In [1]:
```
from google.colab import drive
drive.mount('/gdrive')
%cd /gdrive
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g
3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=c
ode&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fau
th%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /gdrive
/gdrive
```

In [4]:
```
import tensorflow as tf
import numpy as np
tf.__version__
tf.test.gpu_device_name()
```

Out[4]:
```
'/device:GPU:0'
```

In [5]:
```
gpu_info = !nvidia-smi
gpu_info = '\n'.join(gpu_info)
if gpu_info.find('failed') >= 0:
  print('Select the Runtime → "Change runtime type" menu to enable a GPU accelerator, ')
  print('and then re-execute this cell.')
else:
  print(gpu_info)
```

```
Fri Apr 24 20:08:28 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.64.00    Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   46C    P0    35W / 250W |    353MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

In [ ]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications.resnet_v2 import ResNet152V2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import cv2
import os
```

In [7]:
```
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
import numpy as np
import argparse
import cv2
import os
from imutils import paths
```

```
Using TensorFlow backend.
```

In [ ]:
```
```

In [ ]:
```
x_train=np.load('My Drive/Colab Notebooks/215-FP_Dataset/numpy5/x_train.npy')
```

In [ ]:
```
y_train=np.load('My Drive/Colab Notebooks/215-FP_Dataset/numpy5/y_train.npy')
```

In [ ]:
```
x_valid=np.load('My Drive/Colab Notebooks/215-FP_Dataset/numpy5/x_valid.npy')
```

In [ ]:
```
y_valid=np.load('My Drive/Colab Notebooks/215-FP_Dataset/numpy5/y_valid.npy')
```

In [ ]:
```

```

In [12]:
```
y_valid
```

Out[12]:
```
array([[1., 0.],
       [1., 0.],
       [1., 0.],
       ...,
       [1., 0.],
       [1., 0.],
       [1., 0.]])
```

In [ ]:
```
y_train2=y_train
y_valid2=y_valid
```

In [ ]:
```
y_train2=np.argmax(y_train,axis=1)
y_valid2=np.argmax(y_valid,axis=1)
```

In [16]:
```
y_train2[:50]
```

Out[16]:
```
array([0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 1])
```

In [17]:
```
y_valid2[:50]
```

Out[17]:
```
array([0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 1])
```

In [18]:
```
summation1=0
for i in range(len(x_valid)):
    if y_valid2[i]==0:
        summation1=summation1+1
summation1
```

Out[18]: 1019

In [19]:
```
summation2=0
for i in range(len(x_train)):
    if y_train2[i]==0:
        summation2=summation2+1
summation2
```

Out[19]: 4074

In [ ]:
```
#y_train=y_train2
y_train=((y_train)/255.).astype('float16')
```

In [ ]:
```
#y_valid=y_valid2
y_valid=((y_valid)/255.).astype('float16')
```

In [ ]:
```
INIT_LR = 1e-3
EPOCHS = 100
BS = 128
```

In [ ]:
```
trainX=x_train
trainY=y_train
testX=x_valid
testY=y_valid
```

In [ ]:
```
# initialize the training data augmentation object
trainAug = ImageDataGenerator(
	rotation_range=15,
	fill_mode="nearest")

# load the VGG16 network, ensuring the head FC layer sets are left
# off
baseModel = ResNet152V2(weights="imagenet", include_top=False,
	input_tensor=Input(shape=(128, 128, 3)))
```

In [ ]:
```
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(4, 4))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(64, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
	layer.trainable = False
```

In [37]:
```
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss='binary_crossentropy', optimizer=opt,
        metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model.fit_generator(
        trainAug.flow(trainX, trainY, batch_size=BS),
        steps_per_epoch=len(trainX) // BS,
        validation_data=(testX, testY),
        validation_steps=len(testX) // BS,
        epochs=EPOCHS)
```

```
[INFO] compiling model...
[INFO] training head...
Epoch 1/100
47/47 [==============================] - 27s 569ms/step - loss: 0.7581 - accuracy: 0.5449 - val_loss: 0.6932
- val_accuracy: 0.5753
Epoch 2/100
47/47 [==============================] - 24s 515ms/step - loss: 0.6933 - accuracy: 0.6112 - val_loss: 0.6932
- val_accuracy: 0.6311
Epoch 3/100
47/47 [==============================] - 24s 507ms/step - loss: 0.6932 - accuracy: 0.6295 - val_loss: 0.6932
- val_accuracy: 0.6423
Epoch 4/100
47/47 [==============================] - 24s 507ms/step - loss: 0.6932 - accuracy: 0.6500 - val_loss: 0.6932
- val_accuracy: 0.6523
Epoch 5/100
47/47 [==============================] - 24s 506ms/step - loss: 0.6932 - accuracy: 0.6508 - val_loss: 0.6932
- val_accuracy: 0.6530
Epoch 6/100
47/47 [==============================] - 24s 506ms/step - loss: 0.6932 - accuracy: 0.6651 - val_loss: 0.6932
- val_accuracy: 0.6549
Epoch 7/100
47/47 [==============================] - 24s 504ms/step - loss: 0.6932 - accuracy: 0.6554 - val_loss: 0.6932
- val_accuracy: 0.6583
Epoch 8/100
47/47 [==============================] - 24s 507ms/step - loss: 0.6932 - accuracy: 0.6575 - val_loss: 0.6932
- val_accuracy: 0.6583
Epoch 9/100
47/47 [==============================] - 24s 513ms/step - loss: 0.6932 - accuracy: 0.6614 - val_loss: 0.6932
- val_accuracy: 0.6589
Epoch 10/100
47/47 [==============================] - 24s 510ms/step - loss: 0.6932 - accuracy: 0.6700 - val_loss: 0.6932
- val_accuracy: 0.6596
Epoch 11/100
47/47 [==============================] - 24s 508ms/step - loss: 0.6932 - accuracy: 0.6612 - val_loss: 0.6932
- val_accuracy: 0.6629
Epoch 12/100
47/47 [==============================] - 24s 505ms/step - loss: 0.6932 - accuracy: 0.6666 - val_loss: 0.6932
- val_accuracy: 0.6642
Epoch 13/100
47/47 [==============================] - 24s 515ms/step - loss: 0.6932 - accuracy: 0.6637 - val_loss: 0.6932
- val_accuracy: 0.6662
Epoch 14/100
47/47 [==============================] - 24s 508ms/step - loss: 0.6932 - accuracy: 0.6668 - val_loss: 0.6932
```

```
- val_accuracy: 0.6669
Epoch 15/100
47/47 [==============================] - 24s 503ms/step - loss: 0.6932 - accuracy: 0.6703 - val_loss: 0.6932
- val_accuracy: 0.6669
Epoch 16/100
47/47 [==============================] - 24s 517ms/step - loss: 0.6932 - accuracy: 0.6734 - val_loss: 0.6932
- val_accuracy: 0.6682
Epoch 17/100
47/47 [==============================] - 24s 507ms/step - loss: 0.6932 - accuracy: 0.6695 - val_loss: 0.6932
- val_accuracy: 0.6689
Epoch 18/100
47/47 [==============================] - 24s 509ms/step - loss: 0.6932 - accuracy: 0.6632 - val_loss: 0.6932
- val_accuracy: 0.6695
Epoch 19/100
47/47 [==============================] - 24s 507ms/step - loss: 0.6932 - accuracy: 0.6785 - val_loss: 0.6932
- val_accuracy: 0.6702
Epoch 20/100
47/47 [==============================] - 24s 507ms/step - loss: 0.6932 - accuracy: 0.6676 - val_loss: 0.6932
- val_accuracy: 0.6702
Epoch 21/100
47/47 [==============================] - 24s 503ms/step - loss: 0.6931 - accuracy: 0.6761 - val_loss: 0.6932
- val_accuracy: 0.6709
Epoch 22/100
47/47 [==============================] - 24s 504ms/step - loss: 0.6931 - accuracy: 0.6693 - val_loss: 0.6932
- val_accuracy: 0.6709
Epoch 23/100
47/47 [==============================] - 24s 507ms/step - loss: 0.6932 - accuracy: 0.6751 - val_loss: 0.6932
- val_accuracy: 0.6715
Epoch 24/100
47/47 [==============================] - 24s 504ms/step - loss: 0.6931 - accuracy: 0.6790 - val_loss: 0.6932
- val_accuracy: 0.6715
Epoch 25/100
47/47 [==============================] - 24s 507ms/step - loss: 0.6931 - accuracy: 0.6725 - val_loss: 0.6932
- val_accuracy: 0.6722
Epoch 26/100
47/47 [==============================] - 24s 507ms/step - loss: 0.6931 - accuracy: 0.6639 - val_loss: 0.6932
- val_accuracy: 0.6715
Epoch 27/100
47/47 [==============================] - 24s 508ms/step - loss: 0.6931 - accuracy: 0.6865 - val_loss: 0.6931
- val_accuracy: 0.6722
Epoch 28/100
47/47 [==============================] - 24s 503ms/step - loss: 0.6931 - accuracy: 0.6664 - val_loss: 0.6931
- val_accuracy: 0.6729
```

```
Epoch 29/100
47/47 [==============================] - 24s 516ms/step - loss: 0.6931 - accuracy: 0.6815 - val_loss: 0.6931
- val_accuracy: 0.6742
Epoch 30/100
47/47 [==============================] - 24s 507ms/step - loss: 0.6931 - accuracy: 0.6746 - val_loss: 0.6931
- val_accuracy: 0.6782
Epoch 31/100
47/47 [==============================] - 24s 508ms/step - loss: 0.6931 - accuracy: 0.6803 - val_loss: 0.6931
- val_accuracy: 0.6868
Epoch 32/100
47/47 [==============================] - 24s 509ms/step - loss: 0.6931 - accuracy: 0.6812 - val_loss: 0.6931
- val_accuracy: 0.6875
Epoch 33/100
47/47 [==============================] - 24s 505ms/step - loss: 0.6931 - accuracy: 0.6834 - val_loss: 0.6931
- val_accuracy: 0.6888
Epoch 34/100
47/47 [==============================] - 24s 506ms/step - loss: 0.6931 - accuracy: 0.6848 - val_loss: 0.6931
- val_accuracy: 0.6928
Epoch 35/100
47/47 [==============================] - 24s 514ms/step - loss: 0.6931 - accuracy: 0.6854 - val_loss: 0.6931
- val_accuracy: 0.7001
Epoch 36/100
47/47 [==============================] - 24s 502ms/step - loss: 0.6931 - accuracy: 0.6992 - val_loss: 0.6931
- val_accuracy: 0.7054
Epoch 37/100
47/47 [==============================] - 24s 509ms/step - loss: 0.6931 - accuracy: 0.6934 - val_loss: 0.6931
- val_accuracy: 0.7120
Epoch 38/100
47/47 [==============================] - 24s 508ms/step - loss: 0.6931 - accuracy: 0.7041 - val_loss: 0.6931
- val_accuracy: 0.7206
Epoch 39/100
47/47 [==============================] - 24s 506ms/step - loss: 0.6931 - accuracy: 0.7107 - val_loss: 0.6931
- val_accuracy: 0.7359
Epoch 40/100
47/47 [==============================] - 24s 502ms/step - loss: 0.6931 - accuracy: 0.7115 - val_loss: 0.6931
- val_accuracy: 0.7372
Epoch 41/100
47/47 [==============================] - 24s 511ms/step - loss: 0.6931 - accuracy: 0.7324 - val_loss: 0.6931
- val_accuracy: 0.7823
Epoch 42/100
47/47 [==============================] - 24s 502ms/step - loss: 0.6931 - accuracy: 0.7590 - val_loss: 0.6931
- val_accuracy: 0.7969
Epoch 43/100
```

```
47/47 [==============================] - 24s 501ms/step - loss: 0.6931 - accuracy: 0.7772 - val_loss: 0.6931
- val_accuracy: 0.8480
Epoch 44/100
47/47 [==============================] - 24s 501ms/step - loss: 0.6931 - accuracy: 0.7970 - val_loss: 0.6931
- val_accuracy: 0.8441
Epoch 45/100
47/47 [==============================] - 23s 498ms/step - loss: 0.6931 - accuracy: 0.8048 - val_loss: 0.6931
- val_accuracy: 0.8374
Epoch 46/100
47/47 [==============================] - 24s 506ms/step - loss: 0.6931 - accuracy: 0.8052 - val_loss: 0.6931
- val_accuracy: 0.8759
Epoch 47/100
47/47 [==============================] - 24s 511ms/step - loss: 0.6931 - accuracy: 0.8163 - val_loss: 0.6931
- val_accuracy: 0.8952
Epoch 48/100
47/47 [==============================] - 24s 504ms/step - loss: 0.6931 - accuracy: 0.8150 - val_loss: 0.6931
- val_accuracy: 0.8626
Epoch 49/100
47/47 [==============================] - 24s 513ms/step - loss: 0.6931 - accuracy: 0.8163 - val_loss: 0.6931
- val_accuracy: 0.8726
Epoch 50/100
47/47 [==============================] - 24s 505ms/step - loss: 0.6931 - accuracy: 0.8177 - val_loss: 0.6931
- val_accuracy: 0.8958
Epoch 51/100
47/47 [==============================] - 23s 492ms/step - loss: 0.6931 - accuracy: 0.8214 - val_loss: 0.6931
- val_accuracy: 0.8998
Epoch 52/100
47/47 [==============================] - 23s 494ms/step - loss: 0.6931 - accuracy: 0.8211 - val_loss: 0.6931
- val_accuracy: 0.8421
Epoch 53/100
47/47 [==============================] - 23s 497ms/step - loss: 0.6931 - accuracy: 0.8201 - val_loss: 0.6931
- val_accuracy: 0.8786
Epoch 54/100
47/47 [==============================] - 23s 489ms/step - loss: 0.6931 - accuracy: 0.8241 - val_loss: 0.6931
- val_accuracy: 0.9005
Epoch 55/100
47/47 [==============================] - 23s 495ms/step - loss: 0.6931 - accuracy: 0.8230 - val_loss: 0.6931
- val_accuracy: 0.8978
Epoch 56/100
47/47 [==============================] - 23s 498ms/step - loss: 0.6931 - accuracy: 0.8269 - val_loss: 0.6931
- val_accuracy: 0.8885
Epoch 57/100
47/47 [==============================] - 23s 496ms/step - loss: 0.6931 - accuracy: 0.8279 - val_loss: 0.6931
```

```
- val_accuracy: 0.9151
Epoch 58/100
47/47 [==============================] - 23s 492ms/step - loss: 0.6931 - accuracy: 0.8294 - val_loss: 0.6931
- val_accuracy: 0.9104
Epoch 59/100
47/47 [==============================] - 24s 502ms/step - loss: 0.6931 - accuracy: 0.8253 - val_loss: 0.6931
- val_accuracy: 0.9171
Epoch 60/100
47/47 [==============================] - 23s 492ms/step - loss: 0.6931 - accuracy: 0.8308 - val_loss: 0.6931
- val_accuracy: 0.9230
Epoch 61/100
47/47 [==============================] - 23s 496ms/step - loss: 0.6931 - accuracy: 0.8248 - val_loss: 0.6931
- val_accuracy: 0.9363
Epoch 62/100
47/47 [==============================] - 23s 496ms/step - loss: 0.6931 - accuracy: 0.8308 - val_loss: 0.6931
- val_accuracy: 0.9217
Epoch 63/100
47/47 [==============================] - 23s 500ms/step - loss: 0.6931 - accuracy: 0.8269 - val_loss: 0.6931
- val_accuracy: 0.8752
Epoch 64/100
47/47 [==============================] - 23s 499ms/step - loss: 0.6931 - accuracy: 0.8189 - val_loss: 0.6931
- val_accuracy: 0.9403
Epoch 65/100
47/47 [==============================] - 24s 503ms/step - loss: 0.6931 - accuracy: 0.8364 - val_loss: 0.6931
- val_accuracy: 0.9237
Epoch 66/100
47/47 [==============================] - 23s 494ms/step - loss: 0.6931 - accuracy: 0.8318 - val_loss: 0.6931
- val_accuracy: 0.9310
Epoch 67/100
47/47 [==============================] - 23s 495ms/step - loss: 0.6931 - accuracy: 0.8282 - val_loss: 0.6931
- val_accuracy: 0.8726
Epoch 68/100
47/47 [==============================] - 23s 498ms/step - loss: 0.6931 - accuracy: 0.8342 - val_loss: 0.6931
- val_accuracy: 0.9297
Epoch 69/100
47/47 [==============================] - 23s 497ms/step - loss: 0.6931 - accuracy: 0.8260 - val_loss: 0.6931
- val_accuracy: 0.9482
Epoch 70/100
47/47 [==============================] - 23s 489ms/step - loss: 0.6931 - accuracy: 0.8386 - val_loss: 0.6931
- val_accuracy: 0.8467
Epoch 71/100
47/47 [==============================] - 24s 506ms/step - loss: 0.6931 - accuracy: 0.8342 - val_loss: 0.6931
- val_accuracy: 0.9456
```

```
Epoch 72/100
47/47 [==============================] - 23s 494ms/step - loss: 0.6931 - accuracy: 0.8314 - val_loss: 0.6931
- val_accuracy: 0.9429
Epoch 73/100
47/47 [==============================] - 23s 493ms/step - loss: 0.6931 - accuracy: 0.8342 - val_loss: 0.6931
- val_accuracy: 0.9529
Epoch 74/100
47/47 [==============================] - 23s 499ms/step - loss: 0.6931 - accuracy: 0.8314 - val_loss: 0.6931
- val_accuracy: 0.9429
Epoch 75/100
47/47 [==============================] - 23s 493ms/step - loss: 0.6931 - accuracy: 0.8286 - val_loss: 0.6931
- val_accuracy: 0.9396
Epoch 76/100
47/47 [==============================] - 23s 499ms/step - loss: 0.6931 - accuracy: 0.8418 - val_loss: 0.6931
- val_accuracy: 0.9463
Epoch 77/100
47/47 [==============================] - 24s 503ms/step - loss: 0.6931 - accuracy: 0.8311 - val_loss: 0.6931
- val_accuracy: 0.9701
Epoch 78/100
47/47 [==============================] - 23s 492ms/step - loss: 0.6931 - accuracy: 0.8304 - val_loss: 0.6931
- val_accuracy: 0.9356
Epoch 79/100
47/47 [==============================] - 23s 495ms/step - loss: 0.6931 - accuracy: 0.8353 - val_loss: 0.6931
- val_accuracy: 0.9681
Epoch 80/100
47/47 [==============================] - 23s 496ms/step - loss: 0.6931 - accuracy: 0.8248 - val_loss: 0.6931
- val_accuracy: 0.9403
Epoch 81/100
47/47 [==============================] - 23s 491ms/step - loss: 0.6931 - accuracy: 0.8333 - val_loss: 0.6931
- val_accuracy: 0.9781
Epoch 82/100
47/47 [==============================] - 23s 498ms/step - loss: 0.6931 - accuracy: 0.8318 - val_loss: 0.6931
- val_accuracy: 0.9721
Epoch 83/100
47/47 [==============================] - 24s 501ms/step - loss: 0.6931 - accuracy: 0.8336 - val_loss: 0.6931
- val_accuracy: 0.9814
Epoch 84/100
47/47 [==============================] - 23s 492ms/step - loss: 0.6931 - accuracy: 0.8365 - val_loss: 0.6931
- val_accuracy: 0.9668
Epoch 85/100
47/47 [==============================] - 23s 494ms/step - loss: 0.6931 - accuracy: 0.8345 - val_loss: 0.6931
- val_accuracy: 0.9595
Epoch 86/100
```

```
47/47 [==============================] - 23s 499ms/step - loss: 0.6931 - accuracy: 0.8331 - val_loss: 0.6931
- val_accuracy: 0.9509
Epoch 87/100
47/47 [==============================] - 23s 498ms/step - loss: 0.6931 - accuracy: 0.8345 - val_loss: 0.6931
- val_accuracy: 0.9317
Epoch 88/100
47/47 [==============================] - 23s 492ms/step - loss: 0.6931 - accuracy: 0.8336 - val_loss: 0.6931
- val_accuracy: 0.9814
Epoch 89/100
47/47 [==============================] - 24s 505ms/step - loss: 0.6931 - accuracy: 0.8331 - val_loss: 0.6931
- val_accuracy: 0.9834
Epoch 90/100
47/47 [==============================] - 23s 493ms/step - loss: 0.6931 - accuracy: 0.8360 - val_loss: 0.6931
- val_accuracy: 0.9754
Epoch 91/100
47/47 [==============================] - 23s 495ms/step - loss: 0.6931 - accuracy: 0.8379 - val_loss: 0.6931
- val_accuracy: 0.9721
Epoch 92/100
47/47 [==============================] - 23s 495ms/step - loss: 0.6931 - accuracy: 0.8348 - val_loss: 0.6931
- val_accuracy: 0.9854
Epoch 93/100
47/47 [==============================] - 23s 494ms/step - loss: 0.6931 - accuracy: 0.8350 - val_loss: 0.6931
- val_accuracy: 0.9741
Epoch 94/100
47/47 [==============================] - 23s 491ms/step - loss: 0.6931 - accuracy: 0.8245 - val_loss: 0.6931
- val_accuracy: 0.9827
Epoch 95/100
47/47 [==============================] - 24s 506ms/step - loss: 0.6931 - accuracy: 0.8335 - val_loss: 0.6931
- val_accuracy: 0.9834
Epoch 96/100
47/47 [==============================] - 23s 488ms/step - loss: 0.6931 - accuracy: 0.8421 - val_loss: 0.6931
- val_accuracy: 0.9774
Epoch 97/100
47/47 [==============================] - 23s 496ms/step - loss: 0.6931 - accuracy: 0.8305 - val_loss: 0.6931
- val_accuracy: 0.9834
Epoch 98/100
47/47 [==============================] - 24s 509ms/step - loss: 0.6931 - accuracy: 0.8384 - val_loss: 0.6931
- val_accuracy: 0.9761
Epoch 99/100
47/47 [==============================] - 23s 490ms/step - loss: 0.6931 - accuracy: 0.8304 - val_loss: 0.6931
- val_accuracy: 0.9801
Epoch 100/100
```

```
47/47 [==============================] - 23s 490ms/step - loss: 0.6931 - accuracy: 0.8387 - val_loss: 0.6931
- val_accuracy: 0.9781
```

In [40]:
```
lbclasses=['covid','normal']
lbclasses
```

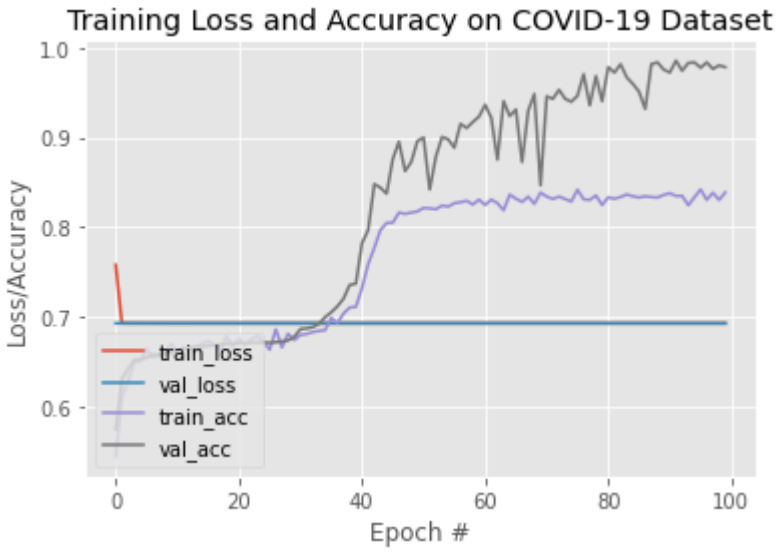
Out[40]:
```
['covid', 'normal']
```

In [41]:
```
# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
        target_names=lbclasses))

# compute the confusion matrix and and use it to derive the raw
# accuracy, sensitivity, and specificity
cm = confusion_matrix(testY.argmax(axis=1), predIdxs)
total = sum(sum(cm))
acc = (cm[0, 0] + cm[1, 1]) / total
sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])

# show the confusion matrix, accuracy, sensitivity, and specificity
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))
```

```
[INFO] evaluating network...
              precision    recall  f1-score   support

       covid       0.97      1.00      0.98      1019
      normal       1.00      0.93      0.97       488

    accuracy                           0.98      1507
   macro avg       0.98      0.97      0.97      1507
weighted avg       0.98      0.98      0.98      1507

[[1018    1]
 [  32  456]]
acc: 0.9781
sensitivity: 0.9990
specificity: 0.9344
```

In [45]:
```
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy on COVID-19 Dataset")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig('My Drive/Colab Notebooks/215-FP_Dataset/')

# serialize the model to disk
print("[INFO] saving COVID-19 detector model...")
model.save('My Drive/Colab Notebooks/215-FP_Dataset/')
```

```
[INFO] saving COVID-19 detector model...
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/resource_variable_ops.p
y:1817: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops) with constra
int is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
INFO:tensorflow:Assets written to: My Drive/Colab Notebooks/215-FP_Dataset/assets
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: