5597 - val_accuracy: 0.7800
Epoch 38/100
21/21 [==============================] - 41s 2s/step - loss: 0.6599 - accuracy:
0.7330 - val_loss: 0.6499 - val_accuracy: 0.7000
Epoch 39/100
21/21 [==============================] - 42s 2s/step - loss: 0.6454 - accuracy:
0.7282 - val_loss: 0.6641 - val_accuracy: 0.7600
Epoch 40/100
21/21 [==============================] - 43s 2s/step - loss: 0.6088 - accuracy:
0.7476 - val_loss: 0.6310 - val_accuracy: 0.6400
Epoch 41/100
21/21 [==============================] - 42s 2s/step - loss: 0.6042 - accuracy:
0.7524 - val_loss: 0.6799 - val_accuracy: 0.6600
Epoch 42/100
21/21 [==============================] - 42s 2s/step - loss: 0.6457 - accuracy:
0.7087 - val_loss: 0.5890 - val_accuracy: 0.7200
Epoch 43/100
21/21 [==============================] - 41s 2s/step - loss: 0.6867 - accuracy:
0.7087 - val_loss: 0.6151 - val_accuracy: 0.6800
Epoch 44/100
21/21 [==============================] - 41s 2s/step - loss: 0.5933 - accuracy:
0.7427 - val_loss: 0.5957 - val_accuracy: 0.6800
Epoch 45/100
21/21 [==============================] - 42s 2s/step - loss: 0.5856 - accuracy:
0.7282 - val_loss: 0.7275 - val_accuracy: 0.6400
Epoch 46/100
21/21 [==============================] - 41s 2s/step - loss: 0.5594 - accuracy:
0.7621 - val_loss: 0.5492 - val_accuracy: 0.7600
Epoch 47/100
21/21 [==============================] - 41s 2s/step - loss: 0.5682 - accuracy:
0.7379 - val_loss: 0.6900 - val_accuracy: 0.7000
Epoch 48/100
21/21 [==============================] - 41s 2s/step - loss: 0.5531 - accuracy:
0.7184 - val_loss: 0.6908 - val_accuracy: 0.7400
Epoch 49/100
```

```
21/21 [==============================] - 41s 2s/step - loss: 0.6268 - accuracy:
0.7670 - val_loss: 0.8129 - val_accuracy: 0.6400
Epoch 50/100
21/21 [==============================] - 41s 2s/step - loss: 0.6257 - accuracy:
0.7233 - val_loss: 0.6456 - val_accuracy: 0.6400
Epoch 51/100
21/21 [==============================] - 41s 2s/step - loss: 0.6895 - accuracy:
0.6699 - val_loss: 0.6519 - val_accuracy: 0.7200
Epoch 52/100
21/21 [==============================] - 42s 2s/step - loss: 0.6567 - accuracy:
0.7184 - val_loss: 0.6582 - val_accuracy: 0.6800
Epoch 53/100
21/21 [==============================] - 42s 2s/step - loss: 0.6184 - accuracy:
0.7039 - val_loss: 0.5914 - val_accuracy: 0.7400
Epoch 54/100
21/21 [==============================] - 41s 2s/step - loss: 0.5922 - accuracy:
0.7573 - val_loss: 0.5934 - val_accuracy: 0.7600
Epoch 55/100
21/21 [==============================] - 41s 2s/step - loss: 0.6085 - accuracy:
0.7282 - val_loss: 0.5334 - val_accuracy: 0.6800
Epoch 56/100
21/21 [==============================] - 43s 2s/step - loss: 0.6476 - accuracy:
0.7233 - val_loss: 0.6180 - val_accuracy: 0.7400
Epoch 57/100
21/21 [==============================] - 42s 2s/step - loss: 0.6245 - accuracy:
0.7379 - val_loss: 0.6934 - val_accuracy: 0.6600
Epoch 58/100
21/21 [==============================] - 42s 2s/step - loss: 0.5529 - accuracy:
0.7427 - val_loss: 0.7933 - val_accuracy: 0.5200
Epoch 59/100
21/21 [==============================] - 41s 2s/step - loss: 0.5898 - accuracy:
0.7524 - val_loss: 0.6689 - val_accuracy: 0.6400
Epoch 60/100
21/21 [==============================] - 41s 2s/step - loss: 0.6561 - accuracy:
0.7330 - val_loss: 0.7516 - val_accuracy: 0.6200
Epoch 61/100
21/21 [==============================] - 41s 2s/step - loss: 0.6259 - accuracy:
0.7476 - val_loss: 0.7884 - val_accuracy: 0.6800
Epoch 62/100
21/21 [==============================] - 42s 2s/step - loss: 0.5427 - accuracy:
0.7573 - val_loss: 0.6802 - val_accuracy: 0.6600
Epoch 63/100
21/21 [==============================] - 41s 2s/step - loss: 0.6357 - accuracy:
0.7573 - val_loss: 0.6655 - val_accuracy: 0.7000
Epoch 64/100
21/21 [==============================] - 40s 2s/step - loss: 0.5894 - accuracy:
0.7573 - val_loss: 0.7215 - val_accuracy: 0.6200
Epoch 65/100
```

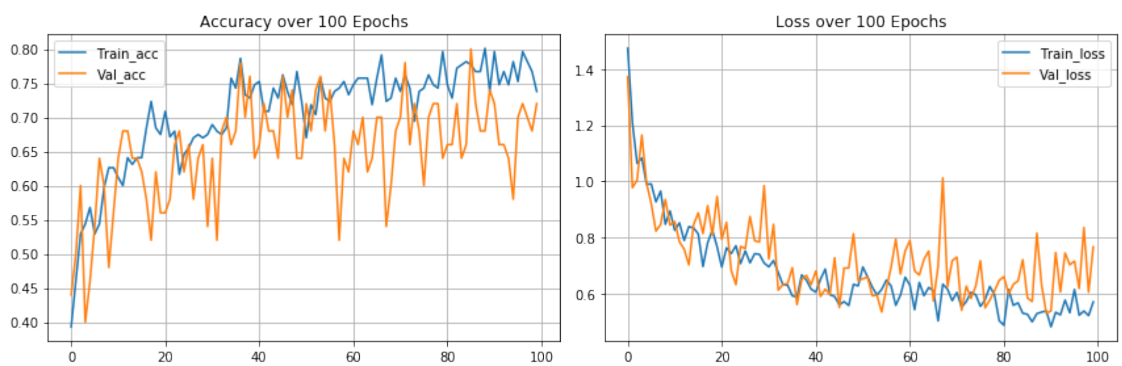
```
21/21 [==============================] - 40s 2s/step - loss: 0.6223 - accuracy:
0.7184 - val_loss: 0.7509 - val_accuracy: 0.6400
Epoch 66/100
21/21 [==============================] - 40s 2s/step - loss: 0.6075 - accuracy:
0.7573 - val_loss: 0.5728 - val_accuracy: 0.7000
Epoch 67/100
21/21 [==============================] - 41s 2s/step - loss: 0.4966 - accuracy:
0.7913 - val_loss: 0.6960 - val_accuracy: 0.7000
Epoch 68/100
21/21 [==============================] - 40s 2s/step - loss: 0.6298 - accuracy:
0.7233 - val_loss: 1.0122 - val_accuracy: 0.5400
Epoch 69/100
21/21 [==============================] - 41s 2s/step - loss: 0.6107 - accuracy:
0.7282 - val_loss: 0.6237 - val_accuracy: 0.6000
Epoch 70/100
21/21 [==============================] - 40s 2s/step - loss: 0.5742 - accuracy:
0.7573 - val_loss: 0.7168 - val_accuracy: 0.6800
Epoch 71/100
21/21 [==============================] - 40s 2s/step - loss: 0.6039 - accuracy:
0.7379 - val_loss: 0.7295 - val_accuracy: 0.7000
Epoch 72/100
21/21 [==============================] - 40s 2s/step - loss: 0.5462 - accuracy:
0.7621 - val_loss: 0.5391 - val_accuracy: 0.7800
Epoch 73/100
21/21 [==============================] - 40s 2s/step - loss: 0.5668 - accuracy:
0.7427 - val_loss: 0.6258 - val_accuracy: 0.6600
Epoch 74/100
21/21 [==============================] - 40s 2s/step - loss: 0.6023 - accuracy:
0.6942 - val_loss: 0.5821 - val_accuracy: 0.7200
Epoch 75/100
21/21 [==============================] - 40s 2s/step - loss: 0.5900 - accuracy:
0.7379 - val_loss: 0.6254 - val_accuracy: 0.6800
Epoch 76/100
21/21 [==============================] - 40s 2s/step - loss: 0.5522 - accuracy:
0.7427 - val_loss: 0.7164 - val_accuracy: 0.6000
Epoch 77/100
21/21 [==============================] - 40s 2s/step - loss: 0.5769 - accuracy:
0.7621 - val_loss: 0.5481 - val_accuracy: 0.7000
Epoch 78/100
21/21 [==============================] - 40s 2s/step - loss: 0.6249 - accuracy:
0.7476 - val_loss: 0.5768 - val_accuracy: 0.7200
Epoch 79/100
21/21 [==============================] - 40s 2s/step - loss: 0.5874 - accuracy:
0.7427 - val_loss: 0.6086 - val_accuracy: 0.7200
Epoch 80/100
21/21 [==============================] - 41s 2s/step - loss: 0.4981 - accuracy:
0.7961 - val_loss: 0.6468 - val_accuracy: 0.6400
Epoch 81/100
```

```
21/21 [==============================] - 40s 2s/step - loss: 0.4849 - accuracy:
0.7476 - val_loss: 0.6594 - val_accuracy: 0.6600
Epoch 82/100
21/21 [==============================] - 41s 2s/step - loss: 0.6108 - accuracy:
0.7282 - val_loss: 0.5979 - val_accuracy: 0.6600
Epoch 83/100
21/21 [==============================] - 40s 2s/step - loss: 0.5529 - accuracy:
0.7718 - val_loss: 0.6322 - val_accuracy: 0.7200
Epoch 84/100
21/21 [==============================] - 40s 2s/step - loss: 0.5609 - accuracy:
0.7767 - val_loss: 0.6447 - val_accuracy: 0.6400
Epoch 85/100
21/21 [==============================] - 40s 2s/step - loss: 0.5266 - accuracy:
0.7816 - val_loss: 0.7202 - val_accuracy: 0.6600
Epoch 86/100
21/21 [==============================] - 40s 2s/step - loss: 0.5234 - accuracy:
0.7767 - val_loss: 0.5835 - val_accuracy: 0.8000
Epoch 87/100
21/21 [==============================] - 40s 2s/step - loss: 0.4951 - accuracy:
0.7670 - val_loss: 0.5707 - val_accuracy: 0.7200
Epoch 88/100
21/21 [==============================] - 40s 2s/step - loss: 0.5226 - accuracy:
0.7670 - val_loss: 0.8152 - val_accuracy: 0.6800
Epoch 89/100
21/21 [==============================] - 41s 2s/step - loss: 0.5342 - accuracy:
0.8010 - val_loss: 0.6309 - val_accuracy: 0.6800
Epoch 90/100
21/21 [==============================] - 40s 2s/step - loss: 0.5374 - accuracy:
0.7381 - val_loss: 0.5297 - val_accuracy: 0.7400
Epoch 91/100
21/21 [==============================] - 40s 2s/step - loss: 0.4812 - accuracy:
0.7961 - val_loss: 0.5368 - val_accuracy: 0.7200
Epoch 92/100
21/21 [==============================] - 40s 2s/step - loss: 0.5290 - accuracy:
0.7476 - val_loss: 0.7456 - val_accuracy: 0.6600
Epoch 93/100
21/21 [==============================] - 40s 2s/step - loss: 0.5242 - accuracy:
0.7670 - val_loss: 0.6047 - val_accuracy: 0.6600
Epoch 94/100
21/21 [==============================] - 40s 2s/step - loss: 0.5712 - accuracy:
0.7476 - val_loss: 0.7440 - val_accuracy: 0.6400
Epoch 95/100
21/21 [==============================] - 42s 2s/step - loss: 0.5338 - accuracy:
0.7816 - val_loss: 0.7018 - val_accuracy: 0.5800
Epoch 96/100
21/21 [==============================] - 41s 2s/step - loss: 0.6164 - accuracy:
0.7524 - val_loss: 0.7154 - val_accuracy: 0.7000
Epoch 97/100
```

```
21/21 [==============================] - 41s 2s/step - loss: 0.5172 - accuracy:
0.7961 - val_loss: 0.6170 - val_accuracy: 0.7200
Epoch 98/100
21/21 [==============================] - 41s 2s/step - loss: 0.5369 - accuracy:
0.7816 - val_loss: 0.8350 - val_accuracy: 0.7000
Epoch 99/100
21/21 [==============================] - 41s 2s/step - loss: 0.5164 - accuracy:
0.7670 - val_loss: 0.6042 - val_accuracy: 0.6800
Epoch 100/100
21/21 [==============================] - 41s 2s/step - loss: 0.5685 - accuracy:
0.7379 - val_loss: 0.7662 - val_accuracy: 0.7200
```

[5 points] Plot Accuracy and Loss During Training

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

plt.figure(figsize = (12,4))

plt.subplot(1, 2, 1)
plt.grid()
plt.plot(history2.history['accuracy'], label='Train_acc')
plt.plot(history2.history['val_accuracy'], label = 'Val_acc')
plt.title("Accuracy over 100 Epochs")
plt.legend(loc='upper left')

plt.subplot(1, 2, 2)
plt.grid()
plt.plot(history2.history['loss'], label='Train_loss')
plt.plot(history2.history['val_loss'], label = 'Val_loss')
plt.title("Loss over 100 Epochs")
plt.legend(loc='upper right')

plt.tight_layout()
plt.show()
```



Testing Model

[18]:
```
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_generator(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])
```

```
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 7s 184ms/step - loss: 0.7583 -
accuracy: 0.7222
Test loss: 0.7583347872914601
Test accuracy: 0.7222222
```

2.7 [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.

[21]:
```
from sklearn.manifold import TSNE

intermediate_layer_model = tf.keras.models.Model(inputs=model2.input, \
                                        outputs=model2.get_layer('dense_4').
↪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")

features = intermediate_layer_model.predict_generator(tsne_eval_generator)

labels = tsne_eval_generator.labels

tsne = TSNE(perplexity = 75, learning_rate = 80).fit_transform(features)

for i in range(len(tsne)):
```

```
    if labels[i] == 0:
        # covid
        c = plt.scatter(tsne[:,0][i], tsne[:,1][i], color = 'blue')
    elif labels[i] == 1:
        # normal
        n = plt.scatter(tsne[:,0][i], tsne[:,1][i], color = 'orange')
    elif labels[i] == 2:
        # pneumonia_bac
        pb = plt.scatter(tsne[:,0][i], tsne[:,1][i], color = 'green')
    else:
        # pneumonia_vir
        pv = plt.scatter(tsne[:,0][i], tsne[:,1][i], color = 'red')

plt.legend((c, n, pb, pv), ("COVID-19", "Normal", "Pneumonia_Bac", \
                            "Pneumonia_Vir"))

plt.show()
```

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