Trần Xuân Triển – B20DCCN691 – 04 – 01

1. Tiền xử lý dữ liệu

- Tạo DataFrame pandas với hình ảnh, nhãn. Lưu DataFrame vào tệp CSV.

```
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/DELL/Desktop/PTHTM/Test"

for label in ["That tinh", "Dang yeu","Tuong tu"]:
    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):
            if filename.endswith(".jpg") or filename.endswith(".PNG"):
                image_path = os.path.join(label_folder, filename)  # Đường dẫn đến ảnh

                # 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()
```

```
# Chuyển danh sách thành mảng NumPy
data = np.array(data)
labels = np.array(labels)

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

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

```
(187, 2)
```

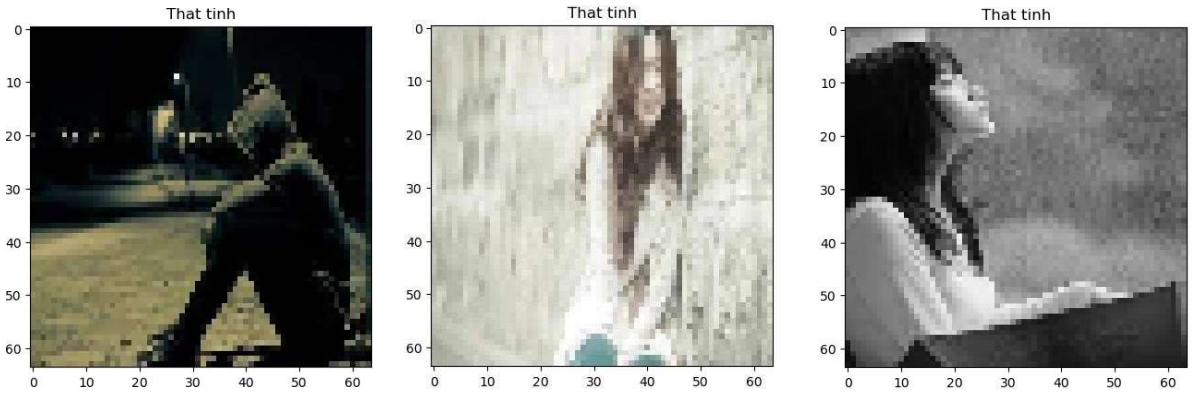
- Hiển thị 5 hàng đầu tiên trong DataFrame.

```
import pandas as pd

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

```
                                               Image      Label
0  [[[1, 1, 1], [1, 1, 1], [1, 1, 1], [1, 1, 1], ...  That tinh
1  [[[203, 205, 191], [197, 199, 185], [192, 194,...  That tinh
2  [[[220, 220, 230], [220, 220, 230], [219, 219,...  That tinh
3  [[[254, 254, 254], [254, 254, 254], [254, 254,...  That tinh
4  [[[122, 122, 122], [115, 115, 115], [122, 122,...  That tinh
```

- Hiển thị 5 hình ảnh đầu tiên trong DataFrame.

```
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()
```



2. Xây dựng model.

2.1 : Model 1

```
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=0.2, random_state=32)

# 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=['accuracy'])

# Huấn luyện mô hình
history = model.fit(X_train, y_train_one_hot, epochs=15, batch_size = 10,  validation_data=(X_test, y_test_one_hot))
```

2.2 : Model 2

```
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=0.2, random_state=32)

# 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=['accuracy'])

# Huấn luyện mô hình
history = model.fit(X_train, y_train_one_hot, epochs=15, batch_size = 10, validation_data=(X_test, y_test_one_hot))
```

2.3 : Model 3

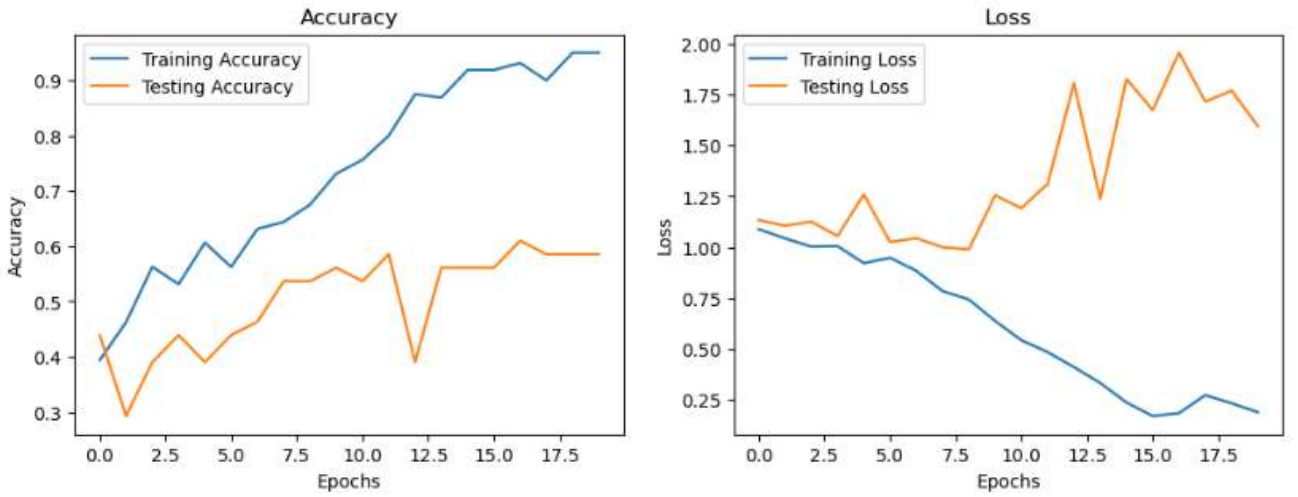
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
import numpy as np
import pandas as pd

# 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=0.2, random_state=32)

# 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=['accuracy'])

# Huấn luyện mô hình
history = model.fit(X_train, y_train_one_hot, epochs=15, batch_size = 10, validation_data=(X_test, y_test_one_hot))
```

3. Đánh giá

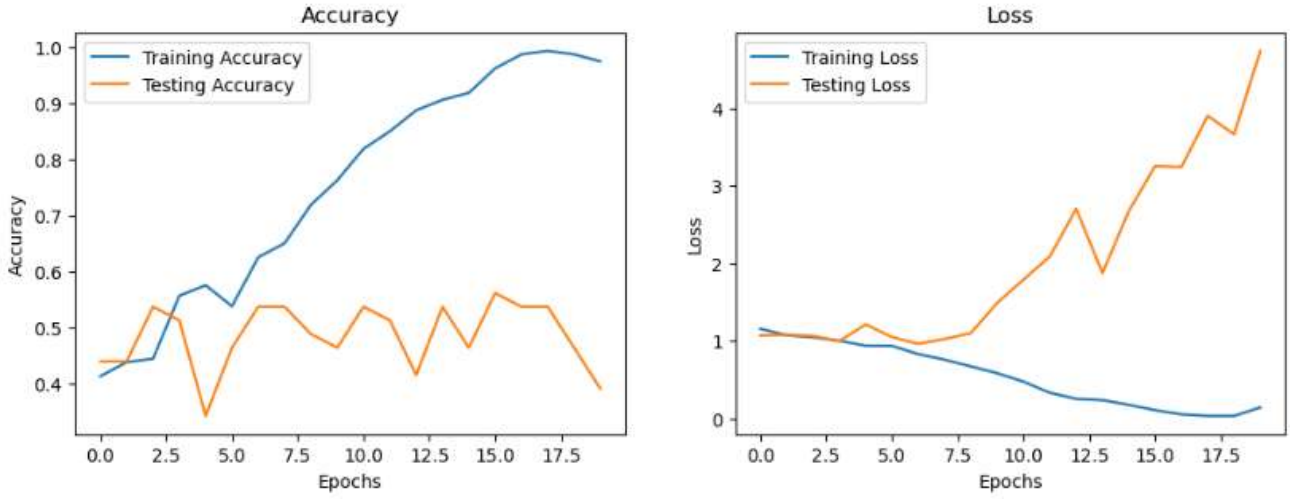
```
import matplotlib.pyplot as plt

# 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}%")
```

3.1 : Model 1



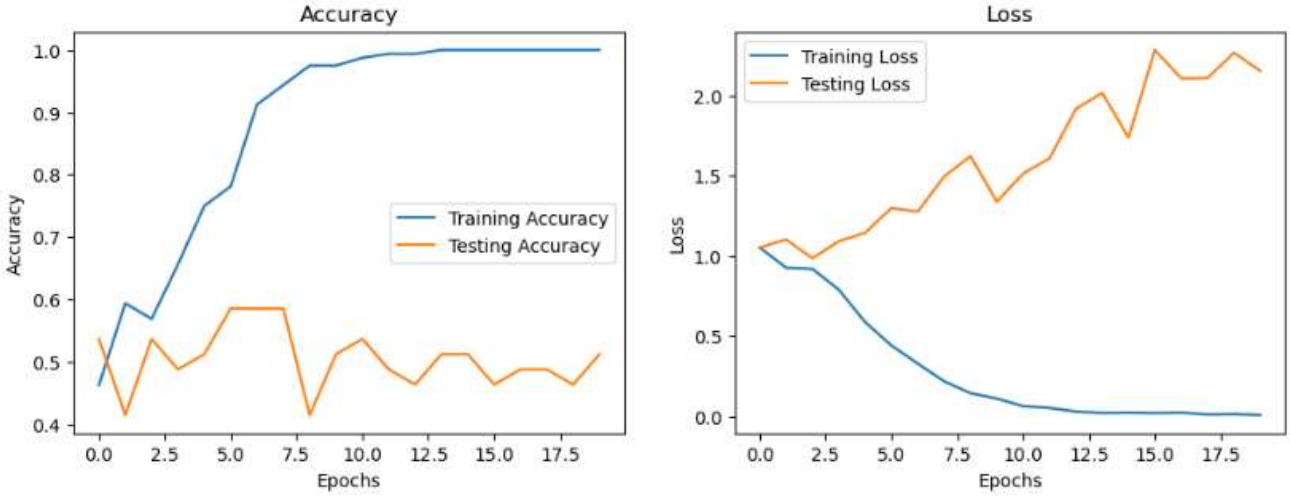

```
2/2 [==============================] - 0s 16ms/step - loss: 1.5958 - accuracy: 0.5854
Test Accuracy: 58.54%
```

3.2 : Model 2




```
2/2 [==============================] - 0s 37ms/step - loss: 4.7373 - accuracy: 0.3902
Test Accuracy: 39.02%
```

3.3 : Model 3




```
2/2 [==============================] - 0s 19ms/step - loss: 2.1554 - accuracy: 0.5122
Test Accuracy: 51.22%
```

***Nhận xét:**

- Model 1: Model 1 có 3 lớp convolutional, mỗi lớp có 32, 64 và 128 neuron. Số lượng lớp và neuron tương đối ít, do đó model này có thể học tập được các đặc trưng cơ bản của dữ liệu ảnh, nhưng không thể học tập được các đặc trưng phức tạp. Dropout được sử dụng với tỷ lệ 0.5, điều này giúp ngăn ngừa overfitting hiệu quả.
    - Độ chính xác (accuracy) khá thấp (0.58), tức là mô hình dự đoán đúng khoảng 58.54% của các mẫu trên tập kiểm tra.
    - Hàm mất mát (loss) cao (1.5958), cho thấy mô hình đang có nhiều dự đoán sai và không khớp tốt với dữ liệu kiểm tra. Điều này có thể cho thấy mô hình này có dấu hiệu overfitting.
- Model 2: Model 2 có 3 lớp convolutional, mỗi lớp có 64, 128 và 256 neuron. Số lượng lớp và neuron nhiều hơn model 1, do đó model này có khả năng học tập được các đặc trưng phức tạp hơn của dữ liệu ảnh. Dropout được sử dụng với tỷ lệ 0.3, điều này giúp giảm khả năng overfitting, nhưng vẫn cho phép mô hình học tập được các đặc trưng quan trọng của dữ liệu ảnh.
    - Độ chính xác (accuracy) rất thấp(0.3902), thấp hơn rất nhiều so với Model 1.
    - Hàm mất mát (loss) rất cao (4.7373), cho thấy mô hình đang có nhiều sai số trong dự đoán. Mặc dù tỷ lệ dropout đã được giảm, mô hình này vẫn không hoạt động tốt trên dữ liệu kiểm tra.
- Model 3: Model 3 có 2 lớp convolutional, mỗi lớp có 8 và 16 neuron. Số lượng lớp và neuron ít nhất trong ba model, do đó model này chỉ có thể học tập được các đặc trưng cơ bản nhất của dữ liệu ảnh. Dropout được sử dụng với tỷ lệ 0.4, điều này có thể giúp ngăn ngừa overfitting, nhưng cũng có thể khiến mô hình không học tập được các đặc trưng quan trọng của dữ liệu ảnh.
    - Độ chính xác (accuracy) tương thấp (0.5122), thấp hơn 1 chút so với Model1 và cao hơn rất nhiều so với Model2.
    - Hàm mất mát (loss) rất cao (2.1554), cho thấy mô hình này đang đưa ra dự đoán rất không ổn, cao hơn rất nhiều so với 2 Model trước đó
- Như vậy, sự khác biệt về hiệu suất giữa ba mô hình có thể do nhiều yếu tố, bao gồm cấu trúc mô hình, tỷ lệ dropout, số lượng lớp và neurons, và phản ứng với tập dữ liệu cụ thể mà bạn đang sử dụng. Mô hình 1 có vẻ là lựa chọn tốt nhất trong số ba mô hình này dựa trên độ chính xác và hàm mất mát trên tập kiểm tra.