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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly (https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly)

Enter your authorization code:
..........
Mounted at /content/drive
```

In [2]:
```
!unzip '/content/drive/My Drive/Covid19Pred/Dataset_kaggle.zip'
```

```
Archive:  /content/drive/My Drive/Covid19Pred/Dataset_kaggle.zip
   creating: Dataset_kaggle/
   creating: Dataset_kaggle/COVID/
 extracting: Dataset_kaggle/COVID/Covid (1).png
  inflating: Dataset_kaggle/COVID/Covid (10).png
  inflating: Dataset_kaggle/COVID/Covid (100).png
  inflating: Dataset_kaggle/COVID/Covid (1000).png
  inflating: Dataset_kaggle/COVID/Covid (1001).png
  inflating: Dataset_kaggle/COVID/Covid (1002).png
  inflating: Dataset_kaggle/COVID/Covid (1003).png
  inflating: Dataset_kaggle/COVID/Covid (1004).png
  inflating: Dataset_kaggle/COVID/Covid (1005).png
  inflating: Dataset_kaggle/COVID/Covid (1006).png
  inflating: Dataset_kaggle/COVID/Covid (1007).png
  inflating: Dataset_kaggle/COVID/Covid (1008).png
  inflating: Dataset_kaggle/COVID/Covid (1009).png
  inflating: Dataset_kaggle/COVID/Covid (101).png
  inflating: Dataset_kaggle/COVID/Covid (1010).png
  inflating: Dataset_kaggle/COVID/Covid (1011).png
```

In [3]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import math
import cv2
import matplotlib.pyplot as plt
import os
import seaborn as sns
import umap
from PIL import Image
from scipy import misc
from os import listdir
from os.path import isfile, join
import numpy as np
from scipy import misc
from random import shuffle
from collections import Counter
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.utils.np_utils import to_categorical
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
Using TensorFlow backend.
```

In [7]:
```
yes=os.listdir('/content/Dataset_kaggle/COVID')
no=os.listdir('/content/Dataset_kaggle/non-COVID')
```

In [8]:
```
data=np.concatenate([yes,no])
len(data)==len(yes)+len(no)
```

Out[8]:
```
True
```

In [9]:
```
target_x=np.full(len(yes),1)
target_y=np.full(len(no),0)
data_target=np.concatenate([target_x,target_y])
```

In [10]:
```
data_target
```

Out[10]:
```
array([1, 1, 1, ..., 0, 0, 0])
```

In [11]:
```
data
```

Out[11]:
```
array(['Covid (1024).png', 'Covid (1213).png', 'Covid (477).png', ...,
       'Non-Covid (511).png', 'Non-Covid (285).png',
       'Non-Covid (805).png'], dtype='<U20')
```

In [12]:
```
yes_values=os.listdir('/content/Dataset_kaggle/COVID')
no_values=os.listdir('/content/Dataset_kaggle/non-COVID')
```

In [15]:
```
X_data =[]
for file in yes_values:
    #face = misc.imread('../input/brain_tumor_dataset/yes/'+file)
    img = cv2.imread('/content/Dataset_kaggle/COVID/'+file)
    face = cv2.resize(img, (32, 32) )
    (b, g, r)=cv2.split(face)
    img=cv2.merge([r,g,b])
    X_data.append(img)
```

In [16]:
```
for file in no_values:
    #face = misc.imread('../input/brain_tumor_dataset/yes/'+file)
    img = cv2.imread('/content/Dataset_kaggle/non-COVID/'+file)
    face = cv2.resize(img, (32, 32) )
    (b, g, r)=cv2.split(face)
    img=cv2.merge([r,g,b])
    X_data.append(img)
```

In [16]:
```

```

In [17]:
```
X = np.squeeze(X_data)
```

In [18]:
```
# normalize data
X = X.astype('float32')
X /= 255
```

In [19]:
```
data_target
```

Out[19]:
```
array([1, 1, 1, ..., 0, 0, 0])
```

In [20]:
```
len(data)
```

Out[20]:
```
2481
```

In [21]:
```
from sklearn.model_selection import train_test_split
```

In [22]:
```
x_train,x_test,y_train,y_test=train_test_split(X, data_target, test_size=0.2, ra
```

In [ ]:
```
y_test
```

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

In [23]:
```
model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=16,kernel_size=9, padding='same', activa
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.45))

model.add(tf.keras.layers.Conv2D(filters=16,kernel_size=9,padding='same', activat
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.25))

model.add(tf.keras.layers.Conv2D(filters=36, kernel_size=9, padding='same', activ
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.25))

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

model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dropout(0.15))
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dropout(0.15))
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dropout(0.15))
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dropout(0.15))
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

# Take a look at the model summary
model.summary()
#tf.keras.utils.plot_model(model, to_file='model_plot.png', show_shapes=True, sho
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 16)        3904
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 16)        20752
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 16)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 16)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 36)          46692
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 36)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 36)          0
_________________________________________________________________
```

```
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 512)               295424
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_5 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_6 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 513
=================================================================
Total params: 1,943,221
Trainable params: 1,943,221
Non-trainable params: 0
_________________________________________________________________
```

In [24]:

```
model.compile(loss='binary_crossentropy',
              optimizer=tf.keras.optimizers.Adam(),
              metrics=['acc'])
```

In [25]:
```
model.fit(x_train,
          y_train,
          batch_size=128,
          epochs=1000,
          validation_data=(x_test, y_test))
```

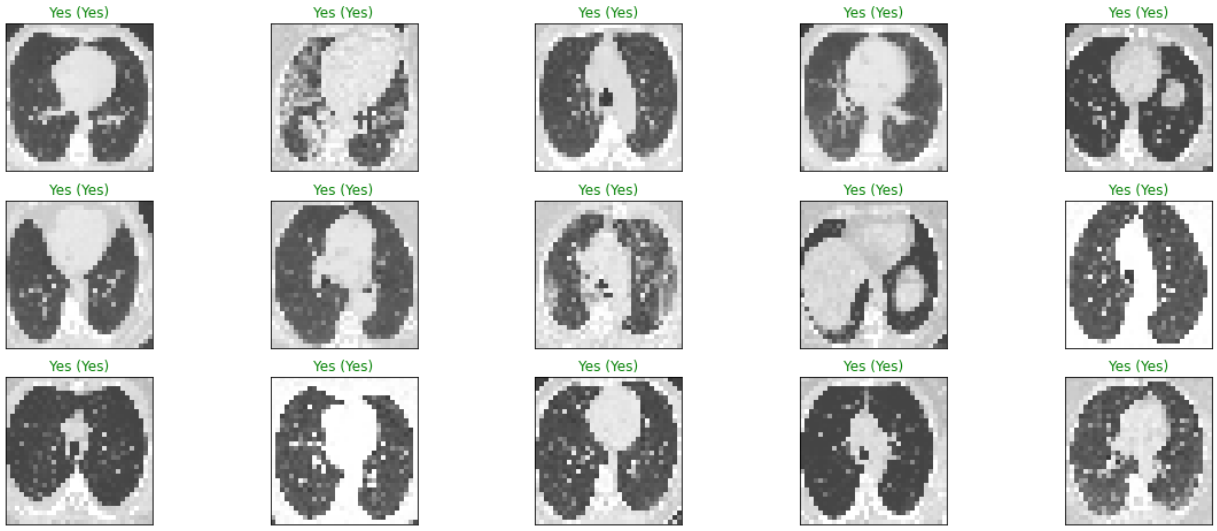
```
Epoch 1/1000
16/16 [==============================] - 1s 32ms/step - loss: 0.6959 - acc:
0.4844 - val_loss: 0.6932 - val_acc: 0.5010
Epoch 2/1000
16/16 [==============================] - 0s 9ms/step - loss: 0.6932 - acc: 0.
5055 - val_loss: 0.6932 - val_acc: 0.5010
Epoch 3/1000
16/16 [==============================] - 0s 9ms/step - loss: 0.6932 - acc: 0.
5030 - val_loss: 0.6931 - val_acc: 0.5010
Epoch 4/1000
16/16 [==============================] - 0s 9ms/step - loss: 0.6935 - acc: 0.
4919 - val_loss: 0.6931 - val_acc: 0.5010
Epoch 5/1000
16/16 [==============================] - 0s 9ms/step - loss: 0.6932 - acc: 0.
5055 - val_loss: 0.6933 - val_acc: 0.5010
Epoch 6/1000
16/16 [==============================] - 0s 9ms/step - loss: 0.6932 - acc: 0.
5055 - val_loss: 0.6930 - val_acc: 0.5010
Epoch 7/1000
```

In [26]:
```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score[1])
```

```
 Test accuracy: 0.9657947421073914
```

In [27]:
```
y_hat = model.predict(x_test)
labels =["Yes",  # index 0
        "No",      # index 1
        ]
# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[2], size=15, replace=Fals
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks=[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                                  color=("green" if predict_index == true_index
plt.show()
```




In [ ]:
```
from keras.models import load_model
model.save('CoronaV3.h5')
```

In [ ]:
```
```