1. Import dependencies:

In [ ]:
```
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
from PIL import Image
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
import cv2
```

2. Loading data:

A- Loading the both data:

In [ ]:
```
data_withMask = os.listdir('data\with_mask')
data_withoutMask = os.listdir('data\without_mask')
```

B- Verify the data;

In [ ]:
```
print(data_withMask[0:5])
print(data_withoutMask[0:5])
```

```
['with_mask_1.jpg', 'with_mask_10.jpg', 'with_mask_100.jpg', 'with_mask_1000.jp
g', 'with_mask_1001.jpg']
['without_mask_1.jpg', 'without_mask_10.jpg', 'without_mask_100.jpg', 'without_ma
sk_1000.jpg', 'without_mask_1001.jpg']
```

C- Number of images:

In [ ]:
```
print(len(data_withMask))
print(len(data_withoutMask))
```

```
3725
3828
```

3. Label of images:

A- Creating the labels:

In [ ]:
```
withMask_label = [1]*3725
withoutMask_label = [0]*3828
```

In [ ]:
```
print(withMask_label[0:5])
print(withoutMask_label[0:5])
```

```
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

In [ ]:
```
print(len(withMask_label))
print(len(withoutMask_label))
```

```
3725
3828
```

B- Creatiing the label of the both datas:

In [ ]:
```
labels = withoutMask_label + withMask_label
print(len(labels))
print(labels[0:5])
print(labels[-5:])
```

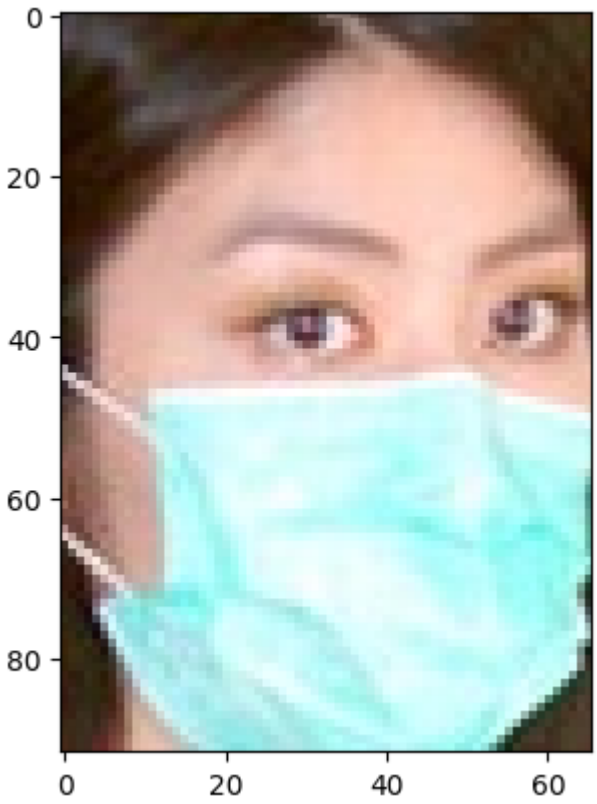
```
7553
[0, 0, 0, 0, 0]
[1, 1, 1, 1, 1]
```

4. Image preprocessing:

A- Displaying one mask images:

In [ ]:
```
img = mpimg.imread('data\with_mask\with_mask_3.jpg')
plt.imshow(img)
```

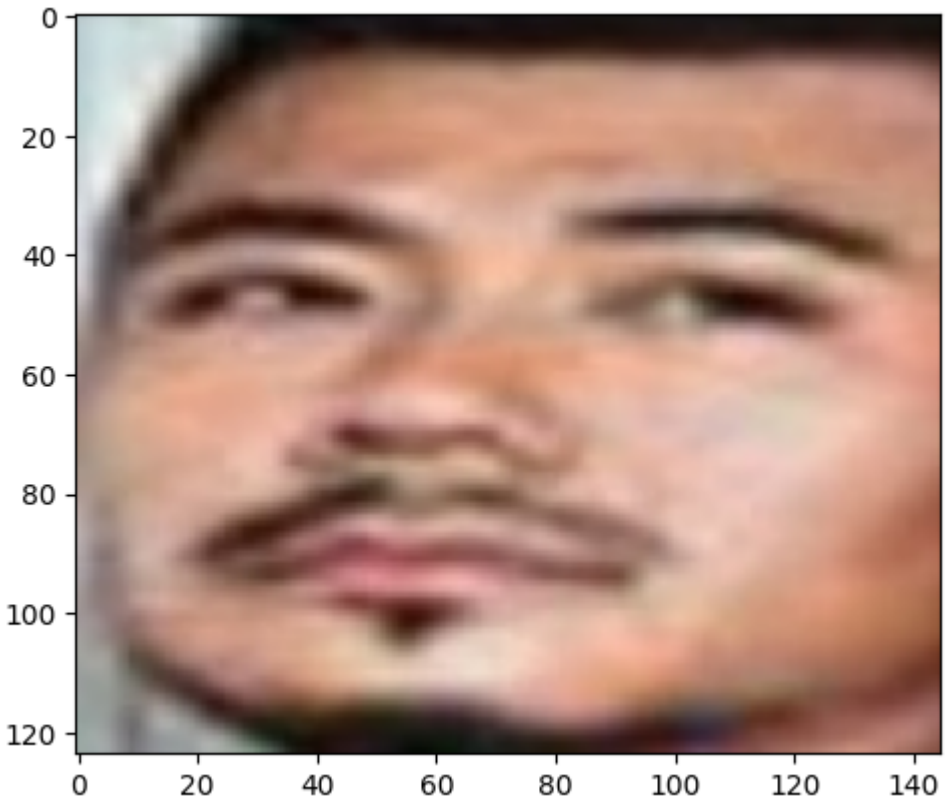
Out[ ]:
```
<matplotlib.image.AxesImage at 0x2b11eaa2c10>
```



B- Displayinh and images without mask:

In [ ]:
```
img = mpimg.imread('data\without_mask\without_mask_11.jpg')
plt.imshow(img)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x2b120cc4550>
```



C- Converting & resising image to numpy array:

In [ ]:
```
fileMask = 'G:/Mon Drive/Deep_learning Python/TestMask/data/with_mask/'
fileWithoutMask = 'G:/Mon Drive/Deep_learning Python/TestMask/data/without_mask/
data = []
for img in data_withMask:
    image = Image.open(fileMask + img)
    image = image.resize( (128 ,128) )
    image = image.convert('RGB')
    image = np.array(image)
    data.append(image)

for img in data_withoutMask:
    image = Image.open(fileWithoutMask + img)
    image = image.resize( (128 ,128) )
    image = image.convert('RGB')
    image = np.array(image)
    data.append(image)
```

```
c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\PIL\Image.p
y:981: UserWarning: Palette images with Transparency expressed in bytes should be
converted to RGBA images
  warnings.warn(
```

D- Type of the data:

In [ ]:
```
type(data)
```

Out[ ]:
```
list
```

E- Number of images in data:

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

Out[ ]: 7553

F- View some images:

In [ ]:
```
data[0:3]
```

Out[ ]:
```
[array([[[50, 58, 31],
         [46, 55, 33],
         [56, 58, 42],
         ...,
         [27, 32, 25],
         [29, 33, 19],
         [33, 40, 17]],

        [[47, 54, 30],
         [49, 58, 36],
         [59, 63, 40],
         ...,
         [18, 21, 11],
         [37, 42, 31],
         [31, 41, 23]],

        [[51, 58, 37],
         [50, 59, 37],
         [59, 64, 37],
         ...,
         [45, 44, 30],
         [29, 34, 27],
         [20, 30, 18]],

        ...,

        [[63, 73, 37],
         [66, 75, 41],
         [74, 84, 50],
         ...,
         [ 5,  3,  0],
         [20, 19, 14],
         [17, 16, 12]],

        [[58, 72, 34],
         [60, 70, 39],
         [68, 75, 49],
         ...,
         [16, 15, 12],
         [15, 13, 10],
         [13,  9,  7]],

        [[53, 69, 30],
         [53, 64, 37],
         [57, 64, 42],
         ...,
         [18, 18, 16],
         [16, 13, 12],
         [19, 12, 13]]], dtype=uint8),
 array([[[117,  61,  15],
         [121,  65,  17],
         [129,  73,  23],
         ...,
         [ 87,  53,  26],
         [ 87,  53,  27],
         [ 88,  53,  31]],

        [[134,  78,  31],
         [136,  80,  33],
         [132,  76,  28],
```

```
         ...,
         [ 78,  44,  17],
         [ 74,  40,  15],
         [ 78,  43,  21]],

        [[124,  68,  21],
         [124,  69,  22],
         [114,  59,  13],
         ...,
         [ 77,  43,  15],
         [ 79,  45,  20],
         [ 78,  43,  23]],

        ...,

        [[175, 135,  70],
         [181, 141,  76],
         [180, 140,  75],
         ...,
         [202, 164,  99],
         [197, 159,  94],
         [186, 147,  82]],

        [[186, 146,  79],
         [185, 144,  77],
         [183, 142,  75],
         ...,
         [208, 170, 104],
         [204, 166, 101],
         [188, 149,  83]],

        [[192, 152,  83],
         [189, 149,  80],
         [183, 143,  74],
         ...,
         [213, 176, 108],
         [214, 177, 109],
         [203, 165,  97]]], dtype=uint8),
 array([[[ 18,  16,  19],
         [ 23,  21,  24],
         [ 28,  26,  29],
         ...,
         [ 32,  27,  34],
         [ 97,  92,  99],
         [161, 156, 163]],

        [[ 24,  22,  25],
         [ 27,  25,  28],
         [ 31,  29,  32],
         ...,
         [ 36,  30,  37],
         [ 80,  74,  81],
         [125, 120, 127]],

        [[ 29,  27,  30],
         [ 31,  29,  32],
         [ 32,  31,  33],
         ...,
         [ 45,  41,  48],
         [ 64,  59,  66],
```

```
         [ 86,  82,  89]],

        ...,

       [[ 32,  12,   5],
        [ 28,   9,   2],
        [ 23,   4,   0],
        ...,
        [ 55,  23,   8],
        [ 64,  29,  13],
        [ 70,  34,  18]],

       [[ 31,  11,   4],
        [ 27,   8,   2],
        [ 22,   3,   0],
        ...,
        [ 55,  23,   8],
        [ 64,  29,  13],
        [ 70,  34,  18]],

       [[ 30,  11,   5],
        [ 27,   8,   3],
        [ 23,   4,   0],
        ...,
        [ 57,  24,   8],
        [ 65,  31,  13],
        [ 71,  36,  16]]], dtype=uint8)]
```

G- Type & dimmenssion of the images:

In [ ]:
```
print(data[0].shape)
print(type(data[0]))
```

```
(128, 128, 3)
<class 'numpy.ndarray'>
```

5. To numpy array:

A- Separating the variables:

In [ ]:
```
X = np.array(data)
Y = np.array(labels)
```

B- Type of X & Y:

In [ ]:
```
print(type(X))
print(type(Y))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

C- Shape of X & Y:

In [ ]:
```
print(X.shape)
print(Y.shape)
```

```
(7553, 128, 128, 3)
(7553,)
```

6. Train test split:

A- Creating the variables of testing & training:

In [ ]:
```
X_train , X_test , Y_train , Y_test = train_test_split(X,Y , test_size=0.2, rand
```

B- Shape of X & X_train & X_test:

In [ ]:
```
print(X.shape , X_train.shape , X_test.shape)
```

```
(7553, 128, 128, 3) (6042, 128, 128, 3) (1511, 128, 128, 3)
```

C- Scaling the data:

In [ ]:
```
X_trainStd = X_train / 255
X_testStd = X_test / 255
```

In [ ]:
```
X_trainStd[0][0][0]
```

Out[ ]:
```
array([0.88627451, 0.91764706, 0.78823529])
```

7. Building our neural network (CNN) :

A- Number of classes:

In [ ]:
```
numClasses = 2 # With mask & without mask
```

B- Creating the model:

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

model.add(keras.layers.Conv2D(32, kernel_size=(3,3), activation='relu', input_sh
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))


model.add(keras.layers.Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))

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

model.add(keras.layers.Dense(128, activation='relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dropout(0.5))


model.add(keras.layers.Dense(numClasses, activation='sigmoid'))
```

```
WARNING:tensorflow:From c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\backend.py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

C- Compiling the model:

In [ ]:
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['acc'])
```

```
WARNING:tensorflow:From c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\optimizers\__init__.py:309: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.
```

D- Training the model:

In [ ]:
```
history = model.fit(X_trainStd, Y_train, validation_split=0.1, epochs=5)
```

```
Epoch 1/5
WARNING:tensorflow:From c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

170/170 [==============================] - 27s 150ms/step - loss: 0.4992 - acc: 0.7830 - val_loss: 0.3132 - val_acc: 0.8843
Epoch 2/5
170/170 [==============================] - 26s 153ms/step - loss: 0.3225 - acc: 0.8766 - val_loss: 0.2710 - val_acc: 0.9091
Epoch 3/5
170/170 [==============================] - 26s 156ms/step - loss: 0.2612 - acc: 0.8981 - val_loss: 0.3203 - val_acc: 0.8793
Epoch 4/5
170/170 [==============================] - 27s 158ms/step - loss: 0.2395 - acc: 0.9161 - val_loss: 0.2379 - val_acc: 0.9207
Epoch 5/5
170/170 [==============================] - 26s 154ms/step - loss: 0.2042 - acc: 0.9257 - val_loss: 0.2391 - val_acc: 0.9240
```

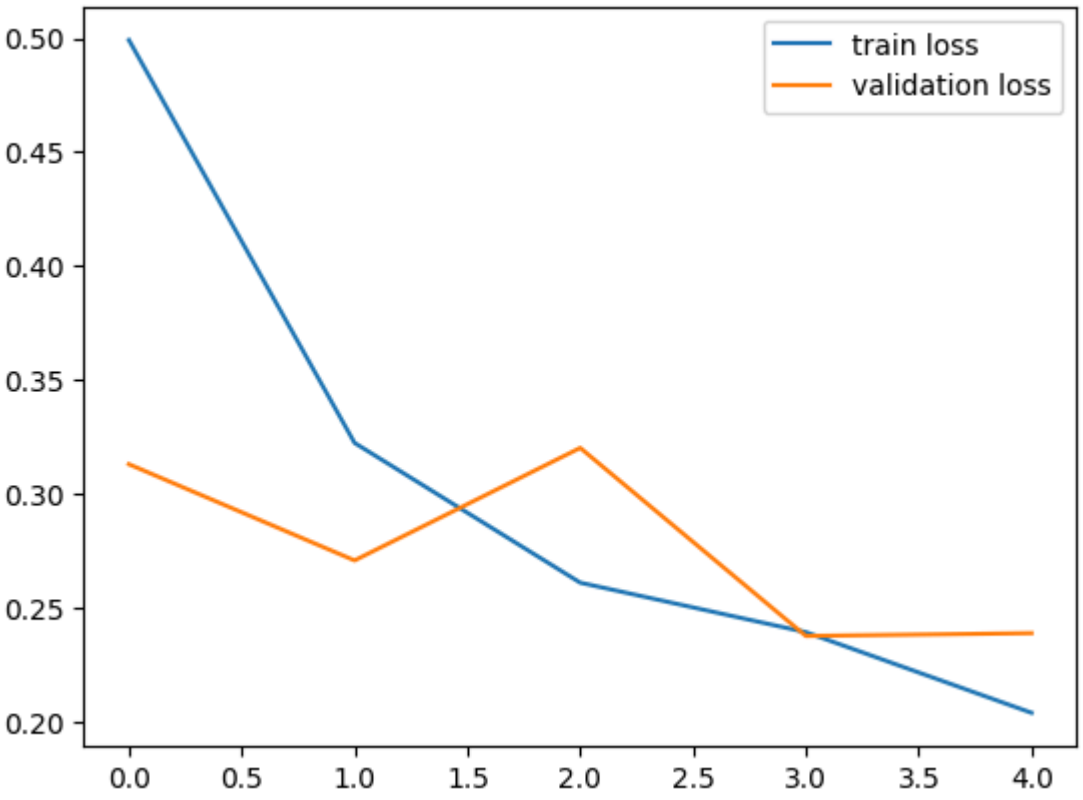
E- Accuracy & score:

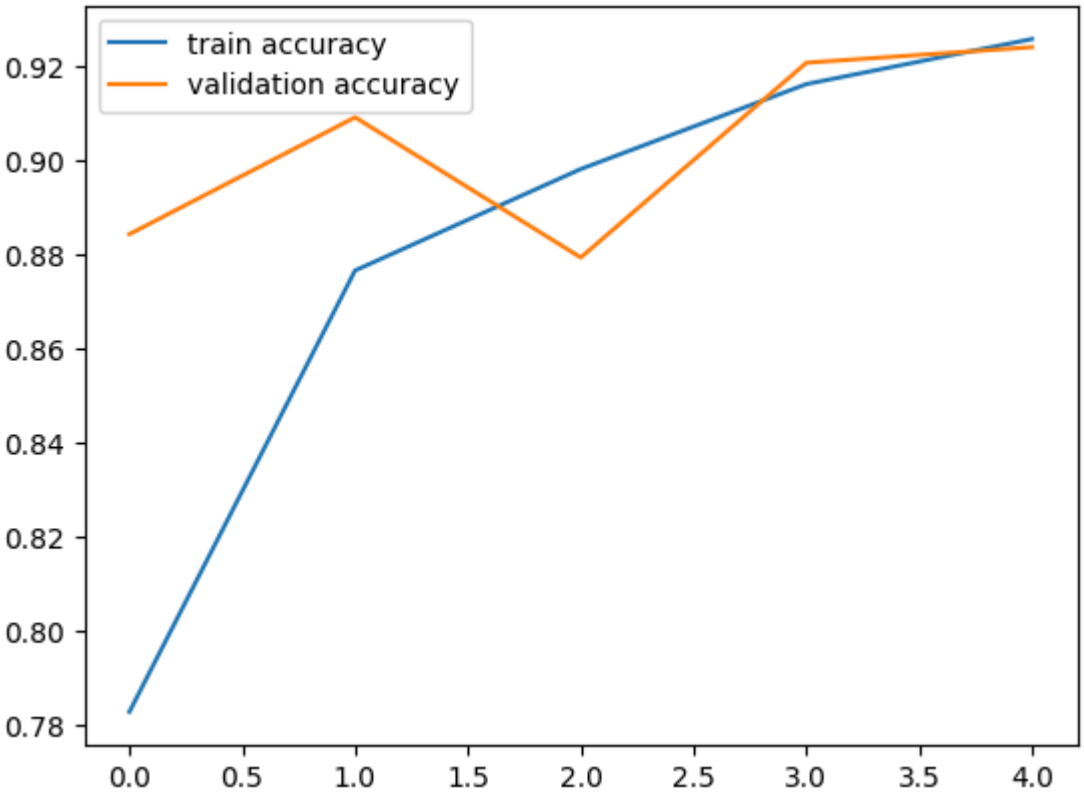
In [ ]:
```
loss , accuracy = model.evaluate(X_testStd , Y_test)
print("Accuracy score:", accuracy)
print("Loss score: ", loss)
```

```
48/48 [==============================] - 1s 30ms/step - loss: 0.2449 - acc: 0.9054
Accuracy score: 0.9053606986999512
Loss score:  0.24487851560115814
```

F- Visulasiation of score & accuracy:

In [ ]:
```
h = history

# plot the loss value
plt.plot(h.history['loss'], label='train loss')
plt.plot(h.history['val_loss'], label='validation loss')
plt.legend()
plt.show()

# plot the accuracy value
plt.plot(h.history['acc'], label='train accuracy')
plt.plot(h.history['val_acc'], label='validation accuracy')
plt.legend()
plt.show()
```





G- Save the model:

In [ ]:
```
# After training the model, save it to a file:
model.save('path_to_my_model.h5')
```

```
c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

8. Example:

A- Loading the model:

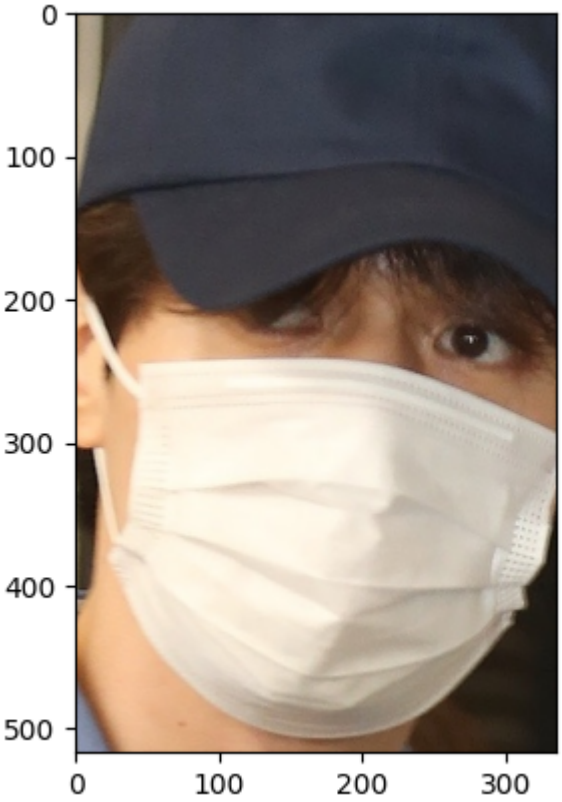
In [ ]:
```
model = keras.models.load_model('path_to_my_model.h5')
```

```
WARNING:tensorflow:From c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\backend.py:1398: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

In [ ]:
```
def maskPrediction(path):
    input_image = Image.open(path)
    # Redimensionner l'image à 128x128 pixels
    image_resized = input_image.resize((128, 128))
    # Convertir l'image PIL en un array numpy
    image_np = np.array(image_resized)
```

```
    # Normaliser les pixels
    image_np_normalized = image_np / 255.0
    image_reshaped = np.reshape(image_np_normalized, [1, 128, 128, 3])
    # Prédire la classe
    predict = model.predict(image_reshaped)
    predict_mask = np.argmax(predict)
    # Afficher l'image originale
    plt.imshow(input_image)
    plt.show()
    print('The predicted number is:', predict_mask)
    if (predict_mask == 0):
        print("The person is wearing a mask")
    else:
        print("The person isn't wearing a mask")

# Exemple d'utilisation
path = "data\with_mask\with_mask_39.jpg"
maskPrediction(path)
```

```
1/1 [==============================] - 0s 31ms/step
```




```
The predicted number is: 0
The person is wearing a mask
```