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

Enter your authorization code:

··········

Mounted at /content/drive

In [4]:

cd /content/drive/My Drive/My\_Projects/Face\_mask

/content/drive/My Drive/My\_Projects/Face\_mask

**Data Preprocessing**

In [6]:

**import** **cv2**,**os**

data\_path='/content/drive/My Drive/My\_Projects/Face\_mask/mask\_datasets/'

categories=os.listdir(data\_path)

labels=[i **for** i **in** range(len(categories))]

label\_dict=dict(zip(categories,labels)) *#empty dictionary*

print(label\_dict)

print(categories)

print(labels)

{'with\_mask': 0, 'without\_mask': 1}

['with\_mask', 'without\_mask']

[0, 1]

In [0]:

img\_size=100

data=[]

target=[]

**for** category **in** categories:

folder\_path=os.path.join(data\_path,category)

img\_names=os.listdir(folder\_path)

**for** img\_name **in** img\_names:

img\_path=os.path.join(folder\_path,img\_name)

img=cv2.imread(img\_path)

**try**:

gray=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

*#Coverting the image into gray scale*

resized=cv2.resize(gray,(img\_size,img\_size))

*#resizing the gray scale into 100x100, since we need a fixed common size for all the images in the dataset*

data.append(resized)

target.append(label\_dict[category])

*#appending the image and the label(categorized) into the list (dataset)*

**except** **Exception** **as** e:

print('Exception:',e)

*#if any exception rasied, the exception will be printed here. And pass to the next image*

**Recale and assign catagorical lables**

In [8]:

**import** **numpy** **as** **np**

data=np.array(data)/255.0

data=np.reshape(data,(data.shape[0],img\_size,img\_size,1))

target=np.array(target)

**from** **keras.utils** **import** np\_utils

new\_target=np\_utils.to\_categorical(target)

Using TensorFlow backend.

In [9]:

new\_target.shape

Out[9]:

(1376, 2)

In [0]:

np.save('images.npy',data)

np.save('lables.npy',new\_target)

**CNN Model**

In [0]:

**import** **numpy** **as** **np**

data=np.load('images.npy')

new\_target=np.load('labels.npy')

In [16]:

data.shape

Out[16]:

(1376, 100, 100, 1)

In [17]:

data.shape[1:]

Out[17]:

(100, 100, 1)

In [0]:

**from** **keras.models** **import** Sequential

**from** **keras.layers** **import** Dense,Activation,Flatten,Dropout

**from** **keras.layers** **import** Conv2D,MaxPooling2D

**from** **keras.callbacks** **import** ModelCheckpoint

model=Sequential()

model.add(Conv2D(200,(3,3),input\_shape=data.shape[1:]))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

*#The first CNN layer followed by Relu and MaxPooling layers*

model.add(Conv2D(100,(3,3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

*#The second convolution layer followed by Relu and MaxPooling layers*

model.add(Flatten())

model.add(Dropout(0.5))

*#Flatten layer to stack the output convolutions from second convolution layer*

model.add(Dense(50,activation='relu'))

*#Dense layer of 64 neurons*

model.add(Dense(2,activation='softmax'))

*#The Final layer with two outputs for two categories*

model.compile(loss='categorical\_crossentropy',optimizer='adam',metrics=['accuracy'])

In [19]:

model.summary()

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_1 (Conv2D) (None, 98, 98, 200) 2000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_1 (Activation) (None, 98, 98, 200) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 49, 49, 200) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D) (None, 47, 47, 100) 180100

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_2 (Activation) (None, 47, 47, 100) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 23, 23, 100) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 52900) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 52900) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 50) 2645050

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 2) 102

=================================================================

Total params: 2,827,252

Trainable params: 2,827,252

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Splittiong data into traning and testing**

In [0]:

**from** **sklearn.model\_selection** **import** train\_test\_split

train\_data,test\_data,train\_target,test\_target=train\_test\_split(data,new\_target,test\_size=0.1)

In [21]:

train\_data.shape

Out[21]:

(1238, 100, 100, 1)

In [22]:

train\_target.shape

Out[22]:

(1238, 2)

In [23]:

checkpoint = ModelCheckpoint('model-**{epoch:03d}**.model',monitor='val\_loss',verbose=0,save\_best\_only=**True**,mode='auto')

history=model.fit(train\_data,train\_target,epochs=100,callbacks=[checkpoint],validation\_split=0.2)

Train on 990 samples, validate on 248 samples

Epoch 1/100

990/990 [==============================] - 9s 9ms/step - loss: 0.6743 - accuracy: 0.5707 - val\_loss: 0.6390 - val\_accuracy: 0.6048

Epoch 2/100

990/990 [==============================] - 2s 2ms/step - loss: 0.5127 - accuracy: 0.7556 - val\_loss: 0.3646 - val\_accuracy: 0.8629

Epoch 3/100

990/990 [==============================] - 2s 2ms/step - loss: 0.3018 - accuracy: 0.8889 - val\_loss: 0.1824 - val\_accuracy: 0.9476

Epoch 4/100

990/990 [==============================] - 2s 2ms/step - loss: 0.1769 - accuracy: 0.9384 - val\_loss: 0.1490 - val\_accuracy: 0.9556

Epoch 5/100

990/990 [==============================] - 2s 2ms/step - loss: 0.1485 - accuracy: 0.9414 - val\_loss: 0.1306 - val\_accuracy: 0.9677

Epoch 6/100

990/990 [==============================] - 2s 2ms/step - loss: 0.1079 - accuracy: 0.9616 - val\_loss: 0.1636 - val\_accuracy: 0.9153

Epoch 7/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0793 - accuracy: 0.9768 - val\_loss: 0.1224 - val\_accuracy: 0.9476

Epoch 8/100

990/990 [==============================] - 2s 2ms/step - loss: 0.1038 - accuracy: 0.9606 - val\_loss: 0.2070 - val\_accuracy: 0.9274

Epoch 9/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0635 - accuracy: 0.9758 - val\_loss: 0.1061 - val\_accuracy: 0.9556

Epoch 10/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0629 - accuracy: 0.9778 - val\_loss: 0.1732 - val\_accuracy: 0.9556

Epoch 11/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0603 - accuracy: 0.9778 - val\_loss: 0.1178 - val\_accuracy: 0.9637

Epoch 12/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0434 - accuracy: 0.9798 - val\_loss: 0.1127 - val\_accuracy: 0.9637

Epoch 13/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0258 - accuracy: 0.9919 - val\_loss: 0.1474 - val\_accuracy: 0.9556

Epoch 14/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0254 - accuracy: 0.9889 - val\_loss: 0.1217 - val\_accuracy: 0.9476

Epoch 15/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0864 - accuracy: 0.9677 - val\_loss: 0.4138 - val\_accuracy: 0.9032

Epoch 16/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0999 - accuracy: 0.9616 - val\_loss: 0.1160 - val\_accuracy: 0.9677

Epoch 17/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0312 - accuracy: 0.9929 - val\_loss: 0.1191 - val\_accuracy: 0.9556

Epoch 18/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0254 - accuracy: 0.9960 - val\_loss: 0.1672 - val\_accuracy: 0.9556

Epoch 19/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0274 - accuracy: 0.9889 - val\_loss: 0.1264 - val\_accuracy: 0.9637

Epoch 20/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0369 - accuracy: 0.9909 - val\_loss: 0.1353 - val\_accuracy: 0.9597

Epoch 21/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0163 - accuracy: 0.9960 - val\_loss: 0.1400 - val\_accuracy: 0.9718

Epoch 22/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0133 - accuracy: 0.9939 - val\_loss: 0.1370 - val\_accuracy: 0.9476

Epoch 23/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0123 - accuracy: 0.9990 - val\_loss: 0.1136 - val\_accuracy: 0.9637

Epoch 24/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0142 - accuracy: 0.9949 - val\_loss: 0.1821 - val\_accuracy: 0.9315

Epoch 25/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0324 - accuracy: 0.9899 - val\_loss: 0.1504 - val\_accuracy: 0.9476

Epoch 26/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0227 - accuracy: 0.9909 - val\_loss: 0.1186 - val\_accuracy: 0.9597

Epoch 27/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0108 - accuracy: 0.9970 - val\_loss: 0.1409 - val\_accuracy: 0.9637

Epoch 28/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0096 - accuracy: 0.9980 - val\_loss: 0.1558 - val\_accuracy: 0.9516

Epoch 29/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0097 - accuracy: 0.9949 - val\_loss: 0.1014 - val\_accuracy: 0.9718

Epoch 30/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0160 - accuracy: 0.9949 - val\_loss: 0.1099 - val\_accuracy: 0.9677

Epoch 31/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0137 - accuracy: 0.9949 - val\_loss: 0.1341 - val\_accuracy: 0.9677

Epoch 32/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0048 - accuracy: 0.9990 - val\_loss: 0.1462 - val\_accuracy: 0.9677

Epoch 33/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0127 - accuracy: 0.9970 - val\_loss: 0.3205 - val\_accuracy: 0.9315

Epoch 34/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0896 - accuracy: 0.9646 - val\_loss: 0.1196 - val\_accuracy: 0.9718

Epoch 35/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0402 - accuracy: 0.9869 - val\_loss: 0.0939 - val\_accuracy: 0.9758

Epoch 36/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0247 - accuracy: 0.9929 - val\_loss: 0.1074 - val\_accuracy: 0.9758

Epoch 37/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0129 - accuracy: 0.9949 - val\_loss: 0.1251 - val\_accuracy: 0.9637

Epoch 38/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0076 - accuracy: 0.9980 - val\_loss: 0.1939 - val\_accuracy: 0.9556

Epoch 39/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0150 - accuracy: 0.9949 - val\_loss: 0.0886 - val\_accuracy: 0.9758

Epoch 40/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0101 - accuracy: 0.9960 - val\_loss: 0.1263 - val\_accuracy: 0.9758

Epoch 41/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0065 - accuracy: 0.9990 - val\_loss: 0.1205 - val\_accuracy: 0.9677

Epoch 42/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0039 - accuracy: 0.9990 - val\_loss: 0.1101 - val\_accuracy: 0.9718

Epoch 43/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0162 - accuracy: 0.9949 - val\_loss: 0.1766 - val\_accuracy: 0.9677

Epoch 44/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0071 - accuracy: 0.9990 - val\_loss: 0.1551 - val\_accuracy: 0.9677

Epoch 45/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0072 - accuracy: 0.9980 - val\_loss: 0.1508 - val\_accuracy: 0.9597

Epoch 46/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0076 - accuracy: 0.9980 - val\_loss: 0.1619 - val\_accuracy: 0.9718

Epoch 47/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0037 - accuracy: 0.9990 - val\_loss: 0.1781 - val\_accuracy: 0.9597

Epoch 48/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0021 - accuracy: 1.0000 - val\_loss: 0.1315 - val\_accuracy: 0.9677

Epoch 49/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0015 - accuracy: 0.9990 - val\_loss: 0.1480 - val\_accuracy: 0.9637

Epoch 50/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0012 - accuracy: 1.0000 - val\_loss: 0.1353 - val\_accuracy: 0.9718

Epoch 51/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0025 - accuracy: 0.9990 - val\_loss: 0.1615 - val\_accuracy: 0.9637

Epoch 52/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0013 - accuracy: 1.0000 - val\_loss: 0.1413 - val\_accuracy: 0.9597

Epoch 53/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0042 - accuracy: 0.9990 - val\_loss: 0.1552 - val\_accuracy: 0.9597

Epoch 54/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0079 - accuracy: 0.9960 - val\_loss: 0.2626 - val\_accuracy: 0.9556

Epoch 55/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0202 - accuracy: 0.9909 - val\_loss: 0.1812 - val\_accuracy: 0.9597

Epoch 56/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0094 - accuracy: 0.9990 - val\_loss: 0.1339 - val\_accuracy: 0.9677

Epoch 57/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0043 - accuracy: 0.9990 - val\_loss: 0.1181 - val\_accuracy: 0.9637

Epoch 58/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0029 - accuracy: 1.0000 - val\_loss: 0.1375 - val\_accuracy: 0.9637

Epoch 59/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0105 - accuracy: 0.9960 - val\_loss: 0.1691 - val\_accuracy: 0.9556

Epoch 60/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0197 - accuracy: 0.9919 - val\_loss: 0.1675 - val\_accuracy: 0.9556

Epoch 61/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0092 - accuracy: 0.9970 - val\_loss: 0.1695 - val\_accuracy: 0.9516

Epoch 62/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0082 - accuracy: 0.9970 - val\_loss: 0.1271 - val\_accuracy: 0.9718

Epoch 63/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0033 - accuracy: 1.0000 - val\_loss: 0.0904 - val\_accuracy: 0.9798

Epoch 64/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0100 - accuracy: 0.9980 - val\_loss: 0.2116 - val\_accuracy: 0.9637

Epoch 65/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0548 - accuracy: 0.9808 - val\_loss: 0.2410 - val\_accuracy: 0.9556

Epoch 66/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0309 - accuracy: 0.9879 - val\_loss: 0.2445 - val\_accuracy: 0.9435

Epoch 67/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0399 - accuracy: 0.9889 - val\_loss: 0.1871 - val\_accuracy: 0.9556

Epoch 68/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0161 - accuracy: 0.9939 - val\_loss: 0.1315 - val\_accuracy: 0.9637

Epoch 69/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0075 - accuracy: 0.9980 - val\_loss: 0.1441 - val\_accuracy: 0.9718

Epoch 70/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0293 - accuracy: 0.9919 - val\_loss: 0.1240 - val\_accuracy: 0.9718

Epoch 71/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0116 - accuracy: 0.9970 - val\_loss: 0.1374 - val\_accuracy: 0.9597

Epoch 72/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0133 - accuracy: 0.9949 - val\_loss: 0.1642 - val\_accuracy: 0.9677

Epoch 73/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0087 - accuracy: 0.9980 - val\_loss: 0.0924 - val\_accuracy: 0.9718

Epoch 74/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0013 - accuracy: 1.0000 - val\_loss: 0.1075 - val\_accuracy: 0.9677

Epoch 75/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0028 - accuracy: 0.9990 - val\_loss: 0.1163 - val\_accuracy: 0.9677

Epoch 76/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0011 - accuracy: 1.0000 - val\_loss: 0.1343 - val\_accuracy: 0.9677

Epoch 77/100

990/990 [==============================] - 2s 2ms/step - loss: 4.4373e-04 - accuracy: 1.0000 - val\_loss: 0.1335 - val\_accuracy: 0.9677

Epoch 78/100

990/990 [==============================] - 2s 2ms/step - loss: 8.7239e-04 - accuracy: 1.0000 - val\_loss: 0.1223 - val\_accuracy: 0.9677

Epoch 79/100

990/990 [==============================] - 2s 2ms/step - loss: 7.3825e-04 - accuracy: 1.0000 - val\_loss: 0.1157 - val\_accuracy: 0.9718

Epoch 80/100

990/990 [==============================] - 2s 2ms/step - loss: 2.4482e-04 - accuracy: 1.0000 - val\_loss: 0.1184 - val\_accuracy: 0.9718

Epoch 81/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0031 - accuracy: 0.9990 - val\_loss: 0.1089 - val\_accuracy: 0.9637

Epoch 82/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0062 - accuracy: 0.9970 - val\_loss: 0.1269 - val\_accuracy: 0.9677

Epoch 83/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0012 - accuracy: 1.0000 - val\_loss: 0.1108 - val\_accuracy: 0.9637

Epoch 84/100

990/990 [==============================] - 2s 2ms/step - loss: 9.5985e-04 - accuracy: 1.0000 - val\_loss: 0.1042 - val\_accuracy: 0.9758

Epoch 85/100

990/990 [==============================] - 2s 2ms/step - loss: 5.4308e-04 - accuracy: 1.0000 - val\_loss: 0.1019 - val\_accuracy: 0.9758

Epoch 86/100

990/990 [==============================] - 2s 2ms/step - loss: 2.4700e-04 - accuracy: 1.0000 - val\_loss: 0.1055 - val\_accuracy: 0.9758

Epoch 87/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0044 - accuracy: 0.9980 - val\_loss: 0.1083 - val\_accuracy: 0.9758

Epoch 88/100

990/990 [==============================] - 2s 2ms/step - loss: 8.7106e-04 - accuracy: 1.0000 - val\_loss: 0.1490 - val\_accuracy: 0.9637

Epoch 89/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0012 - accuracy: 0.9990 - val\_loss: 0.1536 - val\_accuracy: 0.9677

Epoch 90/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0021 - accuracy: 0.9990 - val\_loss: 0.0874 - val\_accuracy: 0.9758

Epoch 91/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0012 - accuracy: 0.9990 - val\_loss: 0.1621 - val\_accuracy: 0.9718

Epoch 92/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0021 - accuracy: 0.9990 - val\_loss: 0.1410 - val\_accuracy: 0.9718

Epoch 93/100

990/990 [==============================] - 2s 2ms/step - loss: 0.0014 - accuracy: 1.0000 - val\_loss: 0.1366 - val\_accuracy: 0.9718

Epoch 94/100

990/990 [==============================] - 2s 2ms/step - loss: 3.8981e-04 - accuracy: 1.0000 - val\_loss: 0.1226 - val\_accuracy: 0.9718

Epoch 95/100

990/990 [==============================] - 2s 2ms/step - loss: 1.5824e-04 - accuracy: 1.0000 - val\_loss: 0.1352 - val\_accuracy: 0.9758

Epoch 96/100

990/990 [==============================] - 2s 2ms/step - loss: 2.2961e-04 - accuracy: 1.0000 - val\_loss: 0.1250 - val\_accuracy: 0.9758

Epoch 97/100

990/990 [==============================] - 2s 2ms/step - loss: 2.3081e-04 - accuracy: 1.0000 - val\_loss: 0.1555 - val\_accuracy: 0.9718

Epoch 98/100

990/990 [==============================] - 2s 2ms/step - loss: 9.4780e-05 - accuracy: 1.0000 - val\_loss: 0.1434 - val\_accuracy: 0.9758

Epoch 99/100

990/990 [==============================] - 2s 2ms/step - loss: 1.2239e-04 - accuracy: 1.0000 - val\_loss: 0.1462 - val\_accuracy: 0.9758

Epoch 100/100

990/990 [==============================] - 2s 2ms/step - loss: 1.0842e-04 - accuracy: 1.0000 - val\_loss: 0.1511 - val\_accuracy: 0.9758

In [0]:

**from** **matplotlib** **import** pyplot **as** plt

In [25]:

*# plot the training loss and accuracy*

N = 100

plt.style.use("ggplot")

plt.figure()

plt.plot(np.arange(0, N), history.history["loss"], label="train\_loss")

plt.plot(np.arange(0, N), history.history["val\_loss"], label="val\_loss")

plt.plot(np.arange(0, N), history.history["accuracy"], label="train\_acc")

plt.plot(np.arange(0, N), history.history["val\_accuracy"], label="val\_acc")

plt.title("Training Loss and Accuracy")

plt.xlabel("Epoch #")

plt.ylabel("Loss/Accuracy")

plt.legend(loc="center right")

plt.savefig("CNN\_Model")

In [0]: