In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import os
import cv2
import imghdr
```

In [2]:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount,
call drive.mount("/content/gdrive", force_remount=True).
```

Loading data

In [3]:

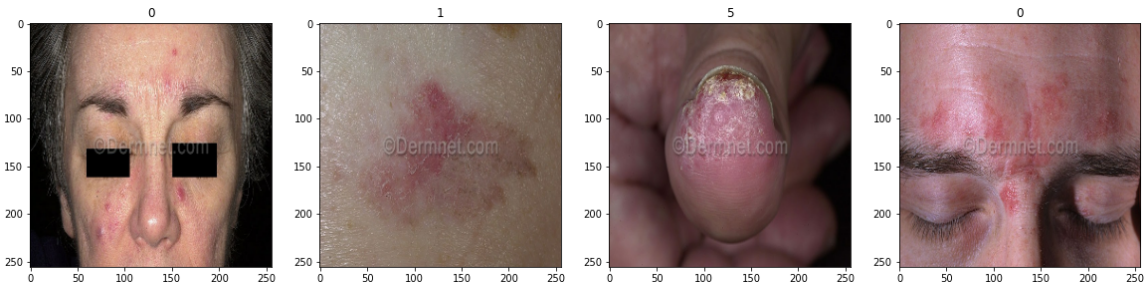
```
data_dir = '/content/gdrive/MyDrive/Data_CNN/train'
test_dir = '/content/gdrive/MyDrive/Data_CNN/test'
data = tf.keras.utils.image_dataset_from_directory('/content/gdrive/MyDrive/Data_CNN/tr
ain')
test = tf.keras.utils.image_dataset_from_directory('/content/gdrive/MyDrive/Data_CNN/te
st')
```

```
Found 5466 files belonging to 10 classes.
Found 1371 files belonging to 10 classes.
```

Pre-processing

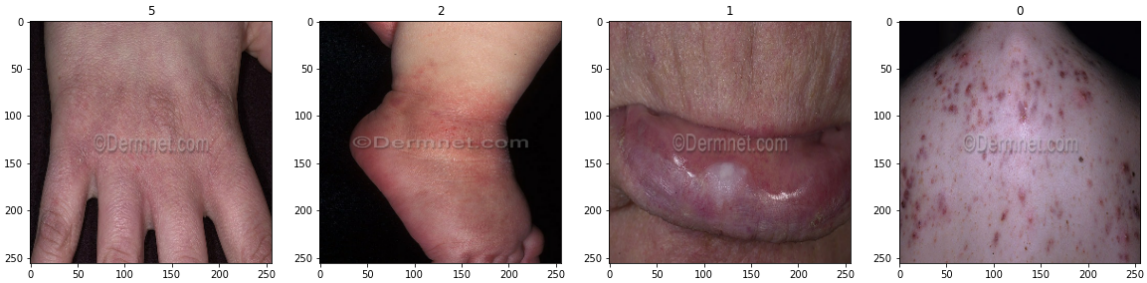
In [4]:

```
# training set
data = data.map(lambda x, y: (x/255, y))
scaled_iterator = data.as_numpy_iterator()
batch = scaled_iterator.next()
fig,ax = plt.subplots(ncols=4, figsize=(20,20))
for idx, img in enumerate(batch[0][:4]):
  ax[idx].imshow(img)
  ax[idx].title.set_text(batch[1][idx])
```



In [5]:

```
# testing set
test = test.map(lambda x, y: (x/255, y))
scaled_iterator = test.as_numpy_iterator()
batch = scaled_iterator.next()
fig,ax = plt.subplots(ncols=4, figsize=(20,20))
for idx, img in enumerate(batch[0][:4]):
  ax[idx].imshow(img)
  ax[idx].title.set_text(batch[1][idx])
```



In [6]:

```
len(data)
```

Out[6]:

```
171
```

In [14]:

```
train_size = int(len(data)*.7) + 1
val_size   = int(len(data)*.3)
train_size + val_size
```

Out[14]:

```
171
```

In [16]:

```
train = data.take(train_size)
val = data.skip(train_size).take(val_size)
print(len(train),len(val))
```

```
120 51
```

Convolutional Deep learning

In [18]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten
```

In [19]:

```
model = Sequential()
```

In [20]:

```
model.add(Conv2D(16,(3,3), 1, activation='relu', input_shape=(256,256,3)))
model.add(MaxPooling2D())

model.add(Conv2D(32,(3,3), 1, activation='relu'))
model.add(MaxPooling2D())

model.add(Conv2D(16,(3,3), 1, activation='relu'))
model.add(MaxPooling2D())

model.add(Flatten())

model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [21]:

```
model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

In [22]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 16)      448

 max_pooling2d (MaxPooling2D  (None, 127, 127, 16)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 32)      4640

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 16)        4624

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 16)       0
 2D)

 flatten (Flatten)           (None, 14400)             0

 dense (Dense)               (None, 256)               3686656

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 3,696,625
Trainable params: 3,696,625
Non-trainable params: 0
_________________________________________________________________
```

Train

In [23]:

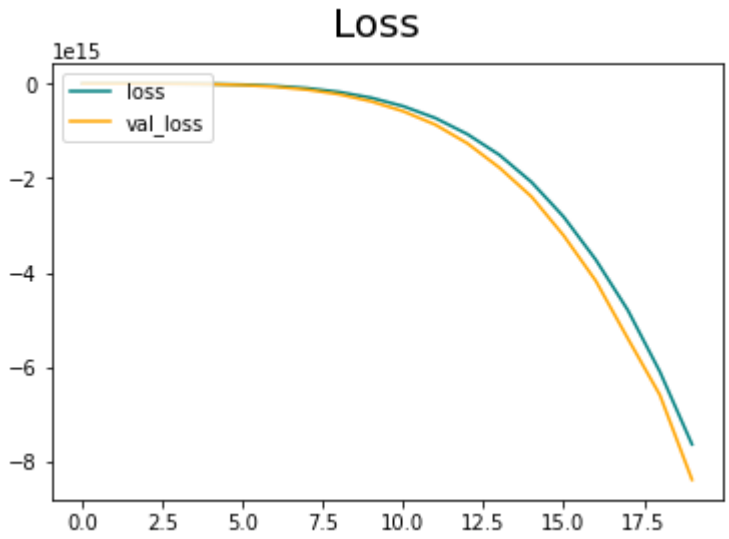
```
logdir = 'logs'
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
hist = model.fit(train, epochs=20, validation_data=val, callbacks=[tensorboard_callback
])
```

```
Epoch 1/20
120/120 [==============================] - 239s 2s/step - loss: -81948848.
0000 - accuracy: 0.2099 - val_loss: -644975744.0000 - val_accuracy: 0.2097
Epoch 2/20
120/120 [==============================] - 163s 1s/step - loss: -160632391
68.0000 - accuracy: 0.2096 - val_loss: -59377172480.0000 - val_accuracy:
0.2073
Epoch 3/20
120/120 [==============================] - 164s 1s/step - loss: -272778608
640.0000 - accuracy: 0.2096 - val_loss: -662882353152.0000 - val_accuracy:
0.2079
Epoch 4/20
120/120 [==============================] - 166s 1s/step - loss: -178006025
8304.0000 - accuracy: 0.2083 - val_loss: -3490529411072.0000 - val_accurac
y: 0.2066
Epoch 5/20
120/120 [==============================] - 154s 1s/step - loss: -696620364
5952.0000 - accuracy: 0.2086 - val_loss: -11593893019648.0000 - val_accura
cy: 0.2017
Epoch 6/20
120/120 [==============================] - 167s 1s/step - loss: -196774199
29600.0000 - accuracy: 0.2096 - val_loss: -29757198565376.0000 - val_accur
acy: 0.2048
Epoch 7/20
120/120 [==============================] - 166s 1s/step - loss: -465313611
12064.0000 - accuracy: 0.2109 - val_loss: -65895091142656.0000 - val_accur
acy: 0.2097
Epoch 8/20
120/120 [==============================] - 169s 1s/step - loss: -951841508
55680.0000 - accuracy: 0.2115 - val_loss: -128089967296512.0000 - val_accu
racy: 0.2085
Epoch 9/20
120/120 [==============================] - 166s 1s/step - loss: -176650075
504640.0000 - accuracy: 0.2104 - val_loss: -225126415073280.0000 - val_acc
uracy: 0.2073
Epoch 10/20
120/120 [==============================] - 154s 1s/step - loss: -298833590
353920.0000 - accuracy: 0.2112 - val_loss: -375494964936704.0000 - val_acc
uracy: 0.2091
Epoch 11/20
120/120 [==============================] - 168s 1s/step - loss: -477780416
397312.0000 - accuracy: 0.2102 - val_loss: -584684031442944.0000 - val_acc
uracy: 0.2134
Epoch 12/20
120/120 [==============================] - 167s 1s/step - loss: -728208685
137920.0000 - accuracy: 0.2099 - val_loss: -868978519965696.0000 - val_acc
uracy: 0.2085
Epoch 13/20
120/120 [==============================] - 155s 1s/step - loss: -107005855
3999360.0000 - accuracy: 0.2107 - val_loss: -1265053760225280.0000 - val_a
ccuracy: 0.2060
Epoch 14/20
120/120 [==============================] - 167s 1s/step - loss: -151045181
6955904.0000 - accuracy: 0.2115 - val_loss: -1780488893104128.0000 - val_a
ccuracy: 0.2103
Epoch 15/20
120/120 [==============================] - 154s 1s/step - loss: -208615768
9028608.0000 - accuracy: 0.2109 - val_loss: -2395324198944768.0000 - val_a
ccuracy: 0.2103
Epoch 16/20
120/120 [==============================] - 153s 1s/step - loss: -281971797
```

```
0526208.0000 - accuracy: 0.2104 - val_loss: -3216282233077760.0000 - val_a
ccuracy: 0.2085
Epoch 17/20
120/120 [==============================] - 167s 1s/step - loss: -372729556
2907648.0000 - accuracy: 0.2099 - val_loss: -4174945508655104.0000 - val_a
ccuracy: 0.2048
Epoch 18/20
120/120 [==============================] - 154s 1s/step - loss: -479717567
4437632.0000 - accuracy: 0.2096 - val_loss: -5396796058632192.0000 - val_a
ccuracy: 0.1999
Epoch 19/20
120/120 [==============================] - 166s 1s/step - loss: -610168609
1251712.0000 - accuracy: 0.2117 - val_loss: -6594565800722432.0000 - val_a
ccuracy: 0.2165
Epoch 20/20
120/120 [==============================] - 166s 1s/step - loss: -763849142
4882688.0000 - accuracy: 0.2091 - val_loss: -8387432252702720.0000 - val_a
ccuracy: 0.2116
```

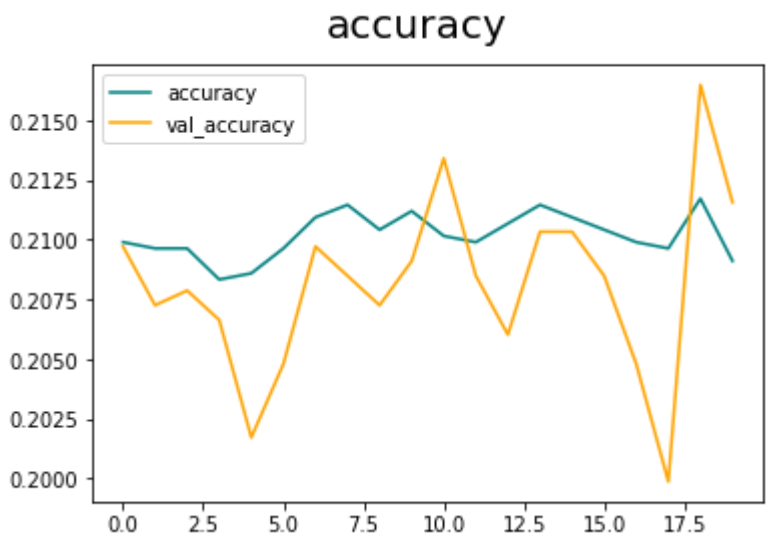
In [25]:

```
fig = plt.figure()
plt.plot(hist.history['loss'],color='teal', label='loss')
plt.plot(hist.history['val_loss'],color='orange', label='val_loss')
fig.suptitle('Loss', fontsize=20)
plt.legend(loc='upper left')
plt.show()
```



In [26]:

```
fig = plt.figure()
plt.plot(hist.history['accuracy'],color='teal', label='accuracy')
plt.plot(hist.history['val_accuracy'],color='orange', label='val_accuracy')
fig.suptitle('accuracy', fontsize=20)
plt.legend(loc='upper left')
plt.show()
```



Testing

In [27]:

```
from tensorflow.keras.metrics import Precision, Recall , BinaryAccuracy
```

In [28]:

```
pre = Precision()
re = Recall()
acc = BinaryAccuracy()
```

In [29]:

```
for batch in test.as_numpy_iterator():
  X, y = batch
  yhat = model.predict(X)
  pre.update_state(y,yhat)
  re.update_state(y, yhat)
  acc.update_state(y, yhat)
```

In [33]:

```
print("Precision:",pre.result().numpy(), "recall:",re.result().numpy(),"Accuracy:",acc.
result().numpy())
```

```
Precision: 0.7724289 recall: 1.0 Accuracy: 0.21078274
```