Trần Xuân Triển - B20DCCN691 - 04 - 01

In [4]:

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

# Tạo danh sách để lưu hình ảnh, nhãn và tên file
data = []
labels = []

# Đường dẫn đến thư mục chứa hình ảnh
root_folder = "C:/Users/Admin/Desktop/PTHTTM/Test"

for label in ["Tuong tu","That tinh", "Dang yeu"]:
    label_folder = os.path.join(root_folder, label)

    if os.path.isdir(label_folder):  # Kiểm tra thư mục tồn tại
        for filename in os.listdir(label_folder):
            try:
                if filename.endswith(".jpg") or filename.endswith(".PNG"):
                    image_path = os.path.join(label_folder, filename)  # Đường

                    # Đọc hình ảnh bằng OpenCV
                    img = cv2.imread(image_path, cv2.IMREAD_COLOR)
                    # Đảm bảo rằng hình ảnh có kích thước ít nhất 64x64
                    img = cv2.resize(img, (64, 64))
                    # Đảm bảo rằng hình ảnh có 3 kênh màu (RGB)
                    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                    # Chuyển đổi ảnh thành mảng NumPy
                    img_array = np.array(img)

                    data.append(img_array)
                    labels.append(label)

                    # Hiển thị ma trận NumPy dưới dạng ảnh
                    #plt.imshow(img_array)
                    #plt.show()
            except:
                pass
# Chuyển danh sách thành mảng NumPy
data = np.array(data)
labels = np.array(labels)

# Tạo DataFrame pandas với hình ảnh, nhãn
df = pd.DataFrame({'Image': data.tolist(), 'Label': labels})

# Lưu DataFrame vào tệp CSV
df.to_csv('Test.csv', index=False)
print(df.shape)
```

```
(201, 2)
```

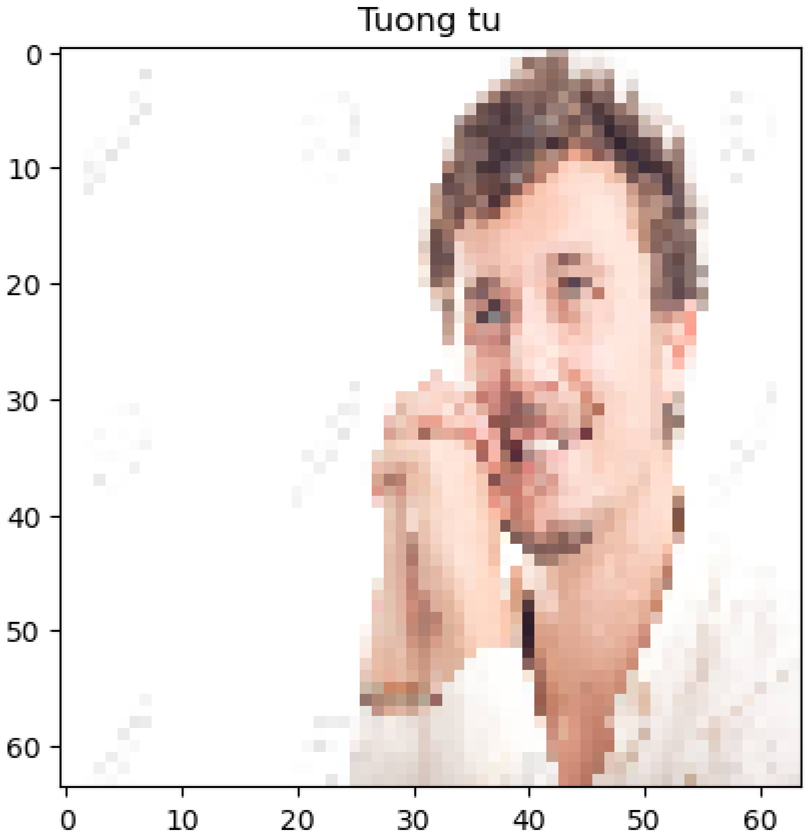
In [5]:

```python
import pandas as pd

df = pd.read_csv('Test.csv')
print(df.head())
```

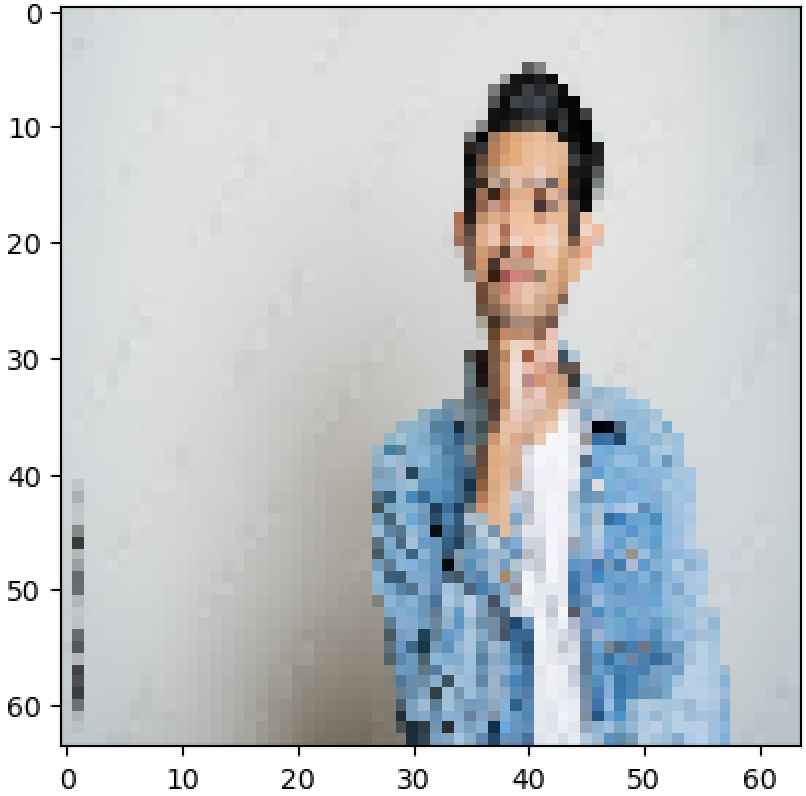
```
                                               Image     Label
0  [[[255, 255, 255], [255, 255, 255], [255, 255,...  Tuong tu
1  [[[206, 214, 216], [209, 214, 217], [210, 214,...  Tuong tu
2  [[[130, 118, 118], [129, 116, 110], [89, 80, 7...  Tuong tu
3  [[[195, 204, 211], [201, 210, 217], [206, 213,...  Tuong tu
4  [[[27, 21, 33], [27, 21, 33], [28, 22, 34], [2...  Tuong tu
```

In [6]:

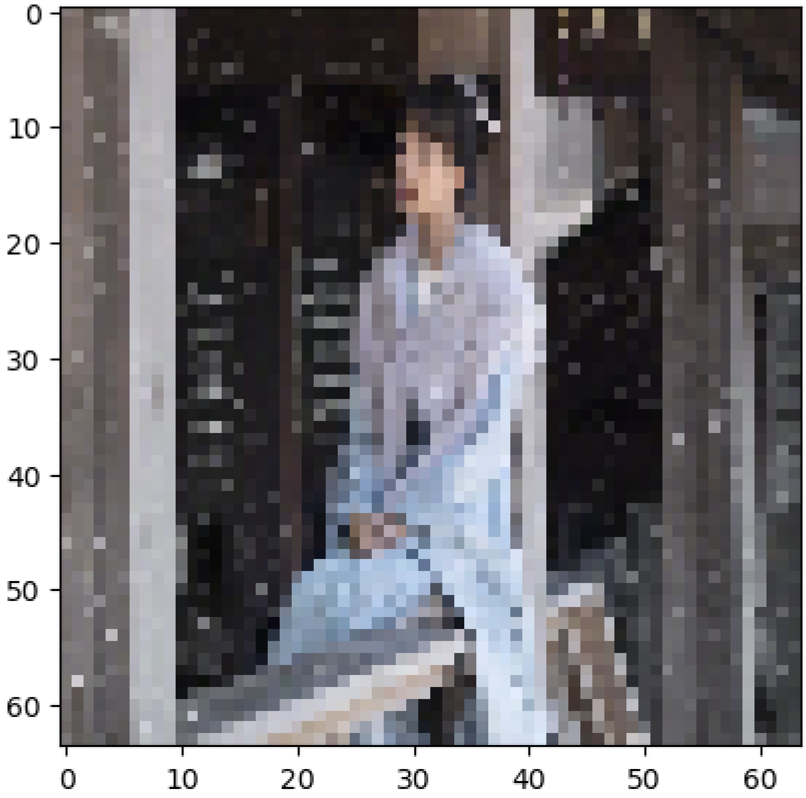
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import ast

# Đọc tệp CSV
df = pd.read_csv('Test.csv')

# Lấy 5 dòng đầu tiên từ DataFrame
first_5_rows = df.head(5)

# Lặp qua từng dòng và hiển thị hình ảnh
for index, row in first_5_rows.iterrows():
    image_data = row['Image']
    label = row['Label']

    # Chuyển chuỗi số thành mảng NumPy
    image_data = np.array(ast.literal_eval(image_data))

    # Hiển thị hình ảnh
    plt.imshow(image_data)
    plt.title(label)
    plt.show()
```

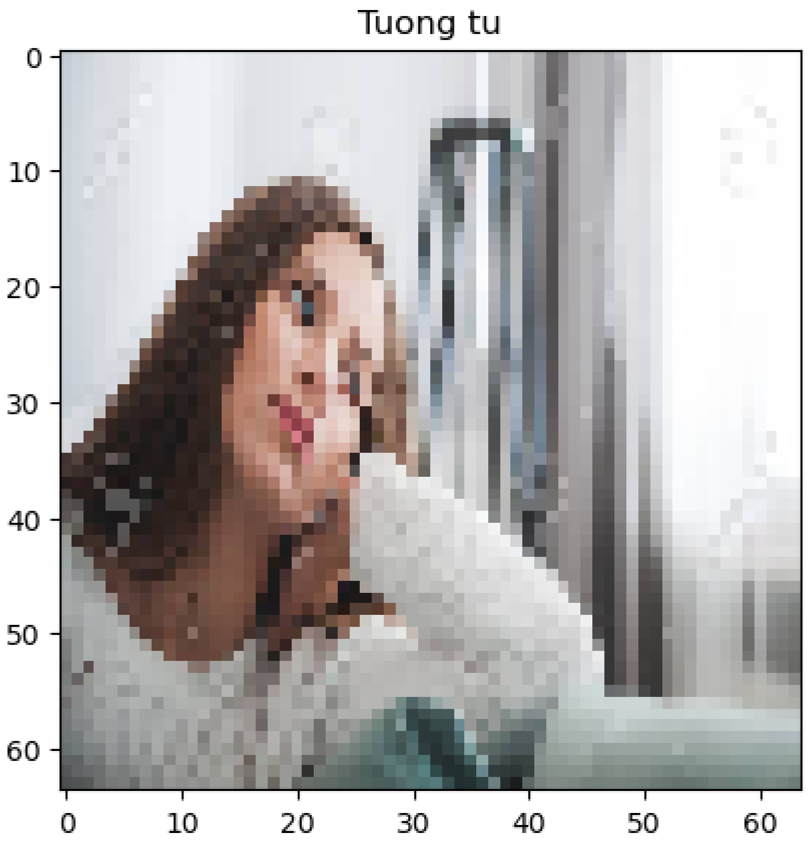


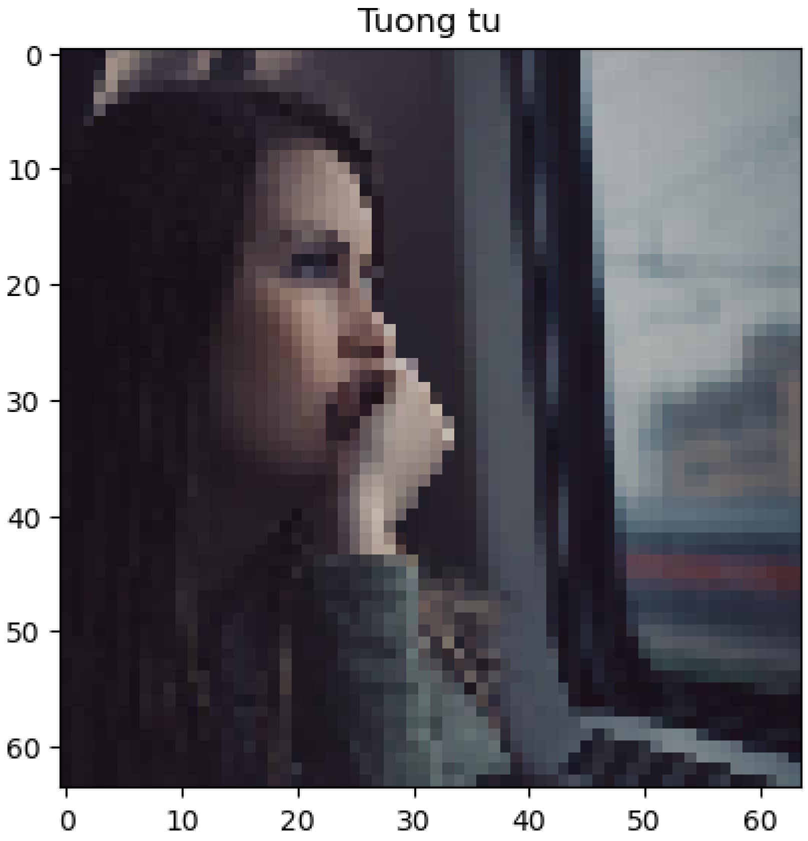
Tuong tu



Tuong tu




Tuong tu
0
10
20
30
40
50
60
0
10
20
30
40
50
60


Tuong tu
0
10
20
30
40
50
60
0
10
20
30
40
50
60

In [7]:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
import numpy as np
import pandas as pd

# Đọc tệp CSV
df = pd.read_csv('Test.csv')

# Chuyển đổi dữ liệu hình ảnh từ chuỗi thành mảng NumPy
data = df['Image'].apply(lambda x: np.array(eval(x))).values
labels = df['Label'].values

# Kết hợp tất cả các mảng hình ảnh thành một tensor
X_data = np.stack(data, axis=0)

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X_data, labels, test_size=

# Chuyển đổi dữ liệu hình ảnh thành tensors
X_train = tf.convert_to_tensor(X_train, dtype=tf.float32)
X_test = tf.convert_to_tensor(X_test, dtype=tf.float32)

# Chia tỷ lệ giá trị pixel vào khoảng [0, 1]
X_train /= 255.0
X_test /= 255.0

# Mã hóa one hot cho nhãn
label_binarizer = LabelBinarizer()
y_train_one_hot = label_binarizer.fit_transform(y_train)
y_test_one_hot = label_binarizer.transform(y_test)

# Xây dựng mô hình CNN
model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(128, activation='relu'),
    layers.Dense(3, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc

# Huấn luyện mô hình
history = model.fit(X_train, y_train_one_hot, epochs=100, batch_size = 32,  va
Epoch 1/100
5/5 [==============================] - 3s 225ms/step - loss: 1.0973 - accurac
y: 0.4000 - val_loss: 1.0273 - val_accuracy: 0.4390
Epoch 2/100
5/5 [==============================] - 1s 136ms/step - loss: 1.0595 - accurac
y: 0.4437 - val_loss: 1.0433 - val_accuracy: 0.4390
Epoch 3/100
5/5 [==============================] - 1s 126ms/step - loss: 1.0095 - accurac
y: 0.4313 - val_loss: 0.9439 - val_accuracy: 0.6098
Epoch 4/100
5/5 [==============================] - 1s 114ms/step - loss: 0.9946 - accurac
y: 0.5000 - val_loss: 1.2354 - val_accuracy: 0.4390
Epoch 5/100
5/5 [==============================] - 1s 135ms/step - loss: 0.9785 - accurac
y: 0.5125 - val_loss: 0.9924 - val_accuracy: 0.5366
Epoch 6/100
5/5 [==============================] - 1s 116ms/step - loss: 0.9518 - accurac
y: 0.5375 - val_loss: 1.0228 - val_accuracy: 0.5610
Epoch 7/100
5/5 [==============================] - 1s 116ms/step - loss: 0.8561 - accurac
y: 0.6062 - val_loss: 0.9719 - val_accuracy: 0.6341
Epoch 8/100
5/5 [==============================] - 1s 116ms/step - loss: 0.7865 - accurac
y: 0.6250 - val_loss: 0.9602 - val_accuracy: 0.6829
Epoch 9/100
5/5 [==============================] - 1s 132ms/step - loss: 0.7638 - accurac
y: 0.6438 - val_loss: 0.9841 - val_accuracy: 0.7073
Epoch 10/100
5/5 [==============================] - 1s 146ms/step - loss: 0.6804 - accurac
y: 0.7000 - val_loss: 0.9964 - val_accuracy: 0.6341
Epoch 11/100
5/5 [==============================] - 1s 135ms/step - loss: 0.6801 - accurac
y: 0.6938 - val_loss: 1.2255 - val_accuracy: 0.5366
Epoch 12/100
5/5 [==============================] - 1s 133ms/step - loss: 0.6024 - accurac
y: 0.7563 - val_loss: 1.4193 - val_accuracy: 0.5610
Epoch 13/100
5/5 [==============================] - 1s 141ms/step - loss: 0.6335 - accurac
y: 0.7312 - val_loss: 1.1151 - val_accuracy: 0.4878
Epoch 14/100
5/5 [==============================] - 1s 132ms/step - loss: 0.5386 - accurac
y: 0.7875 - val_loss: 1.3026 - val_accuracy: 0.5366
Epoch 15/100
5/5 [==============================] - 1s 125ms/step - loss: 0.5612 - accurac
y: 0.7625 - val_loss: 1.4745 - val_accuracy: 0.5366
Epoch 16/100
5/5 [==============================] - 1s 132ms/step - loss: 0.4984 - accurac
y: 0.8313 - val_loss: 1.2229 - val_accuracy: 0.4634
Epoch 17/100
5/5 [==============================] - 1s 129ms/step - loss: 0.4528 - accurac
y: 0.8438 - val_loss: 1.4825 - val_accuracy: 0.4634
Epoch 18/100
5/5 [==============================] - 1s 128ms/step - loss: 0.4514 - accurac
y: 0.8062 - val_loss: 1.3705 - val_accuracy: 0.4390
Epoch 19/100
5/5 [==============================] - 1s 151ms/step - loss: 0.3945 - accurac
y: 0.8500 - val_loss: 1.4239 - val_accuracy: 0.4878
```

```
Epoch 20/100
5/5 [==============================] - 1s 142ms/step - loss: 0.3110 - accurac
y: 0.9187 - val_loss: 1.3472 - val_accuracy: 0.4878
Epoch 21/100
5/5 [==============================] - 1s 141ms/step - loss: 0.2565 - accurac
y: 0.9438 - val_loss: 1.7026 - val_accuracy: 0.4390
Epoch 22/100
5/5 [==============================] - 1s 147ms/step - loss: 0.2413 - accurac
y: 0.9125 - val_loss: 1.8589 - val_accuracy: 0.5122
Epoch 23/100
5/5 [==============================] - 1s 133ms/step - loss: 0.2828 - accurac
y: 0.9125 - val_loss: 1.6527 - val_accuracy: 0.4390
Epoch 24/100
5/5 [==============================] - 1s 122ms/step - loss: 0.1948 - accurac
y: 0.9438 - val_loss: 2.0456 - val_accuracy: 0.5122
Epoch 25/100
5/5 [==============================] - 1s 118ms/step - loss: 0.1405 - accurac
y: 0.9563 - val_loss: 1.8327 - val_accuracy: 0.5122
Epoch 26/100
5/5 [==============================] - 1s 118ms/step - loss: 0.1569 - accurac
y: 0.9625 - val_loss: 2.0783 - val_accuracy: 0.4878
Epoch 27/100
5/5 [==============================] - 1s 146ms/step - loss: 0.1079 - accurac
y: 0.9812 - val_loss: 1.9472 - val_accuracy: 0.4634
Epoch 28/100
5/5 [==============================] - 1s 135ms/step - loss: 0.0863 - accurac
y: 0.9875 - val_loss: 2.3836 - val_accuracy: 0.4634
Epoch 29/100
5/5 [==============================] - 1s 122ms/step - loss: 0.1034 - accurac
y: 0.9688 - val_loss: 2.4973 - val_accuracy: 0.4878
Epoch 30/100
5/5 [==============================] - 1s 118ms/step - loss: 0.0616 - accurac
y: 0.9937 - val_loss: 2.4318 - val_accuracy: 0.4878
Epoch 31/100
5/5 [==============================] - 1s 118ms/step - loss: 0.0687 - accurac
y: 0.9937 - val_loss: 2.2836 - val_accuracy: 0.4390
Epoch 32/100
5/5 [==============================] - 1s 116ms/step - loss: 0.0529 - accurac
y: 0.9937 - val_loss: 2.8563 - val_accuracy: 0.4390
Epoch 33/100
5/5 [==============================] - 1s 121ms/step - loss: 0.0486 - accurac
y: 0.9875 - val_loss: 2.5608 - val_accuracy: 0.4878
Epoch 34/100
5/5 [==============================] - 1s 149ms/step - loss: 0.0441 - accurac
y: 0.9937 - val_loss: 2.8545 - val_accuracy: 0.4390
Epoch 35/100
5/5 [==============================] - 1s 163ms/step - loss: 0.0442 - accurac
y: 1.0000 - val_loss: 2.7371 - val_accuracy: 0.4390
Epoch 36/100
5/5 [==============================] - 1s 134ms/step - loss: 0.0362 - accurac
y: 1.0000 - val_loss: 2.9171 - val_accuracy: 0.4146
Epoch 37/100
5/5 [==============================] - 1s 137ms/step - loss: 0.0575 - accurac
y: 0.9875 - val_loss: 2.6566 - val_accuracy: 0.4390
Epoch 38/100
5/5 [==============================] - 1s 182ms/step - loss: 0.0544 - accurac
y: 0.9937 - val_loss: 2.9831 - val_accuracy: 0.4878
```

```
Epoch 39/100
5/5 [==============================] - 1s 129ms/step - loss: 0.1071 - accurac
y: 0.9625 - val_loss: 2.4921 - val_accuracy: 0.4146
Epoch 40/100
5/5 [==============================] - 1s 132ms/step - loss: 0.0389 - accurac
y: 0.9937 - val_loss: 2.4151 - val_accuracy: 0.5122
Epoch 41/100
5/5 [==============================] - 1s 147ms/step - loss: 0.0503 - accurac
y: 0.9875 - val_loss: 2.7260 - val_accuracy: 0.5122
Epoch 42/100
5/5 [==============================] - 1s 151ms/step - loss: 0.0298 - accurac
y: 1.0000 - val_loss: 2.6894 - val_accuracy: 0.4390
Epoch 43/100
5/5 [==============================] - 1s 137ms/step - loss: 0.0312 - accurac
y: 0.9937 - val_loss: 3.0442 - val_accuracy: 0.4634
Epoch 44/100
5/5 [==============================] - 1s 130ms/step - loss: 0.0322 - accurac
y: 0.9937 - val_loss: 3.1441 - val_accuracy: 0.4146
Epoch 45/100
5/5 [==============================] - 1s 126ms/step - loss: 0.0350 - accurac
y: 0.9937 - val_loss: 2.9932 - val_accuracy: 0.4390
Epoch 46/100
5/5 [==============================] - 1s 154ms/step - loss: 0.0243 - accurac
y: 0.9937 - val_loss: 3.4417 - val_accuracy: 0.4390
Epoch 47/100
5/5 [==============================] - 1s 146ms/step - loss: 0.0320 - accurac
y: 1.0000 - val_loss: 3.1952 - val_accuracy: 0.4146
Epoch 48/100
5/5 [==============================] - 1s 150ms/step - loss: 0.0709 - accurac
y: 0.9812 - val_loss: 3.1400 - val_accuracy: 0.4634
Epoch 49/100
5/5 [==============================] - 1s 132ms/step - loss: 0.0869 - accurac
y: 0.9688 - val_loss: 2.7979 - val_accuracy: 0.4634
Epoch 50/100
5/5 [==============================] - 1s 126ms/step - loss: 0.0359 - accurac
y: 0.9937 - val_loss: 3.4011 - val_accuracy: 0.4634
Epoch 51/100
5/5 [==============================] - 1s 134ms/step - loss: 0.0387 - accurac
y: 0.9875 - val_loss: 2.7101 - val_accuracy: 0.4390
Epoch 52/100
5/5 [==============================] - 1s 126ms/step - loss: 0.0338 - accurac
y: 0.9937 - val_loss: 3.2285 - val_accuracy: 0.4634
Epoch 53/100
5/5 [==============================] - 1s 128ms/step - loss: 0.0207 - accurac
y: 1.0000 - val_loss: 2.8798 - val_accuracy: 0.4634
Epoch 54/100
5/5 [==============================] - 1s 132ms/step - loss: 0.0193 - accurac
y: 1.0000 - val_loss: 3.0231 - val_accuracy: 0.4390
Epoch 55/100
5/5 [==============================] - 1s 116ms/step - loss: 0.0080 - accurac
y: 1.0000 - val_loss: 3.4411 - val_accuracy: 0.4634
Epoch 56/100
5/5 [==============================] - 1s 128ms/step - loss: 0.0092 - accurac
y: 1.0000 - val_loss: 3.5225 - val_accuracy: 0.4390
Epoch 57/100
5/5 [==============================] - 1s 150ms/step - loss: 0.0108 - accurac
y: 1.0000 - val_loss: 3.4604 - val_accuracy: 0.3902
```

```
Epoch 58/100
5/5 [==============================] - 1s 127ms/step - loss: 0.0056 - accurac
y: 1.0000 - val_loss: 3.5102 - val_accuracy: 0.4634
```

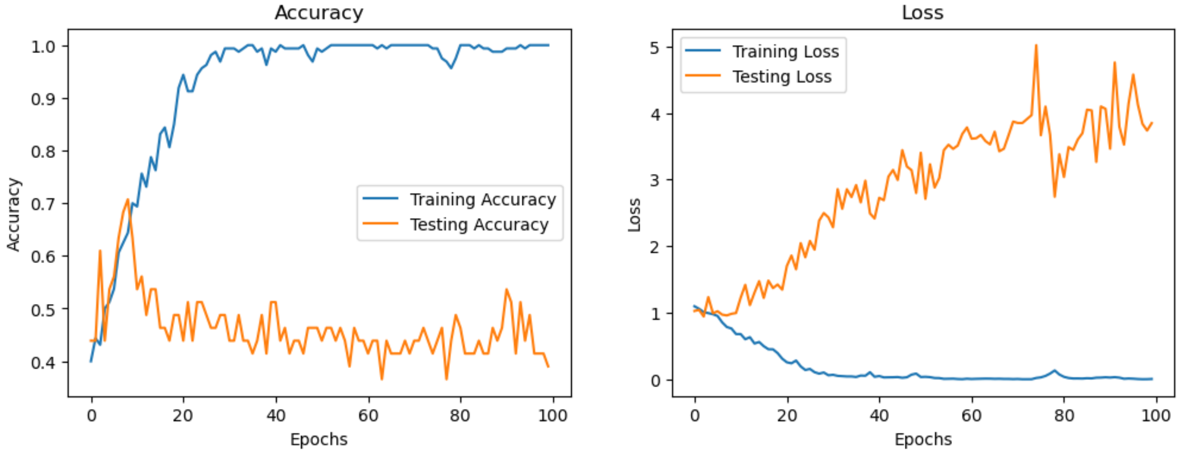
```
Epoch 59/100
5/5 [==============================] - 1s 115ms/step - loss: 0.0033 - accurac
y: 1.0000 - val_loss: 3.6850 - val_accuracy: 0.4390
Epoch 60/100
5/5 [==============================] - 1s 124ms/step - loss: 0.0099 - accurac
y: 1.0000 - val_loss: 3.7817 - val_accuracy: 0.4390
Epoch 61/100
5/5 [==============================] - 1s 127ms/step - loss: 0.0059 - accurac
y: 1.0000 - val_loss: 3.6137 - val_accuracy: 0.4146
Epoch 62/100
5/5 [==============================] - 1s 138ms/step - loss: 0.0083 - accurac
y: 1.0000 - val_loss: 3.6194 - val_accuracy: 0.4390
Epoch 63/100
5/5 [==============================] - 1s 134ms/step - loss: 0.0099 - accurac
y: 0.9937 - val_loss: 3.6691 - val_accuracy: 0.4390
Epoch 64/100
5/5 [==============================] - 1s 163ms/step - loss: 0.0114 - accurac
y: 1.0000 - val_loss: 3.5782 - val_accuracy: 0.3659
Epoch 65/100
5/5 [==============================] - 1s 143ms/step - loss: 0.0109 - accurac
y: 0.9937 - val_loss: 3.5273 - val_accuracy: 0.4390
Epoch 66/100
5/5 [==============================] - 1s 165ms/step - loss: 0.0083 - accurac
y: 1.0000 - val_loss: 3.7183 - val_accuracy: 0.4146
Epoch 67/100
5/5 [==============================] - 1s 180ms/step - loss: 0.0097 - accurac
y: 1.0000 - val_loss: 3.4239 - val_accuracy: 0.4146
Epoch 68/100
5/5 [==============================] - 1s 137ms/step - loss: 0.0069 - accurac
y: 1.0000 - val_loss: 3.4628 - val_accuracy: 0.4146
Epoch 69/100
5/5 [==============================] - 1s 116ms/step - loss: 0.0070 - accurac
y: 1.0000 - val_loss: 3.6633 - val_accuracy: 0.4390
Epoch 70/100
5/5 [==============================] - 1s 126ms/step - loss: 0.0044 - accurac
y: 1.0000 - val_loss: 3.8695 - val_accuracy: 0.4146
Epoch 71/100
5/5 [==============================] - 1s 143ms/step - loss: 0.0064 - accurac
y: 1.0000 - val_loss: 3.8492 - val_accuracy: 0.4390
Epoch 72/100
5/5 [==============================] - 1s 140ms/step - loss: 0.0024 - accurac
y: 1.0000 - val_loss: 3.8485 - val_accuracy: 0.4390
Epoch 73/100
5/5 [==============================] - 1s 137ms/step - loss: 0.0025 - accurac
y: 1.0000 - val_loss: 3.9079 - val_accuracy: 0.4634
Epoch 74/100
5/5 [==============================] - 1s 142ms/step - loss: 0.0028 - accurac
y: 1.0000 - val_loss: 3.9674 - val_accuracy: 0.4390
Epoch 75/100
5/5 [==============================] - 1s 134ms/step - loss: 0.0212 - accurac
y: 0.9937 - val_loss: 5.0171 - val_accuracy: 0.4146
Epoch 76/100
5/5 [==============================] - 1s 137ms/step - loss: 0.0305 - accurac
y: 0.9937 - val_loss: 3.6646 - val_accuracy: 0.4390
Epoch 77/100
5/5 [==============================] - 1s 143ms/step - loss: 0.0518 - accurac
y: 0.9750 - val_loss: 4.0946 - val_accuracy: 0.4634
```

```
Epoch 78/100
5/5 [==============================] - 1s 159ms/step - loss: 0.0873 - accurac
y: 0.9688 - val_loss: 3.6701 - val_accuracy: 0.3659
Epoch 79/100
5/5 [==============================] - 1s 201ms/step - loss: 0.1341 - accurac
y: 0.9563 - val_loss: 2.7415 - val_accuracy: 0.4390
Epoch 80/100
5/5 [==============================] - 1s 176ms/step - loss: 0.0731 - accurac
y: 0.9750 - val_loss: 3.3793 - val_accuracy: 0.4878
Epoch 81/100
5/5 [==============================] - 1s 164ms/step - loss: 0.0383 - accurac
y: 1.0000 - val_loss: 3.0376 - val_accuracy: 0.4634
Epoch 82/100
5/5 [==============================] - 1s 165ms/step - loss: 0.0209 - accurac
y: 1.0000 - val_loss: 3.4869 - val_accuracy: 0.4146
Epoch 83/100
5/5 [==============================] - 1s 170ms/step - loss: 0.0135 - accurac
y: 1.0000 - val_loss: 3.4451 - val_accuracy: 0.4146
Epoch 84/100
5/5 [==============================] - 1s 162ms/step - loss: 0.0140 - accurac
y: 0.9937 - val_loss: 3.6012 - val_accuracy: 0.4146
Epoch 85/100
5/5 [==============================] - 1s 211ms/step - loss: 0.0124 - accurac
y: 1.0000 - val_loss: 3.6961 - val_accuracy: 0.4390
Epoch 86/100
5/5 [==============================] - 1s 162ms/step - loss: 0.0173 - accurac
y: 0.9937 - val_loss: 4.0485 - val_accuracy: 0.4146
Epoch 87/100
5/5 [==============================] - 1s 172ms/step - loss: 0.0151 - accurac
y: 0.9937 - val_loss: 4.0375 - val_accuracy: 0.4146
Epoch 88/100
5/5 [==============================] - 1s 164ms/step - loss: 0.0253 - accurac
y: 0.9875 - val_loss: 3.2646 - val_accuracy: 0.4634
Epoch 89/100
5/5 [==============================] - 1s 151ms/step - loss: 0.0269 - accurac
y: 0.9875 - val_loss: 4.0964 - val_accuracy: 0.4390
Epoch 90/100
5/5 [==============================] - 1s 167ms/step - loss: 0.0319 - accurac
y: 0.9875 - val_loss: 4.0613 - val_accuracy: 0.4634
Epoch 91/100
5/5 [==============================] - 1s 180ms/step - loss: 0.0278 - accurac
y: 0.9937 - val_loss: 3.4629 - val_accuracy: 0.5366
Epoch 92/100
5/5 [==============================] - 1s 184ms/step - loss: 0.0335 - accurac
y: 0.9937 - val_loss: 4.7573 - val_accuracy: 0.5122
Epoch 93/100
5/5 [==============================] - 1s 172ms/step - loss: 0.0261 - accurac
y: 0.9937 - val_loss: 3.7925 - val_accuracy: 0.4146
Epoch 94/100
5/5 [==============================] - 1s 188ms/step - loss: 0.0093 - accurac
y: 1.0000 - val_loss: 3.5238 - val_accuracy: 0.5122
Epoch 95/100
5/5 [==============================] - 1s 192ms/step - loss: 0.0133 - accurac
y: 0.9937 - val_loss: 4.1426 - val_accuracy: 0.4390
Epoch 96/100
5/5 [==============================] - 1s 188ms/step - loss: 0.0102 - accurac
y: 1.0000 - val_loss: 4.5756 - val_accuracy: 0.4878
```

```
Epoch 97/100
5/5 [==============================] - 1s 195ms/step - loss: 0.0054 - accurac
y: 1.0000 - val_loss: 4.1273 - val_accuracy: 0.4146
Epoch 98/100
5/5 [==============================] - 1s 174ms/step - loss: 0.0026 - accurac
y: 1.0000 - val_loss: 3.8354 - val_accuracy: 0.4146
Epoch 99/100
5/5 [==============================] - 1s 213ms/step - loss: 0.0032 - accurac
y: 1.0000 - val_loss: 3.7355 - val_accuracy: 0.4146
Epoch 100/100
5/5 [==============================] - 1s 188ms/step - loss: 0.0061 - accurac
y: 1.0000 - val_loss: 3.8472 - val_accuracy: 0.3902
```

In [8]:

```
import matplotlib.pyplot as plt

# Lấy giá trị accuracy và loss từ biến "history"
train_accuracy = history.history['accuracy']
test_accuracy = history.history['val_accuracy']
train_loss = history.history['loss']
test_loss = history.history['val_loss']

# Vẽ biểu đồ

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(train_accuracy, label='Training Accuracy')
plt.plot(test_accuracy, label='Testing Accuracy')
plt.legend()
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')

plt.subplot(1, 2, 2)
plt.plot(train_loss, label='Training Loss')
plt.plot(test_loss, label='Testing Loss')
plt.legend()
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')

plt.show()

# Đánh giá mô hình trên tập kiểm tra
test_loss, test_accuracy = model.evaluate(X_test, y_test_one_hot)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
```

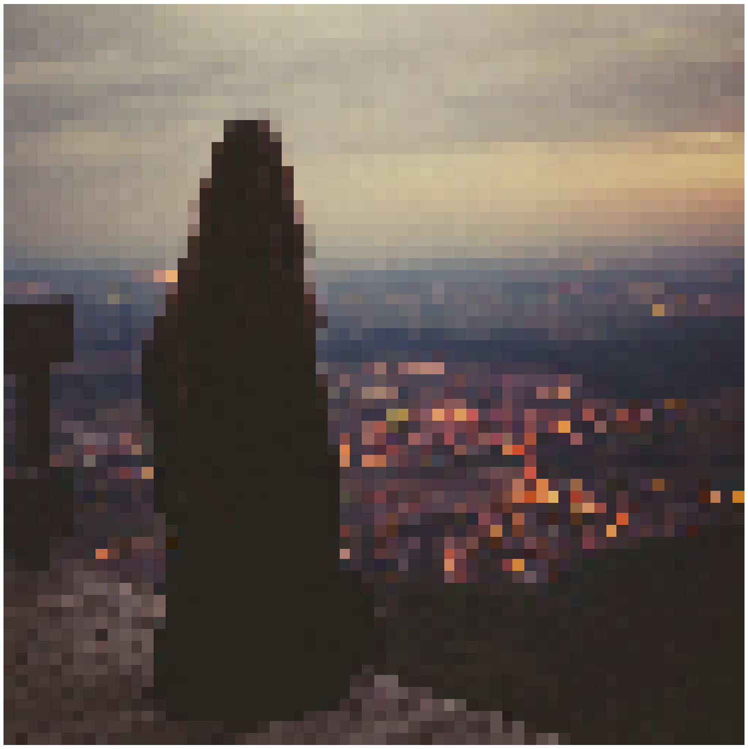


```
2/2 [==============================] - 0s 16ms/step - loss: 3.8472 - accurac
y: 0.3902
Test Accuracy: 39.02%
```

In [12]:

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

# Đọc ảnh từ máy tính
image_path = "C:/Users/Admin/Pictures/demo3.jpg"
image = cv2.imread(image_path)

resized_image = cv2.resize(image, (64, 64))
preprocessed_image = resized_image / 255.0

# Chuyển đổi ảnh thành tensor
input_image = tf.convert_to_tensor(preprocessed_image, dtype=tf.float32)

# Dự đoán bằng mô hình đã huấn luyện
predictions = model.predict(tf.expand_dims(input_image, axis=0))

# Đưa ra kết quả
class_labels = ["Tuong tu", "That tinh", "Dang yeu"]
predicted_class_index = np.argmax(predictions)
predicted_class_label = class_labels[predicted_class_index]

# Hiển thị ma trận ảnh
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))  # Chuyển đổi màu t
plt.axis('off')  # Ẩn trục
plt.show()

# Hiển thị xác suất cho từng nhãn
for i in range(len(class_labels)):
    probability = predictions[0][i] * 100
    print(f"{class_labels[i]}: {probability:.2f}%")

print(f"Predicted class: {predicted_class_label}")
```

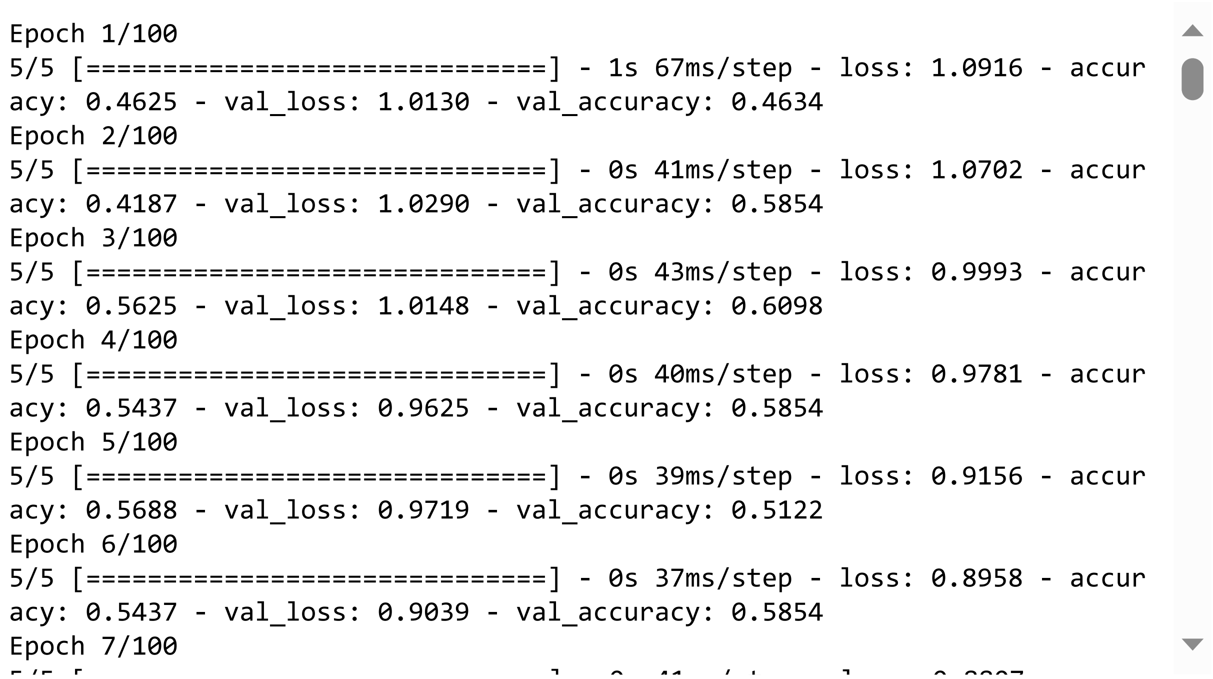
```
1/1 [==============================] - 0s 116ms/step
```



```
Tuong tu: 0.00%
That tinh: 100.00%
Dang yeu: 0.00%
Predicted class: That tinh
```

In [13]:

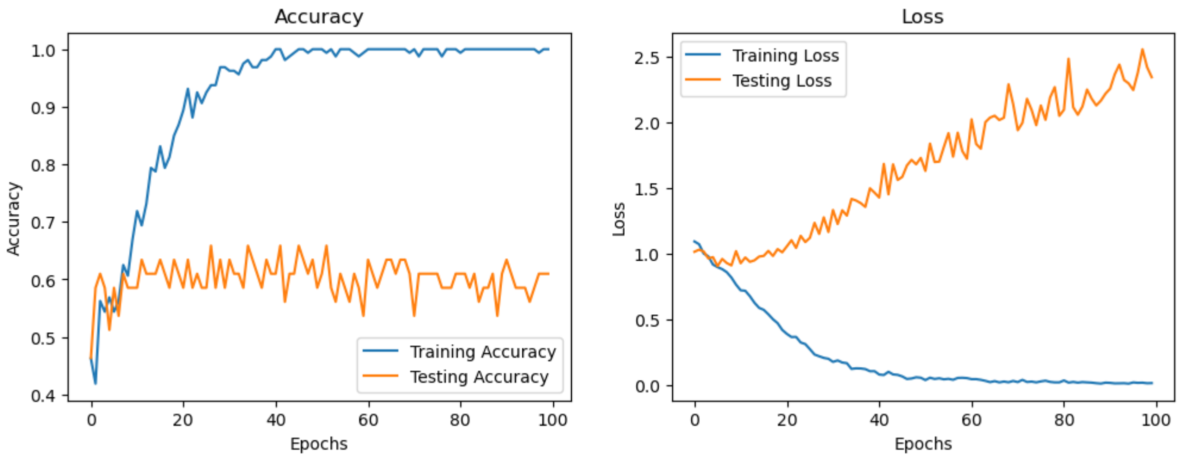
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
import numpy as np
import pandas as pd

# Đọc tệp CSV
df = pd.read_csv('Test.csv')

# Chuyển đổi dữ liệu hình ảnh từ chuỗi thành mảng NumPy
data = df['Image'].apply(lambda x: np.array(eval(x))).values
labels = df['Label'].values

# Kết hợp tất cả các mảng hình ảnh thành một tensor
X_data = np.stack(data, axis=0)

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X_data, labels, test_size=

# Chuyển đổi dữ liệu hình ảnh thành tensors
X_train = tf.convert_to_tensor(X_train, dtype=tf.float32)
X_test = tf.convert_to_tensor(X_test, dtype=tf.float32)

# Chia tỷ lệ giá trị pixel vào khoảng [0, 1]
X_train /= 255.0
X_test /= 255.0

# Mã hóa one hot cho nhãn
label_binarizer = LabelBinarizer()
y_train_one_hot = label_binarizer.fit_transform(y_train)
y_test_one_hot = label_binarizer.transform(y_test)

# Xây dựng mô hình CNN
model = keras.Sequential([
    layers.Conv2D(8, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(16, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dropout(0.4),
    layers.Dense(16, activation='relu'),
    layers.Dense(3, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc

# Huấn luyện mô hình
history = model.fit(X_train, y_train_one_hot, epochs=100, batch_size = 32, val
```



```
Epoch 1/100
5/5 [==============================] - 1s 67ms/step - loss: 1.0916 - accur
acy: 0.4625 - val_loss: 1.0130 - val_accuracy: 0.4634
Epoch 2/100
5/5 [==============================] - 0s 41ms/step - loss: 1.0702 - accur
acy: 0.4187 - val_loss: 1.0290 - val_accuracy: 0.5854
Epoch 3/100
5/5 [==============================] - 0s 43ms/step - loss: 0.9993 - accur
acy: 0.5625 - val_loss: 1.0148 - val_accuracy: 0.6098
Epoch 4/100
5/5 [==============================] - 0s 40ms/step - loss: 0.9781 - accur
acy: 0.5437 - val_loss: 0.9625 - val_accuracy: 0.5854
Epoch 5/100
5/5 [==============================] - 0s 39ms/step - loss: 0.9156 - accur
acy: 0.5688 - val_loss: 0.9719 - val_accuracy: 0.5122
Epoch 6/100
5/5 [==============================] - 0s 37ms/step - loss: 0.8958 - accur
acy: 0.5437 - val_loss: 0.9039 - val_accuracy: 0.5854
Epoch 7/100
5/5 [==============================] - 0s 41ms/step - loss: 0.8807
```

In [14]:

```python
import matplotlib.pyplot as plt

# Lấy giá trị accuracy và loss từ biến "history"
train_accuracy = history.history['accuracy']
test_accuracy = history.history['val_accuracy']
train_loss = history.history['loss']
test_loss = history.history['val_loss']


# Vẽ biểu đồ
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(train_accuracy, label='Training Accuracy')
plt.plot(test_accuracy, label='Testing Accuracy')
plt.legend()
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')


plt.subplot(1, 2, 2)
plt.plot(train_loss, label='Training Loss')
plt.plot(test_loss, label='Testing Loss')
plt.legend()
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')

plt.show()

# Đánh giá mô hình trên tập kiểm tra
test_loss, test_accuracy = model.evaluate(X_test, y_test_one_hot)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
```

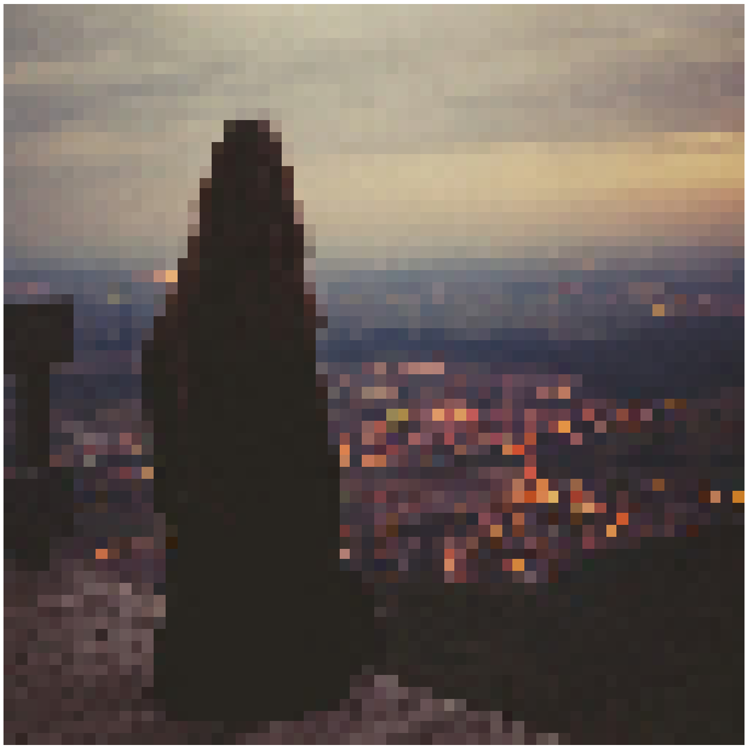


```
2/2 [==============================] - 0s 8ms/step - loss: 2.3440 - accuracy:
0.6098
Test Accuracy: 60.98%
```

In [15]:

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

# Đọc ảnh từ máy tính
image_path = "C:/Users/Admin/Pictures/demo3.jpg"
image = cv2.imread(image_path)

resized_image = cv2.resize(image, (64, 64))
preprocessed_image = resized_image / 255.0

# Chuyển đổi ảnh thành tensor
input_image = tf.convert_to_tensor(preprocessed_image, dtype=tf.float32)

# Dự đoán bằng mô hình đã huấn luyện
predictions = model.predict(tf.expand_dims(input_image, axis=0))

# Đưa ra kết quả
class_labels = ["Tuong tu", "That tinh", "Dang yeu"]
predicted_class_index = np.argmax(predictions)
predicted_class_label = class_labels[predicted_class_index]

# Hiển thị ma trận ảnh
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))  # Chuyển đổi màu t
plt.axis('off')  # Ẩn trục
plt.show()

# Hiển thị xác suất cho từng nhãn
for i in range(len(class_labels)):
    probability = predictions[0][i] * 100
    print(f"{class_labels[i]}: {probability:.2f}%")

print(f"Predicted class: {predicted_class_label}")
```

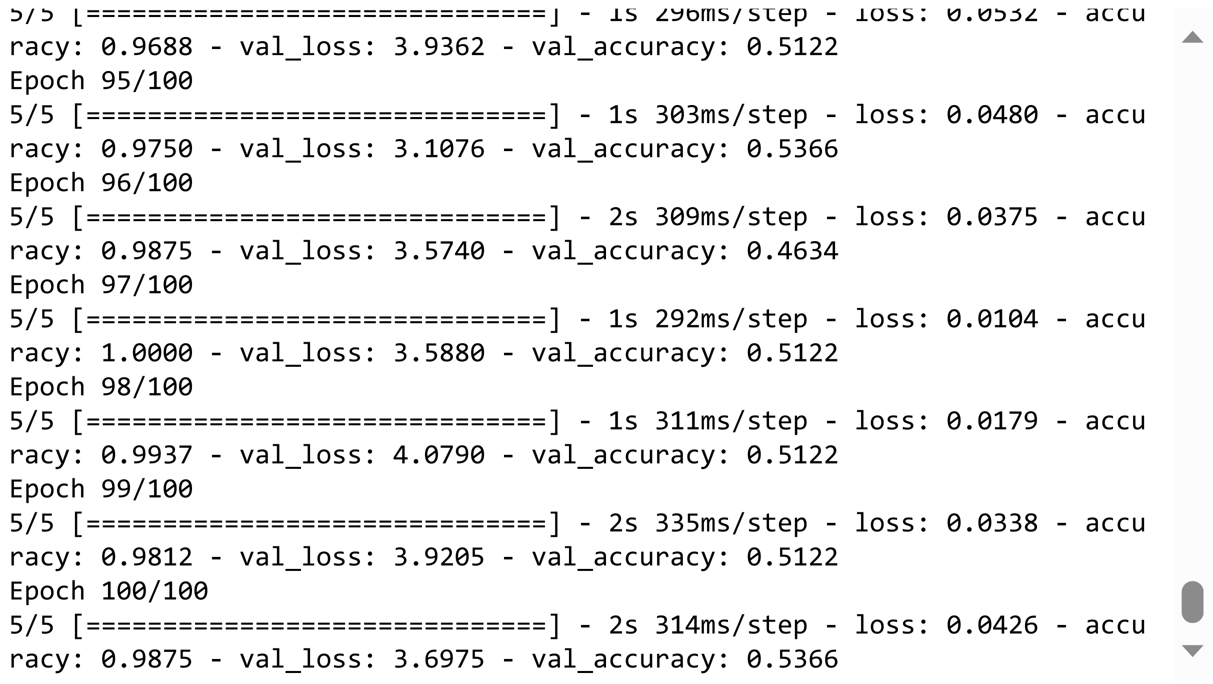
```
1/1 [==============================] - 0s 64ms/step
```



```
Tuong tu: 0.05%
That tinh: 99.94%
Dang yeu: 0.00%
Predicted class: That tinh
```

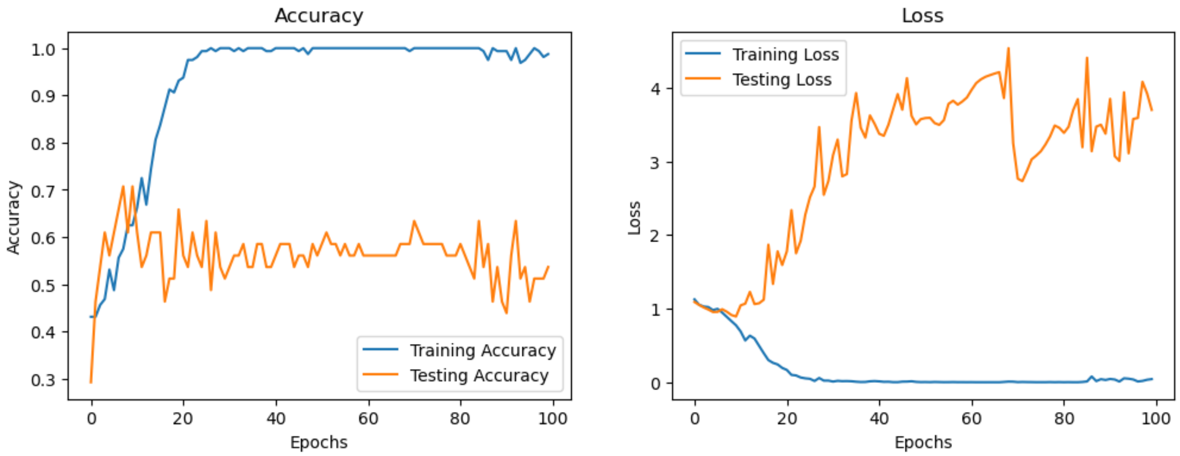
In [16]:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
import numpy as np
import pandas as pd

# Đọc tệp CSV
df = pd.read_csv('Test.csv')

# Chuyển đổi dữ liệu hình ảnh từ chuỗi thành mảng NumPy
data = df['Image'].apply(lambda x: np.array(eval(x))).values
labels = df['Label'].values

# Kết hợp tất cả các mảng hình ảnh thành một tensor
X_data = np.stack(data, axis=0)

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X_data, labels, test_size=

# Chuyển đổi dữ liệu hình ảnh thành tensors
X_train = tf.convert_to_tensor(X_train, dtype=tf.float32)
X_test = tf.convert_to_tensor(X_test, dtype=tf.float32)

# Chia tỷ lệ giá trị pixel vào khoảng [0, 1]
X_train /= 255.0
X_test /= 255.0

# Mã hóa one hot cho nhãn
label_binarizer = LabelBinarizer()
y_train_one_hot = label_binarizer.fit_transform(y_train)
y_test_one_hot = label_binarizer.transform(y_test)

# Xây dựng mô hình CNN
model = keras.Sequential([
    layers.Conv2D(64, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(256, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dropout(0.3),  # Giảm dropout để giảm khả năng overfitting
    layers.Dense(256, activation='relu'),
    layers.Dense(3, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc

# Huấn luyện mô hình
history = model.fit(X_train, y_train_one_hot, epochs=100, batch_size = 32, val
```



```
5/5 [==============================] - 1s 296ms/step - loss: 0.0532 - accu
racy: 0.9688 - val_loss: 3.9362 - val_accuracy: 0.5122
Epoch 95/100
5/5 [==============================] - 1s 303ms/step - loss: 0.0480 - accu
racy: 0.9750 - val_loss: 3.1076 - val_accuracy: 0.5366
Epoch 96/100
5/5 [==============================] - 2s 309ms/step - loss: 0.0375 - accu
racy: 0.9875 - val_loss: 3.5740 - val_accuracy: 0.4634
Epoch 97/100
5/5 [==============================] - 1s 292ms/step - loss: 0.0104 - accu
racy: 1.0000 - val_loss: 3.5880 - val_accuracy: 0.5122
Epoch 98/100
5/5 [==============================] - 1s 311ms/step - loss: 0.0179 - accu
racy: 0.9937 - val_loss: 4.0790 - val_accuracy: 0.5122
Epoch 99/100
5/5 [==============================] - 2s 335ms/step - loss: 0.0338 - accu
racy: 0.9812 - val_loss: 3.9205 - val_accuracy: 0.5122
Epoch 100/100
5/5 [==============================] - 2s 314ms/step - loss: 0.0426 - accu
racy: 0.9875 - val_loss: 3.6975 - val_accuracy: 0.5366
```

In [17]:
```
import matplotlib.pyplot as plt

# Lấy giá trị accuracy và loss từ biến "history"
train_accuracy = history.history['accuracy']
test_accuracy = history.history['val_accuracy']
train_loss = history.history['loss']
test_loss = history.history['val_loss']


# Vẽ biểu đồ
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(train_accuracy, label='Training Accuracy')
plt.plot(test_accuracy, label='Testing Accuracy')
plt.legend()
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')

plt.subplot(1, 2, 2)
plt.plot(train_loss, label='Training Loss')
plt.plot(test_loss, label='Testing Loss')
plt.legend()
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')

plt.show()

# Đánh giá mô hình trên tập kiểm tra
test_loss, test_accuracy = model.evaluate(X_test, y_test_one_hot)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
```

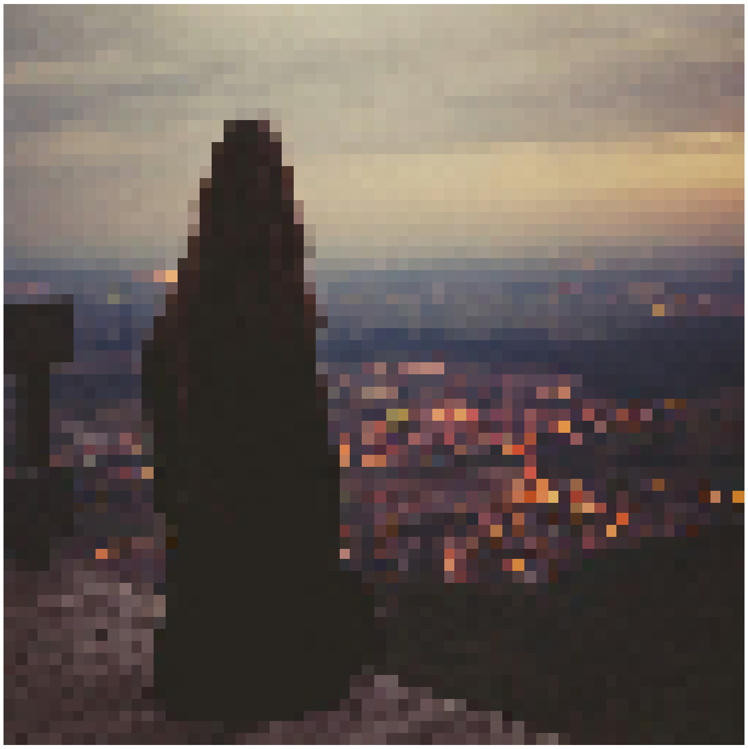


```
2/2 [==============================] - 0s 28ms/step - loss: 3.6975 - accurac
y: 0.5366
Test Accuracy: 53.66%
```

In [18]:

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

# Đọc ảnh từ máy tính
image_path = "C:/Users/Admin/Pictures/demo3.jpg"
image = cv2.imread(image_path)

resized_image = cv2.resize(image, (64, 64))
preprocessed_image = resized_image / 255.0

# Chuyển đổi ảnh thành tensor
input_image = tf.convert_to_tensor(preprocessed_image, dtype=tf.float32)

# Dự đoán bằng mô hình đã huấn luyện
predictions = model.predict(tf.expand_dims(input_image, axis=0))

# Đưa ra kết quả
class_labels = ["Tuong tu", "That tinh", "Dang yeu"]
predicted_class_index = np.argmax(predictions)
predicted_class_label = class_labels[predicted_class_index]

# Hiển thị ma trận ảnh
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))  # Chuyển đổi màu t
plt.axis('off')  # Ẩn trục
plt.show()

# Hiển thị xác suất cho từng nhãn
for i in range(len(class_labels)):
    probability = predictions[0][i] * 100
    print(f"{class_labels[i]}: {probability:.2f}%")

print(f"Predicted class: {predicted_class_label}")
```

```
1/1 [==============================] - 0s 64ms/step
```



```
Tuong tu: 0.00%
That tinh: 100.00%
Dang yeu: 0.00%
Predicted class: That tinh
```

```
In [ ]:
```