Họ và tên: Quách Xuân Phúc

MSV: B20DCCN513

8.

- Tạo các tệp cần thiết:

- Original: folder chứa dữ liệu gốc chưa qua xử lý.
- Processed: folder chứa dữ liệu đã qua xử lý.
- Base: folder chứa dữ liệu để huấn luyện, xác thực và kiểm thử

```
import os
import shutil
from pathlib import Path
from PIL import Image

# ------------------------------Tạo các tệp cần thiết ------------------------------------------------------------
def check_folder_exist(folder_path):
    if not os.path.exists(folder_path):
        os.mkdir(folder_path)

# Đường dẫn đến thư mục nơi bộ dữ liệu gốc đã được giải nén
original_dataset_dir = Path('C:/Users/PhucQuach/OneDrive/Desktop/TKHTTM/BaiTap8/DataCollection/Original')
# Đường dẫn đến thư mục nơi bộ dữ liệu đã qua xử lý
processed_dataset_dir = Path('C:/Users/PhucQuach/OneDrive/Desktop/TKHTTM/BaiTap8/DataCollection/Processed')
check_folder_exist(processed_dataset_dir)
# Thư mục nơi bạn sẽ lưu trữ bộ dữ liệu nhỏ của mình
base_dir = Path('C:/Users/PhucQuach/OneDrive/Desktop/TKHTTM/BaiTap8/DataCollection/Base')
check_folder_exist(base_dir)

processed_loving_dir = os.path.join(processed_dataset_dir, 'DangYeu')
check_folder_exist(processed_loving_dir)
processed_lovelorn_dir = os.path.join(processed_dataset_dir, 'ThatTinh')
check_folder_exist(processed_lovelorn_dir)
processed_lovesick_dir = os.path.join(processed_dataset_dir, 'TuongTu')
check_folder_exist(processed_lovesick_dir)
```

```
# Các thư mục cho các phần tách huấn luyện, xác thực và kiểm thử
train_dir = os.path.join(base_dir, 'train')
check_folder_exist(train_dir)
validation_dir = os.path.join(base_dir, 'validation')
check_folder_exist(validation_dir)
test_dir = os.path.join(base_dir, 'test')
check_folder_exist(test_dir)
```

```
# Thư mục với hình ảnh đang yêu huấn luyện
train_loving_dir = os.path.join(train_dir, 'DangYeu')
check_folder_exist(train_loving_dir)

# Thư mục với hình ảnh thất tình huấn luyện
train_lovelorn_dir = os.path.join(train_dir, 'ThatTinh')
check_folder_exist(train_lovelorn_dir)

# Thư mục với hình ảnh tương tư huấn luyện
train_lovesick_dir = os.path.join(train_dir, 'TuongTu')
check_folder_exist(train_lovesick_dir)

# Thư mục với hình ảnh đang yêu xác thực
validation_loving_dir = os.path.join(validation_dir, 'DangYeu')
check_folder_exist(validation_loving_dir)

# Thư mục với hình ảnh thất tình xác thực
validation_lovelorn_dir = os.path.join(validation_dir, 'ThatTinh')
check_folder_exist(validation_lovelorn_dir)

# Thư mục với hình ảnh tương tư xác thực
validation_lovesick_dir = os.path.join(validation_dir, 'TuongTu')
check_folder_exist(validation_lovesick_dir)
```

```
# Thư mục với hình ảnh đang yêu kiểm thử
test_loving_dir = os.path.join(test_dir, 'DangYeu')
check_folder_exist(test_loving_dir)

# Thư mục với hình ảnh thất tình kiểm thử
test_lovelorn_dir = os.path.join(test_dir, 'ThatTinh')
check_folder_exist(test_lovelorn_dir)

# Thư mục với hình ảnh tương tư kiểm thử
test_lovesick_dir = os.path.join(test_dir, 'TuongTu')
check_folder_exist(test_lovesick_dir)
print("Tạo thành công!")
```

- Xử lý hình ảnh: Thay đổi các hình ảnh từ RGBA và P thành JPEG.

```
def convert_images_to_jpg(input_folder, output_folder):
    # Lặp qua tất cả tệp trong thư mục đầu vào
    for filename in os.listdir(input_folder):
        input_path = os.path.join(input_folder, filename)

        # Kiểm tra nếu tệp là hình ảnh
        if os.path.isfile(input_path) or any(input_path.lower().endswith(ext) for ext in ('.jpg', '.jpeg', '.png', '.gif', '.bmp', '.webp')):
            # Đọc và chuyển đổi hình ảnh thành định dạng JPG
            try:
                image = Image.open(input_path)
                if image.mode == 'RGBA' or image.mode == 'P':
                    image = image.convert('RGB')
                output_path = os.path.join(output_folder, os.path.splitext(filename)[0] + '.jpg')
                image.save(output_path, 'JPEG')
            except Exception as e:
                os.remove(input_path)
                print(f"Lỗi khi chuyển đổi {input_path}: {str(e)}")

# Định nghĩa các thư mục nguồn cho mỗi loại hình ảnh
source_loving_dir = os.path.join(original_dataset_dir, 'DangYeu')
source_lovelorn_dir = os.path.join(original_dataset_dir, 'ThatTinh')
source_lovesick_dir = os.path.join(original_dataset_dir, 'TuongTu')

convert_images_to_jpg(source_loving_dir, processed_loving_dir)
convert_images_to_jpg(source_lovelorn_dir, processed_lovelorn_dir)
convert_images_to_jpg(source_lovesick_dir, processed_lovesick_dir)
print("Convert done!")
```

- Chia tập dữ liệu: Dữ liệu được chia theo tỉ lệ train 70%, val 20%, test 10%

```
# Lấy danh sách các tệp hình ảnh trong từng thư mục đã qua xử lý
loving_images = os.listdir(processed_loving_dir)
lovelorn_images = os.listdir(processed_lovelorn_dir)
lovesick_images = os.listdir(processed_lovesick_dir)

# Tính số lượng hình ảnh cho từng tập dữ liệu
loving_images_count = len(loving_images)
lovelorn_images_count = len(lovelorn_images)
lovesick_images_count = len(lovesick_images)

train_loving_count = int(loving_images_count * 0.7)
validation_loving_count = int(loving_images_count * 0.2)
test_loving_count = loving_images_count - train_loving_count - validation_loving_count

train_lovelorn_count = int(lovelorn_images_count * 0.7)
validation_lovelorn_count = int(lovelorn_images_count * 0.2)
test_lovelorn_count = lovelorn_images_count - train_lovelorn_count - validation_lovelorn_count

train_lovesick_count = int(lovesick_images_count * 0.7)
validation_lovesick_count = int(lovesick_images_count * 0.2)
test_lovesick_count = lovesick_images_count - train_lovesick_count - validation_lovesick_count
```

```
# Di chuyển hình ảnh vào các thư mục tương ứng
for image in loving_images[:train_loving_count]:
    source = os.path.join(processed_loving_dir, image)
    destination = os.path.join(train_loving_dir, image)
    shutil.copy(source, destination)

for image in loving_images[train_loving_count:train_loving_count + validation_loving_count]:
    source = os.path.join(processed_loving_dir, image)
    destination = os.path.join(validation_loving_dir, image)
    shutil.copy(source, destination)

for image in loving_images[train_loving_count + validation_loving_count:train_loving_count + validation_loving_count + test_loving_count]:
    source = os.path.join(processed_loving_dir, image)
    destination = os.path.join(test_loving_dir, image)
    shutil.copy(source, destination)
```

```
for image in lovelorn_images[:train_lovelorn_count]:
    source = os.path.join(processed_lovelorn_dir, image)
    destination = os.path.join(train_lovelorn_dir, image)
    shutil.copy(source, destination)

for image in lovelorn_images[train_lovelorn_count:train_lovelorn_count + validation_lovelorn_count]:
    source = os.path.join(processed_lovelorn_dir, image)
    destination = os.path.join(validation_lovelorn_dir, image)
    shutil.copy(source, destination)

for image in lovelorn_images[train_lovelorn_count + validation_lovelorn_count:train_lovelorn_count + validation_lovelorn_count + test_lovelorn_count]:
    source = os.path.join(processed_lovelorn_dir, image)
    destination = os.path.join(test_lovelorn_dir, image)
    shutil.copy(source, destination)
```

```
for image in lovesick_images[:train_lovesick_count]:
    source = os.path.join(processed_lovesick_dir, image)
    destination = os.path.join(train_lovesick_dir, image)
    shutil.copy(source, destination)

for image in lovesick_images[train_lovesick_count:train_lovesick_count + validation_lovesick_count]:
    source = os.path.join(processed_lovesick_dir, image)
    destination = os.path.join(validation_lovesick_dir, image)
    shutil.copy(source, destination)

for image in lovesick_images[train_lovesick_count + validation_lovesick_count:train_lovesick_count + validation_lovesick_count + test_lovesick_count]:
    source = os.path.join(processed_lovesick_dir, image)
    destination = os.path.join(test_lovesick_dir, image)
    shutil.copy(source, destination)

print("Chia tập dữ liệu thành công!")
```

- Số lượng của tập dữ liệu:

```
print('total training loving images:', len(os.listdir(train_loving_dir)))
print('total validation loving images:', len(os.listdir(validation_loving_dir)))
print('total test loving images:', len(os.listdir(test_loving_dir)))

print('total training lovelorn images:', len(os.listdir(train_lovelorn_dir)))
print('total validation lovelorn images:', len(os.listdir(validation_lovelorn_dir)))
print('total test lovelorn images:', len(os.listdir(test_lovelorn_dir)))

print('total training lovesick images:', len(os.listdir(train_lovesick_dir)))
print('total validation lovesick images:', len(os.listdir(validation_lovesick_dir)))
print('total test lovesick images:', len(os.listdir(test_lovesick_dir)))
```

```
total training loving images: 234
total validation loving images: 67
total test loving images: 34
total training lovelorn images: 186
total validation lovelorn images: 53
total test lovelorn images: 28
total training lovesick images: 152
total validation lovesick images: 43
total test lovesick images: 23
```

- Load dữ liệu:

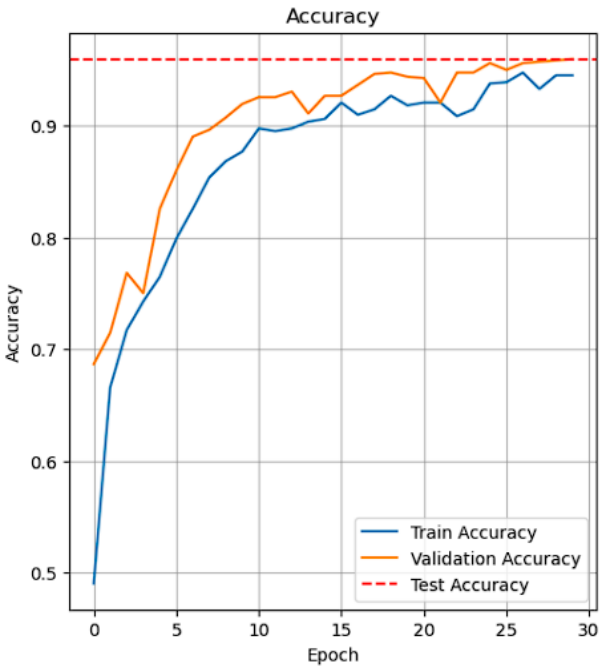
```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dropout, Dense
import matplotlib.pyplot as plt

# Load dữ liệu training và validation
train_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    base_dir,
    image_size=(150, 150),
    batch_size=64
)
validation_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    base_dir,
    image_size=(150, 150),
    batch_size=64
)

# Đánh giá mô hình trên tập dữ liệu test
test_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    base_dir,
    image_size=(150, 150),
    batch_size=64
)
```

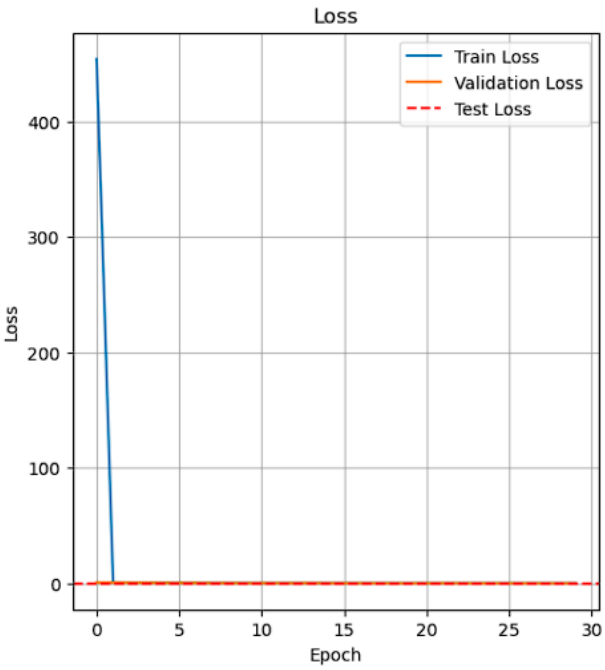
- Mô hình 1: Dropout 20%

```python
# Tạo mô hình
model_1 = Sequential()
model_1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model_1.add(MaxPooling2D((2, 2)))
model_1.add(Conv2D(64, (3, 3), activation='relu'))
model_1.add(MaxPooling2D((2, 2)))
model_1.add(Conv2D(128, (3, 3), activation='relu'))
model_1.add(MaxPooling2D((2, 2)))
model_1.add(Flatten())
model_1.add(Dropout(0.2))
model_1.add(Dense(512, activation='relu'))
model_1.add(Dense(3, activation='softmax'))
```

```python
# Compile mô hình
model_1.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Đào tạo mô hình
history = model_1.fit(train_dataset, epochs=30, validation_data=validation_dataset)

# Lấy thông tin accuracy và loss của tập kiểm tra
test_loss, test_accuracy = model_1.evaluate(test_dataset)

# Tạo biểu đồ so sánh
plt.figure(figsize=(12, 6))

# Biểu đồ Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.axhline(y=test_accuracy, color='r', linestyle='--', label='Test Accuracy')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)

# Biểu đồ Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.axhline(y=test_loss, color='r', linestyle='--', label='Test Loss')
plt.title('Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)

plt.show()

print('Test loss:', test_loss)
print('Test accuracy:', test_accuracy)
```

```
Found 820 files belonging to 3 classes.
Found 820 files belonging to 3 classes.
Found 820 files belonging to 3 classes.
Epoch 1/30
13/13 [==============================] - 17s 1s/step - loss: 454.2341 - accuracy: 0.4902 - val_loss: 0.8988 - val_accuracy: 0.6866
Epoch 2/30
13/13 [==============================] - 15s 1s/step - loss: 0.9913 - accuracy: 0.6659 - val_loss: 0.8629 - val_accuracy: 0.7146
Epoch 3/30
13/13 [==============================] - 14s 1s/step - loss: 0.8389 - accuracy: 0.7171 - val_loss: 0.6475 - val_accuracy: 0.7683
Epoch 4/30
13/13 [==============================] - 14s 1s/step - loss: 0.7278 - accuracy: 0.7427 - val_loss: 0.6538 - val_accuracy: 0.7500
Epoch 5/30
13/13 [==============================] - 14s 1s/step - loss: 0.6217 - accuracy: 0.7646 - val_loss: 0.4667 - val_accuracy: 0.8256
Epoch 6/30
13/13 [==============================] - 14s 1s/step - loss: 0.5288 - accuracy: 0.7988 - val_loss: 0.4277 - val_accuracy: 0.8598
Epoch 7/30
13/13 [==============================] - 14s 1s/step - loss: 0.4862 - accuracy: 0.8256 - val_loss: 0.3796 - val_accuracy: 0.8902
Epoch 8/30
13/13 [==============================] - 14s 1s/step - loss: 0.3861 - accuracy: 0.8537 - val_loss: 0.3117 - val_accuracy: 0.8963
Epoch 9/30
13/13 [==============================] - 14s 1s/step - loss: 0.3657 - accuracy: 0.8683 - val_loss: 0.3034 - val_accuracy: 0.9073
Epoch 10/30
13/13 [==============================] - 14s 1s/step - loss: 0.3509 - accuracy: 0.8768 - val_loss: 0.2342 - val_accuracy: 0.9195
Epoch 11/30
13/13 [==============================] - 14s 1s/step - loss: 0.2825 - accuracy: 0.8976 - val_loss: 0.2558 - val_accuracy: 0.9256
Epoch 12/30
13/13 [==============================] - 14s 1s/step - loss: 0.2905 - accuracy: 0.8951 - val_loss: 0.2306 - val_accuracy: 0.9256
Epoch 13/30
13/13 [==============================] - 14s 1s/step - loss: 0.2920 - accuracy: 0.8976 - val_loss: 0.2346 - val_accuracy: 0.9305
Epoch 14/30
13/13 [==============================] - 14s 1s/step - loss: 0.2764 - accuracy: 0.9037 - val_loss: 0.3078 - val_accuracy: 0.9110
Epoch 15/30
13/13 [==============================] - 14s 1s/step - loss: 0.2891 - accuracy: 0.9061 - val_loss: 0.1945 - val_accuracy: 0.9268
Epoch 16/30
13/13 [==============================] - 14s 1s/step - loss: 0.2375 - accuracy: 0.9207 - val_loss: 0.1693 - val_accuracy: 0.9268
Epoch 17/30
13/13 [==============================] - 14s 1s/step - loss: 0.2279 - accuracy: 0.9098 - val_loss: 0.1671 - val_accuracy: 0.9366
Epoch 18/30
13/13 [==============================] - 14s 1s/step - loss: 0.2328 - accuracy: 0.9146 - val_loss: 0.1534 - val_accuracy: 0.9463
Epoch 19/30
13/13 [==============================] - 14s 1s/step - loss: 0.2066 - accuracy: 0.9268 - val_loss: 0.1418 - val_accuracy: 0.9476
Epoch 20/30
13/13 [==============================] - 14s 1s/step - loss: 0.2155 - accuracy: 0.9183 - val_loss: 0.1309 - val_accuracy: 0.9439
Epoch 21/30
13/13 [==============================] - 14s 1s/step - loss: 0.2020 - accuracy: 0.9207 - val_loss: 0.1496 - val_accuracy: 0.9427
Epoch 22/30
13/13 [==============================] - 14s 1s/step - loss: 0.2104 - accuracy: 0.9207 - val_loss: 0.2103 - val_accuracy: 0.9207
Epoch 23/30
13/13 [==============================] - 15s 1s/step - loss: 0.2243 - accuracy: 0.9085 - val_loss: 0.1333 - val_accuracy: 0.9476
Epoch 24/30
13/13 [==============================] - 15s 1s/step - loss: 0.1754 - accuracy: 0.9146 - val_loss: 0.1245 - val_accuracy: 0.9476
Epoch 25/30
13/13 [==============================] - 15s 1s/step - loss: 0.1676 - accuracy: 0.9378 - val_loss: 0.1133 - val_accuracy: 0.9561
Epoch 26/30
13/13 [==============================] - 15s 1s/step - loss: 0.1846 - accuracy: 0.9390 - val_loss: 0.1386 - val_accuracy: 0.9500
Epoch 27/30
13/13 [==============================] - 15s 1s/step - loss: 0.1553 - accuracy: 0.9476 - val_loss: 0.1120 - val_accuracy: 0.9561
Epoch 28/30
13/13 [==============================] - 15s 1s/step - loss: 0.1633 - accuracy: 0.9329 - val_loss: 0.1030 - val_accuracy: 0.9573
Epoch 29/30
13/13 [==============================] - 15s 1s/step - loss: 0.1501 - accuracy: 0.9451 - val_loss: 0.1015 - val_accuracy: 0.9585
Epoch 30/30
13/13 [==============================] - 15s 1s/step - loss: 0.1595 - accuracy: 0.9451 - val_loss: 0.0972 - val_accuracy: 0.9598
13/13 [==============================] - 2s 147ms/step - loss: 0.0972 - accuracy: 0.9598
```





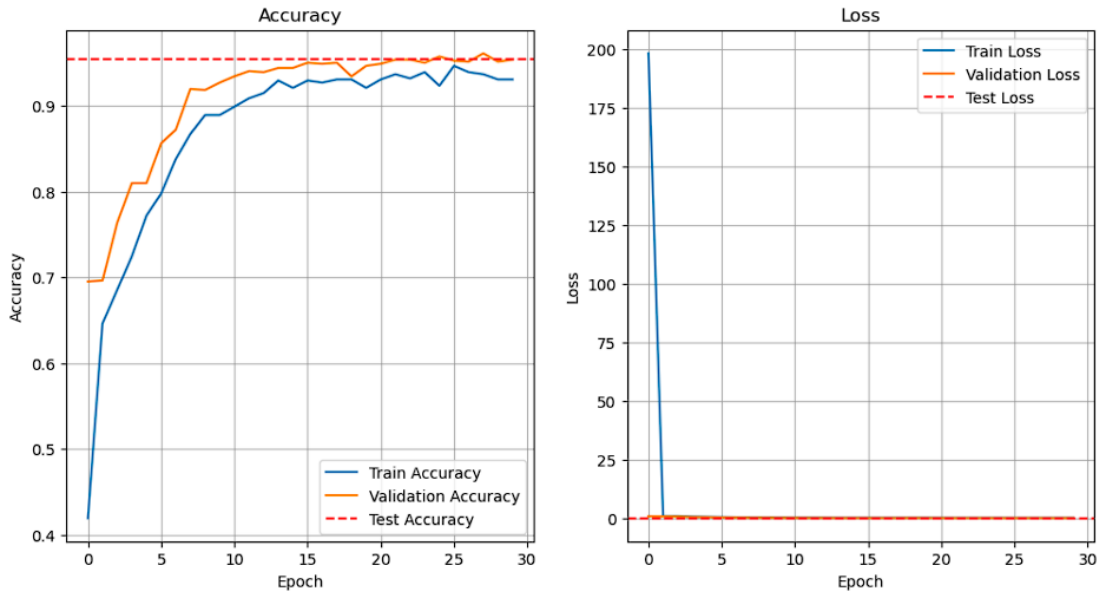

```
Test loss: 0.0972299799323082
Test accuracy: 0.959756076335907
```

- Mô hình 2: Dropout 20%

```
# Mô hình 2
model_2 = Sequential()
model_2.add(Conv2D(64, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model_2.add(MaxPooling2D((2, 2)))
model_2.add(Conv2D(128, (3, 3), activation='relu'))
model_2.add(MaxPooling2D((2, 2)))
model_2.add(Conv2D(256, (3, 3), activation='relu'))
model_2.add(MaxPooling2D((2, 2)))
model_2.add(Flatten())
model_2.add(Dropout(0.2))
model_2.add(Dense(512, activation='relu'))
model_2.add(Dense(3, activation='softmax'))
```

```
# Compile mô hình
model_2.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Đào tạo mô hình
history = model_2.fit(train_dataset, epochs=30, validation_data=validation_dataset)

# Lấy thông tin accuracy và loss của tập kiểm tra
test_loss, test_accuracy = model_2.evaluate(test_dataset)

# Tạo biểu đồ so sánh
plt.figure(figsize=(12, 6))

# Biểu đồ Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.axhline(y=test_accuracy, color='r', linestyle='--', label='Test Accuracy')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)

# Biểu đồ Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.axhline(y=test_loss, color='r', linestyle='--', label='Test Loss')
plt.title('Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)

plt.show()

print('Test loss:', test_loss)
print('Test accuracy:', test_accuracy)
```

```
Epoch 1/30
13/13 [==============================] - 34s 3s/step - loss: 198.3458 - accuracy: 0.4195 - val_loss: 0.9127 - val_accuracy: 0.6951
Epoch 2/30
13/13 [==============================] - 31s 2s/step - loss: 0.9735 - accuracy: 0.6463 - val_loss: 0.8397 - val_accuracy: 0.6963
Epoch 3/30
13/13 [==============================] - 30s 2s/step - loss: 0.9630 - accuracy: 0.6854 - val_loss: 0.7663 - val_accuracy: 0.7634
Epoch 4/30
13/13 [==============================] - 30s 2s/step - loss: 0.7302 - accuracy: 0.7244 - val_loss: 0.5743 - val_accuracy: 0.8098
Epoch 5/30
13/13 [==============================] - 30s 2s/step - loss: 0.6207 - accuracy: 0.7720 - val_loss: 0.5171 - val_accuracy: 0.8098
Epoch 6/30
13/13 [==============================] - 30s 2s/step - loss: 0.5388 - accuracy: 0.7976 - val_loss: 0.3922 - val_accuracy: 0.8561
Epoch 7/30
13/13 [==============================] - 31s 2s/step - loss: 0.4317 - accuracy: 0.8378 - val_loss: 0.3383 - val_accuracy: 0.8720
Epoch 8/30
13/13 [==============================] - 33s 3s/step - loss: 0.3741 - accuracy: 0.8671 - val_loss: 0.2528 - val_accuracy: 0.9195
Epoch 9/30
13/13 [==============================] - 33s 3s/step - loss: 0.3333 - accuracy: 0.8890 - val_loss: 0.2470 - val_accuracy: 0.9183
Epoch 10/30
13/13 [==============================] - 34s 3s/step - loss: 0.3139 - accuracy: 0.8890 - val_loss: 0.2076 - val_accuracy: 0.9268
Epoch 11/30
13/13 [==============================] - 33s 3s/step - loss: 0.2574 - accuracy: 0.8988 - val_loss: 0.1975 - val_accuracy: 0.9341
Epoch 12/30
13/13 [==============================] - 32s 2s/step - loss: 0.2697 - accuracy: 0.9085 - val_loss: 0.1621 - val_accuracy: 0.9402
Epoch 13/30
13/13 [==============================] - 31s 2s/step - loss: 0.2386 - accuracy: 0.9146 - val_loss: 0.1761 - val_accuracy: 0.9390
Epoch 14/30
13/13 [==============================] - 30s 2s/step - loss: 0.1907 - accuracy: 0.9293 - val_loss: 0.1381 - val_accuracy: 0.9439
Epoch 15/30
13/13 [==============================] - 33s 3s/step - loss: 0.2064 - accuracy: 0.9207 - val_loss: 0.1394 - val_accuracy: 0.9439
Epoch 16/30
13/13 [==============================] - 34s 3s/step - loss: 0.2019 - accuracy: 0.9293 - val_loss: 0.1299 - val_accuracy: 0.9500
Epoch 17/30
13/13 [==============================] - 34s 3s/step - loss: 0.1886 - accuracy: 0.9268 - val_loss: 0.1336 - val_accuracy: 0.9488
Epoch 18/30
13/13 [==============================] - 32s 2s/step - loss: 0.2082 - accuracy: 0.9305 - val_loss: 0.1393 - val_accuracy: 0.9500
Epoch 19/30
13/13 [==============================] - 32s 2s/step - loss: 0.2119 - accuracy: 0.9305 - val_loss: 0.1716 - val_accuracy: 0.9341
Epoch 20/30
13/13 [==============================] - 31s 2s/step - loss: 0.2267 - accuracy: 0.9207 - val_loss: 0.1372 - val_accuracy: 0.9463
Epoch 21/30
13/13 [==============================] - 32s 2s/step - loss: 0.1788 - accuracy: 0.9305 - val_loss: 0.1268 - val_accuracy: 0.9488
Epoch 22/30
13/13 [==============================] - 32s 2s/step - loss: 0.1640 - accuracy: 0.9366 - val_loss: 0.1179 - val_accuracy: 0.9537
Epoch 23/30
13/13 [==============================] - 32s 2s/step - loss: 0.1702 - accuracy: 0.9317 - val_loss: 0.1168 - val_accuracy: 0.9537
Epoch 24/30
13/13 [==============================] - 36s 3s/step - loss: 0.1512 - accuracy: 0.9390 - val_loss: 0.1309 - val_accuracy: 0.9500
Epoch 25/30
13/13 [==============================] - 33s 3s/step - loss: 0.1833 - accuracy: 0.9232 - val_loss: 0.1074 - val_accuracy: 0.9573
Epoch 26/30
13/13 [==============================] - 32s 2s/step - loss: 0.1380 - accuracy: 0.9463 - val_loss: 0.1200 - val_accuracy: 0.9524
Epoch 27/30
13/13 [==============================] - 32s 2s/step - loss: 0.1509 - accuracy: 0.9390 - val_loss: 0.1220 - val_accuracy: 0.9512
Epoch 28/30
13/13 [==============================] - 30s 2s/step - loss: 0.1529 - accuracy: 0.9366 - val_loss: 0.1108 - val_accuracy: 0.9610
Epoch 29/30
13/13 [==============================] - 32s 2s/step - loss: 0.1649 - accuracy: 0.9305 - val_loss: 0.1103 - val_accuracy: 0.9512
Epoch 30/30
13/13 [==============================] - 31s 2s/step - loss: 0.1725 - accuracy: 0.9305 - val_loss: 0.1351 - val_accuracy: 0.9537
13/13 [==============================] - 5s 373ms/step - loss: 0.1351 - accuracy: 0.9537
```



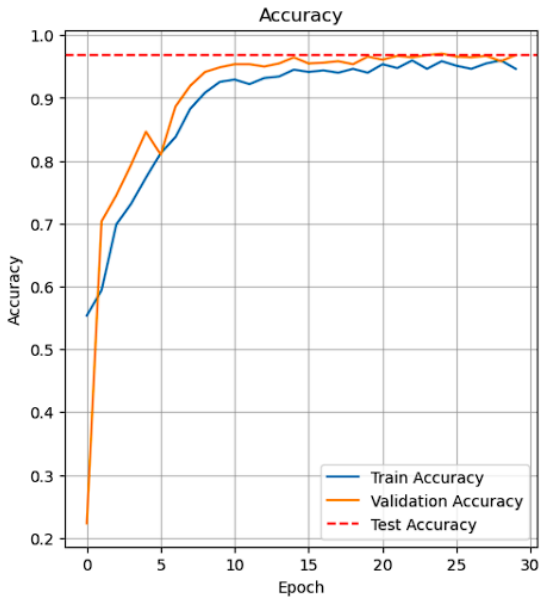

```
Test loss: 0.13509592413902283
Test accuracy: 0.9536585211753845
```

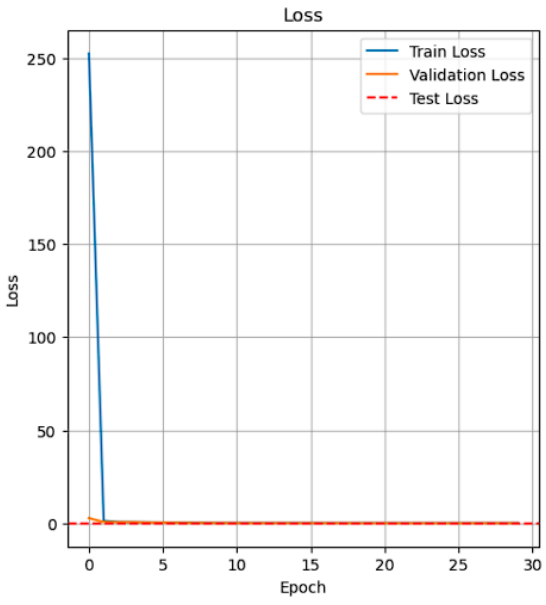
- Mô hình 3:

```
# Mô hình 3
model_3 = Sequential()
model_3.add(Conv2D(16, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model_3.add(MaxPooling2D((2, 2)))
model_3.add(Conv2D(32, (3, 3), activation='relu'))
model_3.add(MaxPooling2D((2, 2)))
model_3.add(Conv2D(64, (3, 3), activation='relu'))
model_3.add(MaxPooling2D((2, 2)))
model_3.add(Flatten())
model_3.add(Dropout(0.2))
model_3.add(Dense(512, activation='relu'))
model_3.add(Dense(3, activation='softmax'))
```

```
# Compile mô hình
model_3.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Đào tạo mô hình
history = model_3.fit(train_dataset, epochs=30, validation_data=validation_dataset)

# Lấy thông tin accuracy và loss của tập kiểm tra
test_loss, test_accuracy = model_3.evaluate(test_dataset)

# Tạo biểu đồ so sánh
plt.figure(figsize=(12, 6))

# Biểu đồ Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.axhline(y=test_accuracy, color='r', linestyle='--', label='Test Accuracy')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)

# Biểu đồ Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.axhline(y=test_loss, color='r', linestyle='--', label='Test Loss')
plt.title('Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)

plt.show()

print('Test loss:', test_loss)
print('Test accuracy:', test_accuracy)
```

```
Epoch 1/30
13/13 [==============================] - 9s 589ms/step - loss: 252.3664 - accuracy: 0.5537 - val_loss: 2.8893 - val_accuracy: 0.2232
Epoch 2/30
13/13 [==============================] - 8s 570ms/step - loss: 1.3877 - accuracy: 0.5939 - val_loss: 0.7890 - val_accuracy: 0.7037
Epoch 3/30
13/13 [==============================] - 8s 561ms/step - loss: 0.7763 - accuracy: 0.6988 - val_loss: 0.6849 - val_accuracy: 0.7451
Epoch 4/30
13/13 [==============================] - 7s 541ms/step - loss: 0.6813 - accuracy: 0.7317 - val_loss: 0.7031 - val_accuracy: 0.7939
Epoch 5/30
13/13 [==============================] - 8s 594ms/step - loss: 0.6112 - accuracy: 0.7732 - val_loss: 0.4787 - val_accuracy: 0.8463
Epoch 6/30
13/13 [==============================] - 8s 567ms/step - loss: 0.5012 - accuracy: 0.8122 - val_loss: 0.4090 - val_accuracy: 0.8898
Epoch 7/30
13/13 [==============================] - 8s 569ms/step - loss: 0.4142 - accuracy: 0.8378 - val_loss: 0.3054 - val_accuracy: 0.8866
Epoch 8/30
13/13 [==============================] - 8s 620ms/step - loss: 0.3425 - accuracy: 0.8829 - val_loss: 0.2365 - val_accuracy: 0.9195
Epoch 9/30
13/13 [==============================] - 8s 621ms/step - loss: 0.2608 - accuracy: 0.9085 - val_loss: 0.1988 - val_accuracy: 0.9415
Epoch 10/30
13/13 [==============================] - 8s 572ms/step - loss: 0.2245 - accuracy: 0.9256 - val_loss: 0.1581 - val_accuracy: 0.9488
Epoch 11/30
13/13 [==============================] - 8s 578ms/step - loss: 0.2226 - accuracy: 0.9293 - val_loss: 0.1518 - val_accuracy: 0.9537
Epoch 12/30
13/13 [==============================] - 8s 579ms/step - loss: 0.2117 - accuracy: 0.9220 - val_loss: 0.1369 - val_accuracy: 0.9537
Epoch 13/30
13/13 [==============================] - 8s 562ms/step - loss: 0.1955 - accuracy: 0.9317 - val_loss: 0.1318 - val_accuracy: 0.9500
Epoch 14/30
13/13 [==============================] - 7s 548ms/step - loss: 0.1869 - accuracy: 0.9341 - val_loss: 0.1352 - val_accuracy: 0.9549
Epoch 15/30
13/13 [==============================] - 8s 589ms/step - loss: 0.2013 - accuracy: 0.9451 - val_loss: 0.1081 - val_accuracy: 0.9646
Epoch 16/30
13/13 [==============================] - 8s 614ms/step - loss: 0.1729 - accuracy: 0.9415 - val_loss: 0.1214 - val_accuracy: 0.9549
Epoch 17/30
13/13 [==============================] - 7s 546ms/step - loss: 0.1601 - accuracy: 0.9439 - val_loss: 0.1117 - val_accuracy: 0.9561
Epoch 18/30
13/13 [==============================] - 7s 541ms/step - loss: 0.1663 - accuracy: 0.9402 - val_loss: 0.1123 - val_accuracy: 0.9585
Epoch 19/30
13/13 [==============================] - 7s 547ms/step - loss: 0.1551 - accuracy: 0.9463 - val_loss: 0.1113 - val_accuracy: 0.9537
Epoch 20/30
13/13 [==============================] - 7s 537ms/step - loss: 0.1619 - accuracy: 0.9402 - val_loss: 0.0973 - val_accuracy: 0.9659
Epoch 21/30
13/13 [==============================] - 7s 541ms/step - loss: 0.1470 - accuracy: 0.9537 - val_loss: 0.1009 - val_accuracy: 0.9610
Epoch 22/30
13/13 [==============================] - 7s 543ms/step - loss: 0.1441 - accuracy: 0.9476 - val_loss: 0.0915 - val_accuracy: 0.9671
Epoch 23/30
13/13 [==============================] - 7s 546ms/step - loss: 0.1258 - accuracy: 0.9598 - val_loss: 0.0903 - val_accuracy: 0.9646
Epoch 24/30
13/13 [==============================] - 8s 553ms/step - loss: 0.1405 - accuracy: 0.9463 - val_loss: 0.0834 - val_accuracy: 0.9683
Epoch 25/30
13/13 [==============================] - 7s 538ms/step - loss: 0.1203 - accuracy: 0.9585 - val_loss: 0.0790 - val_accuracy: 0.9707
Epoch 26/30
13/13 [==============================] - 7s 538ms/step - loss: 0.1294 - accuracy: 0.9512 - val_loss: 0.0844 - val_accuracy: 0.9659
Epoch 27/30
13/13 [==============================] - 7s 532ms/step - loss: 0.1313 - accuracy: 0.9463 - val_loss: 0.0945 - val_accuracy: 0.9646
Epoch 28/30
13/13 [==============================] - 8s 553ms/step - loss: 0.1413 - accuracy: 0.9549 - val_loss: 0.0901 - val_accuracy: 0.9671
Epoch 29/30
13/13 [==============================] - 7s 537ms/step - loss: 0.1165 - accuracy: 0.9598 - val_loss: 0.1141 - val_accuracy: 0.9585
Epoch 30/30
13/13 [==============================] - 8s 573ms/step - loss: 0.1642 - accuracy: 0.9463 - val_loss: 0.0968 - val_accuracy: 0.9683
13/13 [==============================] - 2s 89ms/step - loss: 0.0968 - accuracy: 0.9683
```






```
Test loss: 0.09684915095567703
Test accuracy: 0.9682926535606384
```

- Đánh giá:

Cả 3 mô hình đều có xu hướng chung, đó là:

- Accuracy của tập train có xu hướng tăng qua từng epoch và vượt qua accuracy của tập test ở các epoch sau. Điều này cho thấy rằng mô hình đã học được các mẫu dữ liệu trong tập train rất tốt, nhưng nó chưa thể áp dụng các mẫu đó để dự đoán chính xác các điểm dữ liệu trong tập test. Nguyên nhân có thể là do tập test có dữ liệu khác biệt so với tập train, hoặc mô hình đã bị overfit.
- Sự tăng cao đột ngột của loss ở epoch đầu tiên xảy ra vì mô hình đang bắt đầu với trọng số ngẫu nhiên. Loss của tập train giảm dần theo số epoch. Điều này cho thấy rằng mô hình đang dần cải thiện khả năng dự đoán của mình.