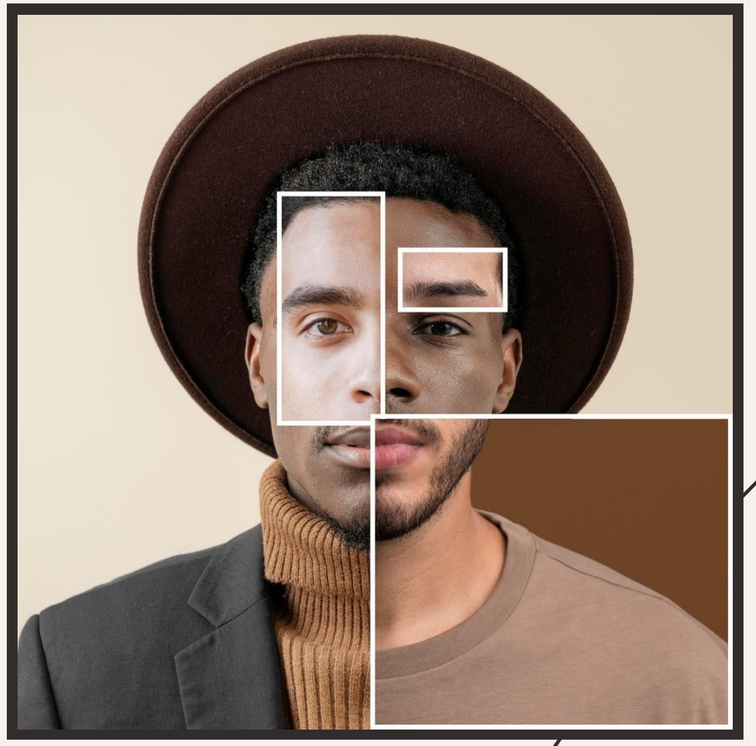
Enhancing Security with MaskSense AI Technology

Introduction

The use of MaskSense AI technology is becoming crucial for enhancing security in public spaces. This technology can identify individuals wearing masks and provide an additional layer of protection. With the increasing need for safety measures, MaskSense AI is a valuable tool for security.

Understanding MaskSense AI

MaskSense AI technology utilizes advanced algorithms to detect and identify individuals wearing masks. By analyzing facial features such as the eyes, eyebrows, and forehead, the system can accurately recognize individuals even when their faces are partially covered. This technology is essential for security applications in various environments.



Key Features of MaskSense AI

The key features of MaskSense AI technology include real-time detection of masked individuals, seamless integration with existing security systems, and the ability to generate alerts when unauthorized individuals are detected. Additionally, the technology offers a high level of accuracy and can adapt to different lighting conditions.

Applications in Public Safety

MaskSense AI technology has diverse applications in public safety, including airport security, public transportation, and access control in public buildings. By implementing this technology, organizations can enhance security protocols and mitigate potential risks associated with masked individuals in public spaces.

Challenges and Considerations

While MaskSense AI technology offers significant benefits, there are challenges to consider, such as privacy concerns, potential biases in the recognition process, and the need for ongoing updates to adapt to new mask designs. Addressing these challenges is essential for the responsible implementation of this technology.

Enhancing Security Protocols

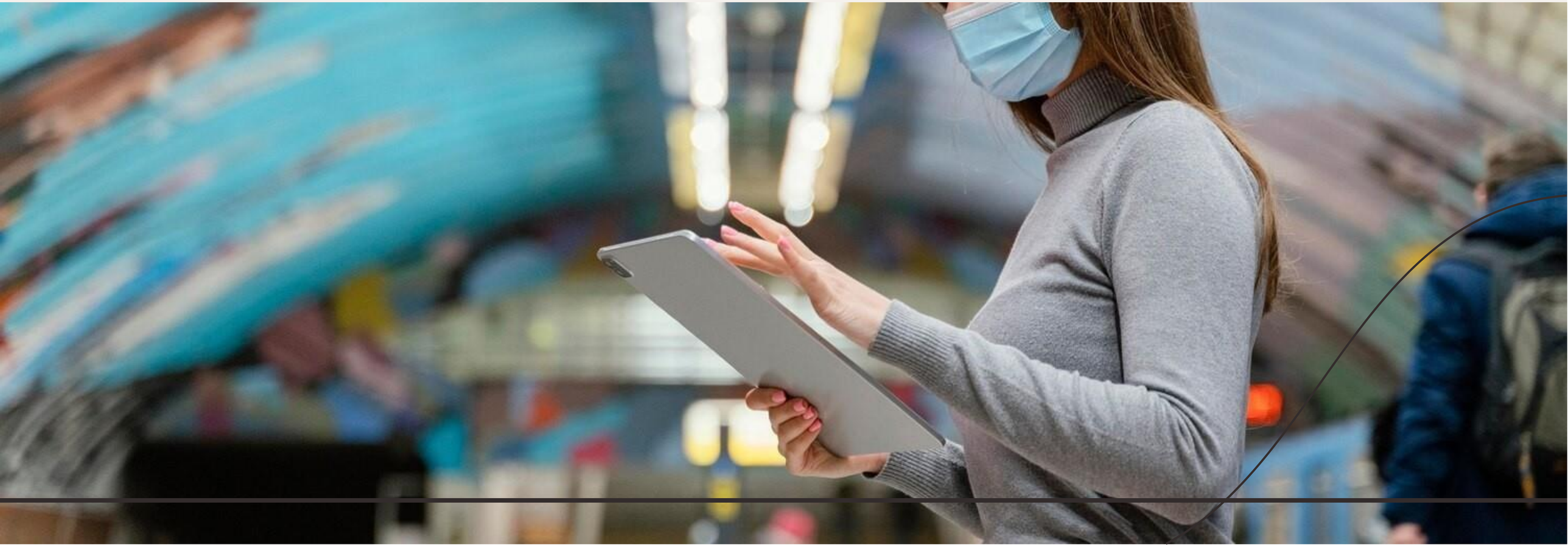
Integrating MaskSense AI technology into existing security protocols can significantly enhance the overall
and security of public spaces. By leveraging this technology, organizations can strengthen their defense against security threats and ensure a proactive approach to identifying individuals in various settings.

The future of MaskSense AI technology is poised for continuous advancements, including improved accuracy, enhanced AI algorithms, and the integration of additional biometric identifiers. These developments will further strengthen the capabilities of the technology and expand its applications across diverse industries.

Adoption and Implementation

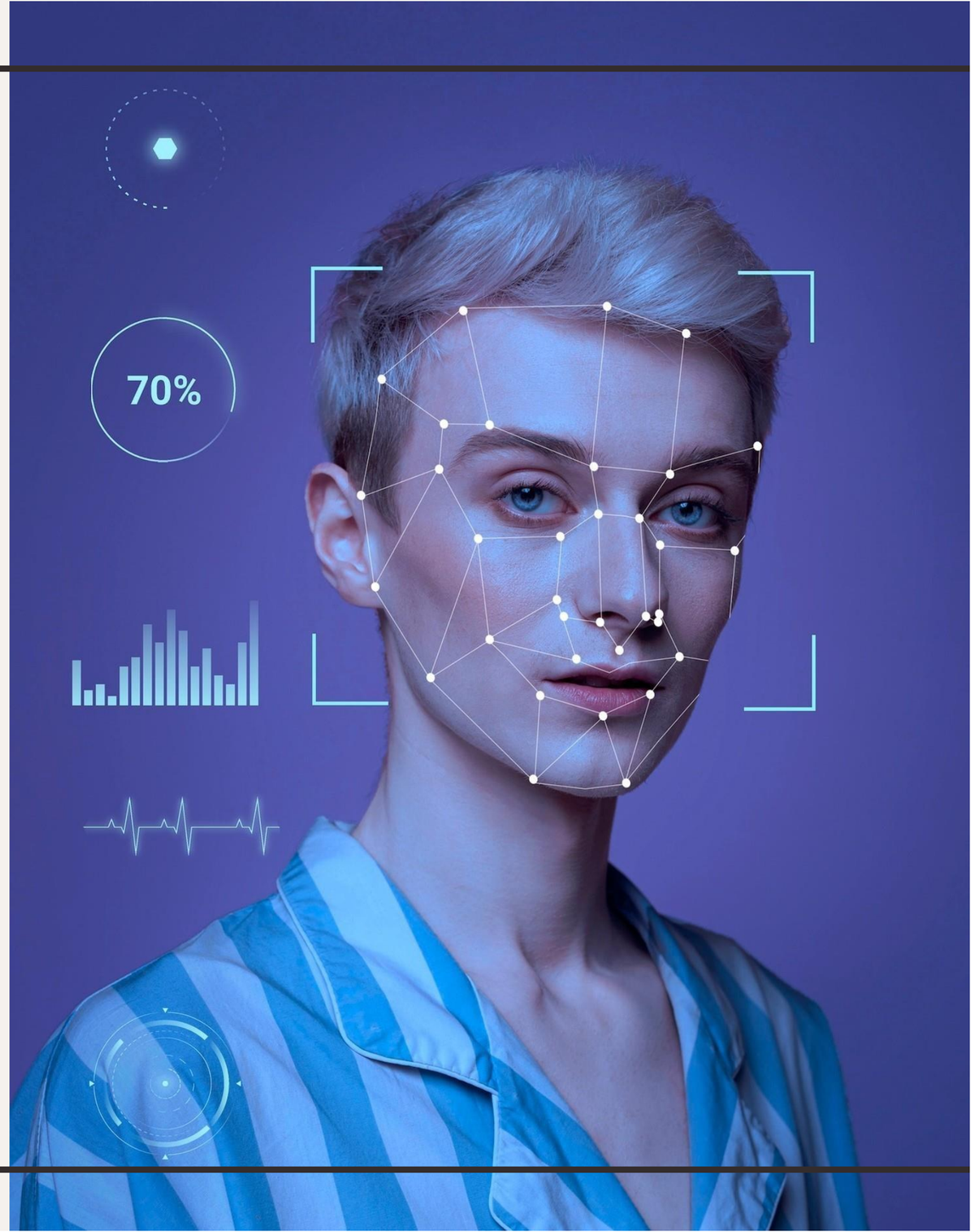
The adoption and implementation of MaskSense AI technology require careful planning, stakeholder collaboration, and adherence to regulatory standards. Organizations must prioritize ethical considerations, data protection, and user consent to ensure responsible deployment and effective utilization of this technology.

The widespread adoption of MaskSense AI technology offers substantial benefits, including enhanced security, improved public safety, and the ability to mitigate security risks associated with masked individuals. This technology has a significant impact on safeguarding public spaces and ensuring the well-being of individuals.



Conclusion

In conclusion, MaskSense AI technology plays a vital role in enhancing security measures and ensuring public safety. With its ability to identify individuals wearing masks, this technology offers a valuable solution for security challenges in various environments. As advancements continue, MaskSense AI will continue to be a cornerstone of security protocols.



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import os
```

In [2]:

```
# Import Tenosrflow
import tensorflow as tf
from tensorflow import keras
import zipfile
```

```
2024-04-22 18:44:48.272272: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
2024-04-22 18:44:48.272392: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered
2024-04-22 18:44:48.446023: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
```

In [3]:

```
train_mask_dir = os.path.join('/kaggle/input/face-mask-dataset/data/with_mask')
train_without_mask_dir = os.path.join('/kaggle/input/face-mask-dataset/data/without_mask')
```

In [4]:

```
train_mask = os.listdir(train_mask_dir)
print(f'  Mask Data \n\n{train_mask[:10]}')
train_without = os.listdir(train_without_mask_dir)
print(f'\n\nWithout MaskData\n{train_without[:10]}')
```

```
  Mask Data

['with_mask_3326.jpg', 'with_mask_3139.jpg', 'with_mask_696.jpg', 'with_mask_2867.jpg', 'with_mask_39.jpg', 'with_mask_1811.jpg', 'with_mask_532.jpg', 'with_mask_1619.jpg', 'with_mask_3471.jpg', 'with_mask_3501.jpg']


Without MaskData
['without_mask_3248.jpg', 'without_mask_2803.jpg', 'without_mask_650.jpg', 'without_mask_2060.jpg', 'without_mask_559.jpg', 'without_mask_3273.jpg', 'without_mask_124.jpg', 'without_mask_1849.jpg', 'without_mask_139.jpg', '
without_mask_2137.jpg']
```

In [5]:

```
print('Mask Length: ',len(os.listdir(train_mask_dir)))
print('With Out Mask Length: ',len(os.listdir(train_without_mask_dir)))
```

```
Mask Length:  3725
With Out Mask Length:  3828
```

In [6]:

```
import matplotlib.image as mping
%matplotlib inline
nrows = 4
ncols = 4
pic_index = 0
```

In [7]:

```
import os
import matplotlib.pyplot as plt

folder_path = "/kaggle/input/face-mask-dataset/data/with_mask"
files = os.listdir(folder_path)

num_printed = 0

fig, axs = plt.subplots(2, 4, figsize=(12, 6))

print('Mask Image\n\n')

for file in files:

    if file.endswith(".jpg") or file.endswith(".png") or file.endswith(".jpeg"):
```

```
        image_path = os.path.join(folder_path, file)

        img = plt.imread(image_path)
        row_index = num_printed // 4
        col_index = num_printed % 4
        axs[row_index, col_index].imshow(img)
        axs[row_index, col_index].axis('off')

        num_printed += 1
        if num_printed == 8:
            break

# Hide any empty subplots
for i in range(num_printed, 8):
    row_index = i // 4
    col_index = i % 4
    axs[row_index, col_index].axis('off')

# Adjust layout
plt.tight_layout()
plt.show()
```

Mask Image

In [8]:

```
import os
import matplotlib.pyplot as plt

folder_path = "/kaggle/input/face-mask-dataset/data/without_mask"
files = os.listdir(folder_path)

num_printed = 0

fig, axs = plt.subplots(2, 4, figsize=(12, 6))

print('With Out Mask Image\n\n')

for file in files:

    if file.endswith(".jpg") or file.endswith(".png") or file.endswith(".jpeg"):

        image_path = os.path.join(folder_path, file)

        img = plt.imread(image_path)
        row_index = num_printed // 4
        col_index = num_printed % 4
        axs[row_index, col_index].imshow(img)
```

```
        axs[row_index, col_index].axis('off')

        num_printed += 1
        if num_printed == 8:
            break

# Hide any empty subplots
for i in range(num_printed, 8):
    row_index = i // 4
    col_index = i % 4
    axs[row_index, col_index].axis('off')

# Adjust layout
plt.tight_layout()
plt.show()
```

With Out Mask Image

In [9]:

```
# //

model = tf.keras.Sequential([
```

In [10]:

```
    tf.keras.layers.Conv2D(16,(3,3),activation='relu',input_shape= (300,300,3)),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(32,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512,activation='relu'),
    tf.keras.layers.Dense(1,activation='sigmoid')
])
```

In [11]:

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 298, 298, 16)	448
max_pooling2d (MaxPooling2D)	(None, 149, 149, 16)	0
conv2d_1 (Conv2D)	(None, 147, 147, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 73, 73, 32)	0
conv2d_2 (Conv2D)	(None, 71, 71, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 35, 35, 64)	0
conv2d_3 (Conv2D)	(None, 33, 33, 64)	36,928
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 64)	0

```
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_4 (Conv2D)               │ (None, 14, 14, 64)     │        36,928 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_4 (MaxPooling2D)  │ (None, 7, 7, 64)       │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 3136)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 512)            │     1,606,144 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 1)              │           513 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 1,704,097 (6.50 MB)

 Trainable params: 1,704,097 (6.50 MB)

 Non-trainable params: 0 (0.00 B)
```

In [12]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_data = ImageDataGenerator(rescale = 1 /255)
train_gernater = train_data.flow_from_directory(
    '/kaggle/input/face-mask-dataset/data',
    target_size=(300,300),
    batch_size=128,
    class_mode = 'binary'
)
```

```
Found 7553 images belonging to 2 classes.
```

In [13]:

```
class myCallBack(tf.keras.callbacks.Callback):
    def on_epoch(self,epoch,logs={}):
        if (logs.get('accuracy' >= 0.98)):
            print('Reach ((98% Accuracy so Canclle traning')
            self.model.stop_traning = True
callbacks  = myCallBack()
```

In [14]:

```
from tensorflow.keras.optimizers import RMSprop

model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(learning_rate=0.001),
              metrics=['accuracy'])
```

In [15]:

```
history = model.fit(
    train_gernater,
    steps_per_epoch=8,
    epochs=15,
    verbose=1,
    callbacks=[callbacks]
)
```

```
Epoch 1/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 66s 6s/step - accuracy: 0.4858 - loss: 0.7044
Epoch 2/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 50s 6s/step - accuracy: 0.5727 - loss: 0.6872
Epoch 3/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 45s 6s/step - accuracy: 0.5719 - loss: 0.6583
Epoch 4/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 50s 6s/step - accuracy: 0.7311 - loss: 0.8187
Epoch 5/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 50s 6s/step - accuracy: 0.8458 - loss: 0.3887
Epoch 6/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 50s 6s/step - accuracy: 0.8566 - loss: 0.3428
Epoch 7/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 50s 6s/step - accuracy: 0.8397 - loss: 0.3663
Epoch 8/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 25s 3s/step - accuracy: 0.8302 - loss: 0.3973
Epoch 9/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 56s 6s/step - accuracy: 0.8959 - loss: 0.3024
Epoch 10/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 50s 6s/step - accuracy: 0.8646 - loss: 0.3291
Epoch 11/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 50s 6s/step - accuracy: 0.8417 - loss: 0.3595
Epoch 12/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 50s 6s/step - accuracy: 0.8801 - loss: 0.2776
Epoch 13/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 50s 6s/step - accuracy: 0.8348 - loss: 0.4358
Epoch 14/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 46s 6s/step - accuracy: 0.8508 - loss: 0.3871
Epoch 15/15
8/8 ━━━━━━━━━━━━━━━━━━━━ 50s 6s/step - accuracy: 0.8983 - loss: 0.3083
```

In [16]:

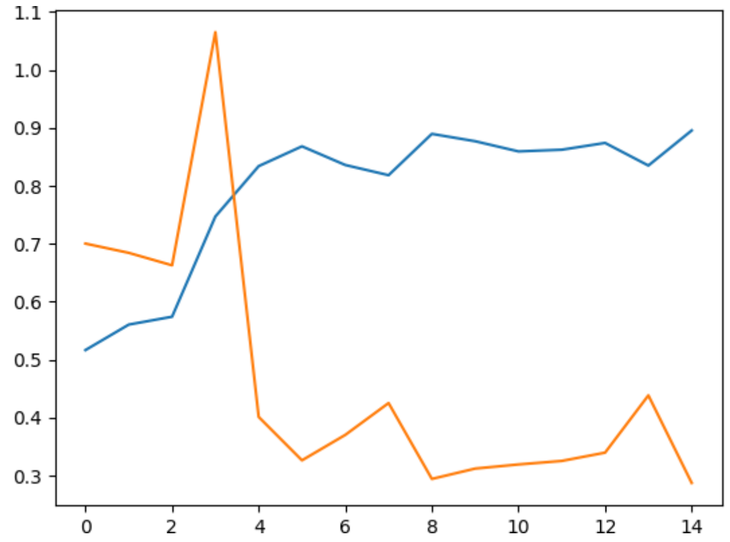
```
acc = history.history['accuracy']
loss = history.history['loss']
epochs = range(len(acc))
```

In [17]:

```
plt.plot(epochs,acc,label='Accuracy')
plt.plot(epochs,loss)
```

Out[17]:

```
[<matplotlib.lines.Line2D at 0x7b968469cb80>]
```



In [ ]:

In [18]:

```
import numpy as np
import os
from keras.preprocessing import image

uploaded_images_folder = "/kaggle/input/face-mask-dataset/data/with_mask"
n=0
for fn in os.listdir(uploaded_images_folder):
    path = os.path.join(uploaded_images_folder, fn)
    read_image = plt.imread(path)
    plt.imshow(read_image)
    img = image.load_img(path, target_size=(300,300))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    images = np.vstack([x])

    classes = model.predict(images, batch_size=10)
    print(classes[0])
    if classes[0] > 0.5:
        print(fn + 'Not mask')
    else:
        print(fn + '\n\nMask Image' )
    n= n+1
    if n==1:
        break
```

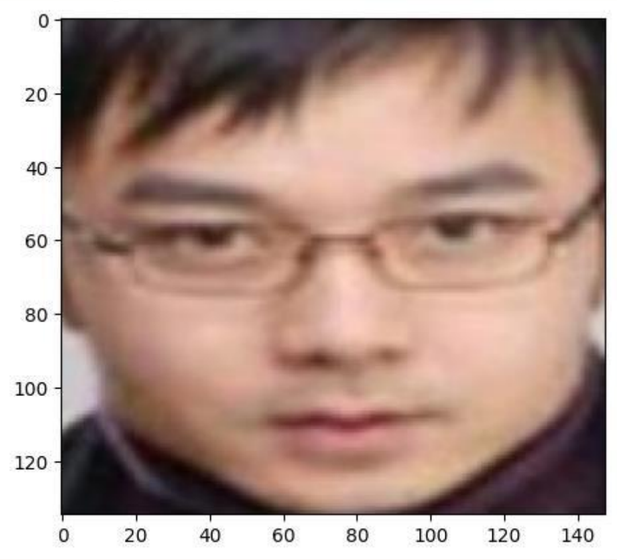
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step
[0.]
with_mask_3326.jpg

Mask Image
```

In [19]:

```
import numpy as np
import os
from keras.preprocessing import image

uploaded_images_folder = "/kaggle/input/face-mask-dataset/data/without_mask"
n=0
for fn in os.listdir(uploaded_images_folder):
    path = os.path.join(uploaded_images_folder, fn)

    read_image = plt.imread(path)
    plt.imshow(read_image)


    img = image.load_img(path, target_size=(300,300))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    images = np.vstack([x])

    classes = model.predict(images, batch_size=10)
    print(classes[0])
    if classes[0] > 0.5:
        print(fn + '\n\nNot maskImage')
    else:
        print(fn + 'Mask Image' )
    n= n+1
    if n==1:
        break
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
[1.]
without_mask_3248.jpg
```

Not maskImage



In [ ]:

linkcode