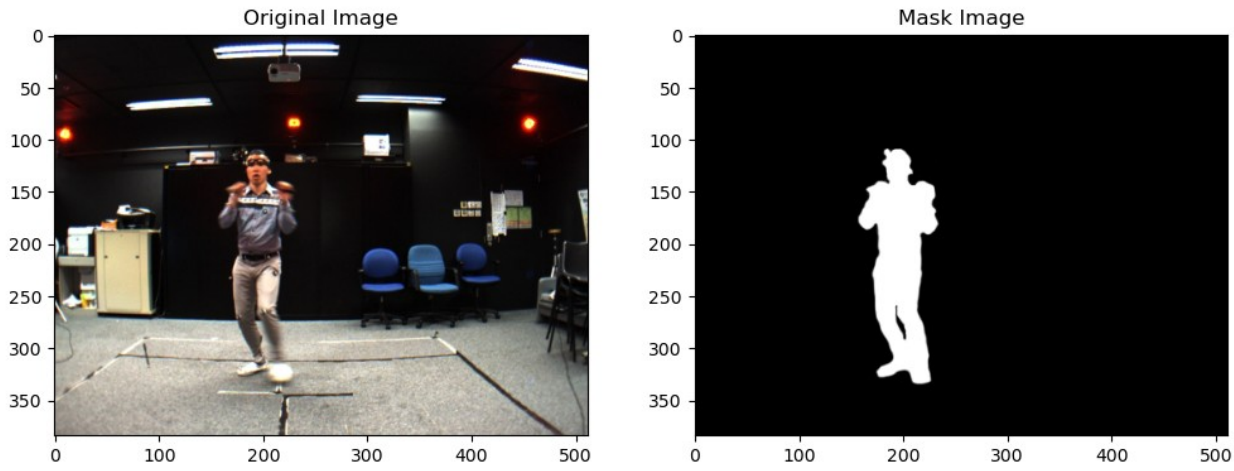
```
import pandas as pd
import numpy as np
import cv2
import os
from matplotlib import pyplot as plt

df = pd.read_csv('data/df.csv')
df.head()
```

```
   Unnamed: 0                                images  \
0           0  images/HipHop_HipHop1_C0_00180.png
1           1  images/HipHop_HipHop1_C0_00225.png
2           2  images/HipHop_HipHop1_C0_00360.png
3           3  images/HipHop_HipHop1_C0_00405.png
4           4  images/HipHop_HipHop1_C0_00450.png

                                masks
collages
0  masks/HipHop_HipHop1_C0_00180.png
collages/HipHop_HipHop1_C0_00180.jpg
1  masks/HipHop_HipHop1_C0_00225.png
collages/HipHop_HipHop1_C0_00225.jpg
2  masks/HipHop_HipHop1_C0_00360.png
collages/HipHop_HipHop1_C0_00360.jpg
3  masks/HipHop_HipHop1_C0_00405.png
collages/HipHop_HipHop1_C0_00405.jpg
4  masks/HipHop_HipHop1_C0_00450.png
collages/HipHop_HipHop1_C0_00450.jpg
```

```
def load_image_and_mask(row):
    image_path = os.path.join('data', row['images'])
    mask_path = os.path.join('data', row['masks'])

    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Converting to
RGB

    mask = cv2.imread(mask_path, 0)  # 0 used for grayscale

    return image, mask

row = df.iloc[0]

# Load image and mask
image, mask = load_image_and_mask(row)

# Display the image and mask
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(image)
plt.title('Original Image')
```

```
plt.subplot(1, 2, 2)
plt.imshow(mask, cmap='gray')
plt.title('Mask Image')
plt.show()
```




```
import pandas as pd
import cv2
import numpy as np
from sklearn.model_selection import train_test_split

# Load the dataset
df = pd.read_csv('data/df.csv')

def load_images(file_paths, resize_dim=(128, 128),
convert_to_gray=False):
    images = []
    for fp in file_paths:
        img = cv2.imread(fp)
        if img is not None:
            if convert_to_gray:
                img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            img = cv2.resize(img, resize_dim,
interpolation=cv2.INTER_AREA)
            images.append(img)
    images = np.array(images, dtype='float32') / 255.0
    if convert_to_gray:
        images = images.reshape(images.shape[0], resize_dim[0],
resize_dim[1], 1)  # Add channel dimension for grayscale
    return images

# Correctly construct file paths
image_paths = df['images'].apply(lambda x: 'data/' + x).tolist()
mask_paths = df['masks'].apply(lambda x: 'data/' + x).tolist()
```

```python
# Load and preprocess images and masks
images = load_images(image_paths)
masks = load_images(mask_paths, convert_to_gray=True)
if images is None or masks is None:
    print("Failed to load images or masks.")
else:
    # Split data into training and testing
    X_train, X_test, y_train, y_test = train_test_split(images, masks,
test_size=0.2, random_state=42)
```

```
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
```

```
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
```

```
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
```

```python
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D,
UpSampling2D, concatenate

def unet_model(input_size=(128, 128, 3)):
    inputs = Input(input_size)

    # Encoder
    conv1 = Conv2D(64, 3, activation='relu', padding='same')(inputs)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = Conv2D(128, 3, activation='relu', padding='same')(pool1)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    # Middle
    conv3 = Conv2D(256, 3, activation='relu', padding='same')(pool2)

    # Decoder
    up1 = UpSampling2D(size=(2, 2))(conv3)
    conv4 = Conv2D(128, 2, activation='relu', padding='same')(up1)
    merged1 = concatenate([conv2, conv4], axis=3)
    conv5 = Conv2D(128, 3, activation='relu', padding='same')(merged1)

    up2 = UpSampling2D(size=(2, 2))(conv5)
    conv6 = Conv2D(64, 2, activation='relu', padding='same')(up2)
    merged2 = concatenate([conv1, conv6], axis=3)
    conv7 = Conv2D(64, 3, activation='relu', padding='same')(merged2)

    # Output layer
    output = Conv2D(1, 1, activation='sigmoid')(conv7)

    model = Model(inputs=inputs, outputs=output)

    return model
```

```
model = unet_model()
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# #Save the model
# model.save('unet_image_segmentation_model.h5')

model.summary()
```

```
Model: "model_3"
__________________________________________________________________________
_____________________________
 Layer (type)                   Output Shape         Param #
Connected to
==========================================================================
============================
 input_6 (InputLayer)           [(None, 128, 128, 3  0           []

                                )]



 conv2d_35 (Conv2D)             (None, 128, 128, 64  1792
['input_6[0][0]']
                                )



 max_pooling2d_10 (MaxPooling2D  (None, 64, 64, 64)  0
['conv2d_35[0][0]']
 )



 conv2d_36 (Conv2D)             (None, 64, 64, 128)  73856
['max_pooling2d_10[0][0]']


 max_pooling2d_11 (MaxPooling2D  (None, 32, 32, 128)  0
['conv2d_36[0][0]']
 )



 conv2d_37 (Conv2D)             (None, 32, 32, 256)  295168
['max_pooling2d_11[0][0]']


 up_sampling2d_9 (UpSampling2D)  (None, 64, 64, 256)  0
['conv2d_37[0][0]']
```

```
 conv2d_38 (Conv2D)              (None, 64, 64, 128)  131200
['up_sampling2d_9[0][0]']


 concatenate_9 (Concatenate)     (None, 64, 64, 256)  0
['conv2d_36[0][0]',

'conv2d_38[0][0]']


 conv2d_39 (Conv2D)              (None, 64, 64, 128)  295040
['concatenate_9[0][0]']


 up_sampling2d_10 (UpSampling2D  (None, 128, 128, 12  0
['conv2d_39[0][0]']
 )                               8)



 conv2d_40 (Conv2D)              (None, 128, 128, 64  32832
['up_sampling2d_10[0][0]']
                                 )



 concatenate_10 (Concatenate)    (None, 128, 128, 12  0
['conv2d_35[0][0]',
                                 8)
'conv2d_40[0][0]']


 conv2d_41 (Conv2D)              (None, 128, 128, 64  73792
['concatenate_10[0][0]']
                                 )



 conv2d_42 (Conv2D)              (None, 128, 128, 1)  65
['conv2d_41[0][0]']


==================================================================
============================
Total params: 903,745
Trainable params: 903,745
Non-trainable params: 0
__________________________________________________________________
_____________________________
```

```
history = model.fit(X_train, y_train, batch_size=32, epochs=20,
validation_split=0.1)

Epoch 1/20
27/27 [==============================] - 57s 2s/step - loss: 0.2580 -
accuracy: 0.9420 - val_loss: 0.1320 - val_accuracy: 0.9497
Epoch 2/20
27/27 [==============================] - 54s 2s/step - loss: 0.1185 -
accuracy: 0.9453 - val_loss: 0.0868 - val_accuracy: 0.9497
Epoch 3/20
27/27 [==============================] - 52s 2s/step - loss: 0.0776 -
accuracy: 0.9453 - val_loss: 0.0651 - val_accuracy: 0.9497
Epoch 4/20
27/27 [==============================] - 52s 2s/step - loss: 0.0641 -
accuracy: 0.9453 - val_loss: 0.0596 - val_accuracy: 0.9497
Epoch 5/20
27/27 [==============================] - 58s 2s/step - loss: 0.0552 -
accuracy: 0.9453 - val_loss: 0.0511 - val_accuracy: 0.9497
Epoch 6/20
27/27 [==============================] - 52s 2s/step - loss: 0.0489 -
accuracy: 0.9545 - val_loss: 0.0455 - val_accuracy: 0.9697
Epoch 7/20
27/27 [==============================] - 52s 2s/step - loss: 0.0469 -
accuracy: 0.9667 - val_loss: 0.0416 - val_accuracy: 0.9705
Epoch 8/20
27/27 [==============================] - 52s 2s/step - loss: 0.0417 -
accuracy: 0.9681 - val_loss: 0.0350 - val_accuracy: 0.9711
Epoch 9/20
27/27 [==============================] - 52s 2s/step - loss: 0.1213 -
accuracy: 0.9532 - val_loss: 0.1306 - val_accuracy: 0.9497
Epoch 10/20
27/27 [==============================] - 52s 2s/step - loss: 0.0852 -
accuracy: 0.9529 - val_loss: 0.0562 - val_accuracy: 0.9637
Epoch 11/20
27/27 [==============================] - 52s 2s/step - loss: 0.0463 -
accuracy: 0.9640 - val_loss: 0.0347 - val_accuracy: 0.9696
Epoch 12/20
27/27 [==============================] - 54s 2s/step - loss: 0.0342 -
accuracy: 0.9677 - val_loss: 0.0297 - val_accuracy: 0.9712
Epoch 13/20
27/27 [==============================] - 56s 2s/step - loss: 0.0281 -
accuracy: 0.9696 - val_loss: 0.0256 - val_accuracy: 0.9720
Epoch 14/20
27/27 [==============================] - 52s 2s/step - loss: 0.0253 -
accuracy: 0.9704 - val_loss: 0.0231 - val_accuracy: 0.9727
Epoch 15/20
27/27 [==============================] - 50s 2s/step - loss: 0.0233 -
accuracy: 0.9709 - val_loss: 0.0219 - val_accuracy: 0.9731
Epoch 16/20
27/27 [==============================] - 50s 2s/step - loss: 0.0213 -
```

```
accuracy: 0.9714 - val_loss: 0.0207 - val_accuracy: 0.9734
Epoch 17/20
27/27 [==============================] - 52s 2s/step - loss: 0.0212 -
accuracy: 0.9715 - val_loss: 0.0224 - val_accuracy: 0.9728
Epoch 18/20
27/27 [==============================] - 52s 2s/step - loss: 0.0197 -
accuracy: 0.9718 - val_loss: 0.0189 - val_accuracy: 0.9738
Epoch 19/20
27/27 [==============================] - 52s 2s/step - loss: 0.0189 -
accuracy: 0.9720 - val_loss: 0.0185 - val_accuracy: 0.9739
Epoch 20/20
27/27 [==============================] - 52s 2s/step - loss: 0.0183 -
accuracy: 0.9722 - val_loss: 0.0179 - val_accuracy: 0.9740
```

```
model.evaluate(X_test, y_test)
predictions = model.predict(X_test)

predictions = (predictions > 0.5).astype(np.uint8)

# Display the first 5 images and their predicted masks
plt.figure(figsize=(15, 15))

for i in range(5):
    plt.subplot(5, 3, 3 * i + 1)
    plt.imshow(X_test[i])
    plt.title('Original Image')
    plt.subplot(5, 3, 3 * i + 2)
    plt.imshow(y_test[i].reshape(128, 128), cmap='gray')
    plt.title('Original Mask')
    plt.subplot(5, 3, 3 * i + 3)
    plt.imshow(predictions[i].reshape(128, 128), cmap='gray')
    plt.title('Predicted Mask')
```

```
8/8 [==============================] - 4s 533ms/step - loss: 0.0189 -
accuracy: 0.9698
8/8 [==============================] - 4s 529ms/step
```


Original Image
Original Mask
Predicted Mask
Original Image
Original Mask
Predicted Mask
Original Image
Original Mask
Predicted Mask
Original Image
Original Mask
Predicted Mask
Original Image
Original Mask
Predicted Mask