Deep Learning trong khoa học dữ liệu
Bài tập thực hành 5 - DS201.M11.2

Đỗ Trọng Hợp, Lưu Thanh Sơn, Nguyễn Thành Luân
Sinh viên: Phạm Đức Thể - 19522253
Ngôn ngữ lập trình: Python

Thứ 4, ngày 24 tháng 11 năm 2021

Bài tập: ỨNG DỤNG MẠNG NEURAL TÍCH CHẬP

Set up

1. Import các thư viện cần thiết

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
import seaborn as sn

from tensorflow.keras.utils import to_categorical
from keras.models import Model, Input
from tensorflow.keras.optimizers import Adam
from keras.losses import BinaryCrossentropy, CategoricalCrossentropy
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score
from keras.regularizers import l2
from keras.preprocessing.image import image_dataset_from_directory
from keras.applications.vgg16 import VGG16
from keras.applications.vgg19 import VGG19
from keras.applications.resnet import ResNet50
from keras.layers import Flatten, Dense, Dropout
from keras.utils.vis_utils import plot_model

sn.set() # Set theme
```

Bài 1: Thực hiện huấn luyện mô hình mạng neural dùng cho phân lớp nhị phân cho bộ dữ liệu Chest X-ray bằng kỹ thuật transfer learning. Tinh chỉnh lại các mô hình sau: VGG-16, VGG-19, ResNet-50. Đánh giá mô hình bằng các độ đo: accuracy, precision, recall và F1-score. Có kết luận gì về khả năng phân lớp của các mô hình?

1. Chuẩn bị dữ liệu

```
IMG_SIZE = 227
IMG_CHANNEL = 3
BATCH_SIZE = 256
COLOR_MODE = 'rgb'

# Load train set
train_set_bai1=image_dataset_from_directory('/chest_xray/chest_xray/train',
                                            labels = 'inferred',
                                            label_mode = 'binary',
                                            class_names = ['NORMAL', 'PNEUMONIA'],
                                            color_mode = COLOR_MODE,
                                            batch_size = BATCH_SIZE,
```

```
                                                    image_size = (IMG_SIZE, IMG_SIZE),
                                                    interpolation = 'bilinear')
# Output: Found 5216 files belonging to 2 classes.
# Load dev set
dev_set_bai1 = image_dataset_from_directory('/chest_xray/chest_xray/val',
                                            labels = 'inferred',
                                            label_mode = 'binary',
                                            class_names = ['NORMAL', 'PNEUMONIA'],
                                            color_mode = COLOR_MODE,
                                            batch_size = BATCH_SIZE,
                                            image_size = (IMG_SIZE, IMG_SIZE),
                                            interpolation = 'bilinear')
# Output: Found 16 files belonging to 2 classes.
# Load test set
test_set_bai1 = image_dataset_from_directory('/chest_xray/chest_xray/test',
                                            labels = 'inferred',
                                            label_mode = 'binary',
                                            class_names = ['NORMAL', 'PNEUMONIA'],
                                            color_mode = COLOR_MODE,
                                            batch_size = BATCH_SIZE,
                                            image_size = (IMG_SIZE, IMG_SIZE),
                                            interpolation = 'bilinear')
# Output: Found 624 files belonging to 2 classes.
```

Sử dụng VGG-16 pre-trained

1. Xây dựng mô hình

```
vgg = VGG16(include_top=False, input_shape = (IMG_SIZE, IMG_SIZE, IMG_CHANNEL))

for layer in vgg.layers:
  layer.trainable = False

flat = Flatten()(vgg.layers[-1].output)
fc1 = Dense(1024, activation='relu')(flat)
output = Dense(1, activation='sigmoid')(fc1)

model_1 = Model(inputs=vgg.inputs, outputs=output)

model_1.summary()
```

2. Huấn luyện mô hình

```
# Compile model
Optimizer = Adam(learning_rate=1e-3)
Loss = BinaryCrossentropy()
model_1.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_1 = model_1.fit(train_set_bai1,
                        validation_data=dev_set_bai1, epochs=10)
```

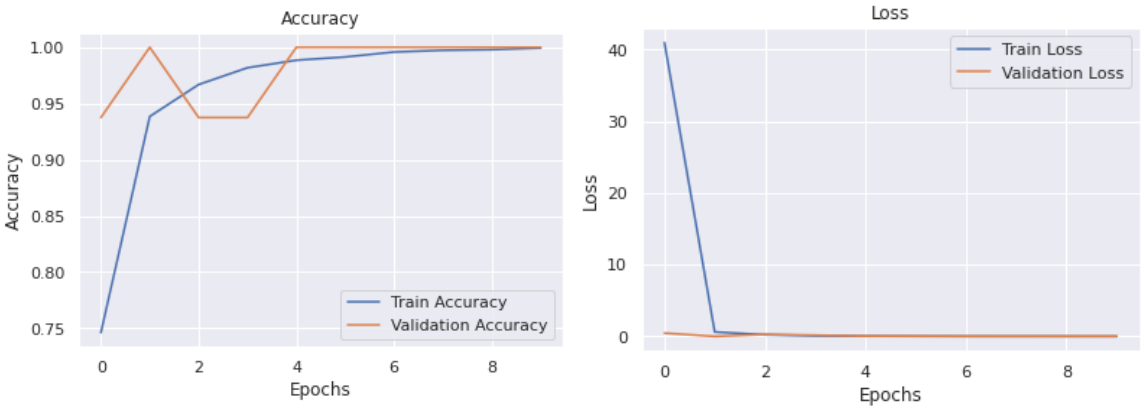
3. Visualization Loss & Accuracy training

```
plt.plot(history_1.history['accuracy'], label = 'Train Accuracy')
plt.plot(history_1.history['val_accuracy'], label = 'Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

– Output: Hình 1a

```
plt.plot(history_1.history['loss'], label = 'Train Loss')
plt.plot(history_1.history['val_loss'], label = 'Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 1b



(a) Accuracy

(b) Loss

Hình 1: Accuracy và Loss của model VGG-16 Pre-trained bài 1

* **Nhận xét**:

– Accuracy trên tập train tăng nhanh và hội tụ sớm. Accuracy trên tập dev có sự dao động mạnh ở những epcohs đầu nhưng cũng hội tụ kể từ epoch thứ 4. Accuracy trên cả 2 tập đều rất cao (Accuracy train $\sim$ 99.9%, Accuracy dev = 100%).

– Loss trên cả 2 tập train và dev đều giảm rất nhanh và hội tụ rất sớm (ngay ở epoch đầu tiên).

4. Đánh giá mô hình

```
y_pred_total = []
y_true = []
for img, label in test_set_bai1:
  y_pred = model_1.predict(img)
  y_pred_total += [1 if i > 0.5 else 0 for i in y_pred]
  y_true += np.array(label).flatten().tolist()
accuracy = round(accuracy_score(y_true, y_pred_total)*100,2)
print('Accuracy test = {}%'.format(accuracy))
# Output: Accuracy test = 78.53%
precision = round(precision_score(y_true, y_pred_total)*100,2)
print('Precision test = {}%'.format(precision))
# Output: Precision test = 74.43%
recall = round(recall_score(y_true, y_pred_total)*100,2)
print('Recall test = {}%'.format(recall))
# Output: Recall test = 100.0%
f1 = round(f1_score(y_true, y_pred_total)*100,2)
print('F1-score test = {}%'.format(f1))
# Output: F1-score test = 85.34%
```

* **Nhận xét**:

– Accuracy của mô hình khi kiểm tra trên tập test khá cao (Accuracy test = 78.53%). Có thể thấy đây là một mô hình khá tốt. Ngoài ra, với các độ đo khác model cũng cho kết quả khá cao (Precision = 74.43%, Recall = 100.0%, F1-score = 85.34%).

– Có thể kết luận mô hình có khả năng phân lớp khá tốt.

Sử dụng VGG-19 pre-trained

1. Xây dựng mô hình

```
vgg = VGG19(include_top=False, input_shape = (IMG_SIZE, IMG_SIZE, IMG_CHANNEL))
for layer in vgg.layers:
  layer.trainable = False

flat = Flatten()(vgg.layers[-1].output)
fc1 = Dense(1024, activation='relu')(flat)
output = Dense(1, activation='sigmoid')(fc1)
model_2 = Model(inputs=vgg.inputs, outputs=output)
model_2.summary()
```

2. Huấn luyện mô hình

```
# Compile model
Optimizer = Adam(learning_rate=1e-3)
Loss = BinaryCrossentropy()
model_2.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])
# Training model
history_2 = model_2.fit(train_set_bai1,
                        validation_data=dev_set_bai1, epochs=10)
```

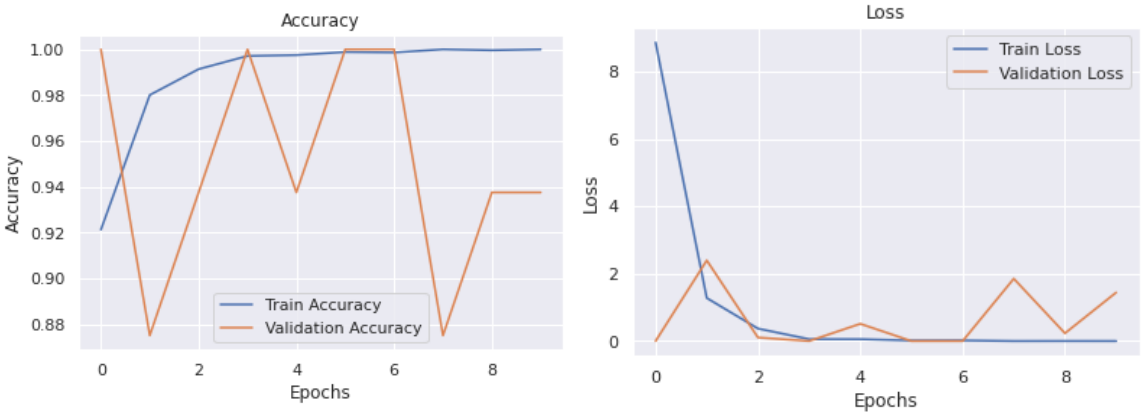
3. Visualization Loss & Accuracy training

```
plt.plot(history_2.history['accuracy'], label = 'Train Accuracy')
plt.plot(history_2.history['val_accuracy'], label = 'Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

– Output: Hình 2a

```
plt.plot(history_2.history['loss'], label = 'Train Loss')
plt.plot(history_2.history['val_loss'], label = 'Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 2b



(a) Accuracy

(b) Loss

Hình 2: Accuracy và Loss của model VGG-19 Pre-trained bài 1

* **Nhận xét**:
  - Accuracy trên tập train tăng nhanh và hội tụ sớm. Accuracy trên tập dev có sự dao động mạnh và chưa có xu hướng hội tụ. Accuracy trên cả 2 tập đều rất cao (Accuracy train $\sim 100\%$, Accuracy dev $\sim 94\%$).
  - Loss trên tập train giảm rất nhanh và hội tụ rất sớm (khoảng epoch thứ 3). Accuracy trên tập dev vẫn còn dao động mạnh và vẫn chưa có xu hướng hôi tụ.

4. Đánh giá mô hình

```
y_pred_total = []
y_true = []

for img, label in test_set_bai1:
  y_pred = model_2.predict(img)
  y_pred_total += [1 if i > 0.5 else 0 for i in y_pred]
  y_true += np.array(label).flatten().tolist()

accuracy = round(accuracy_score(y_true, y_pred_total)*100,2)
print('Accuracy test = {}%'.format(accuracy))
# Output: Accuracy test = 78.53%
precision = round(precision_score(y_true, y_pred_total)*100,2)
print('Precision test = {}%'.format(precision))
# Output: Precision test = 74.52%
recall = round(recall_score(y_true, y_pred_total)*100,2)
print('Recall test = {}%'.format(recall))
# Output: Recall test = 99.74%
f1 = round(f1_score(y_true, y_pred_total)*100,2)
print('F1-score test = {}%'.format(f1))
# Output: F1-score test = 85.31%
```

* **Nhận xét**:
  - Accuracy của mô hình khi kiểm tra trên tập test khá cao (Accuracy test = 78.53%). Có thể thấy đây là một mô hình khá tốt. Ngoài ra, với các độ đo khác model cũng cho kết quả khá cao (Precision = 74.52%, Recall = 99.74%, F1-score = 85.31%).
  - Có thể kết luận mô hình có khả năng phân lớp khá tốt.

Sử dụng ResNet-50 pre-trained

1. Xây dựng mô hình

```
resnet = ResNet50(include_top=False, input_shape = (IMG_SIZE, IMG_SIZE, IMG_CHANNEL))

for layer in resnet.layers:
  layer.trainable = False

flat = Flatten()(resnet.layers[-1].output)
fc1 = Dense(1024, activation='relu')(flat)
output = Dense(1, activation='sigmoid')(fc1)

model_3 = Model(inputs=resnet.inputs, outputs=output)
model_3.summary()
```

2. Huấn luyện mô hình

```
# Compile model
Optimizer = Adam(learning_rate=1e-3)
Loss = BinaryCrossentropy()
model_3.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])
# Training model
history_3 = model_3.fit(train_set_bai1,
                        validation_data=dev_set_bai1, epochs=10)
```

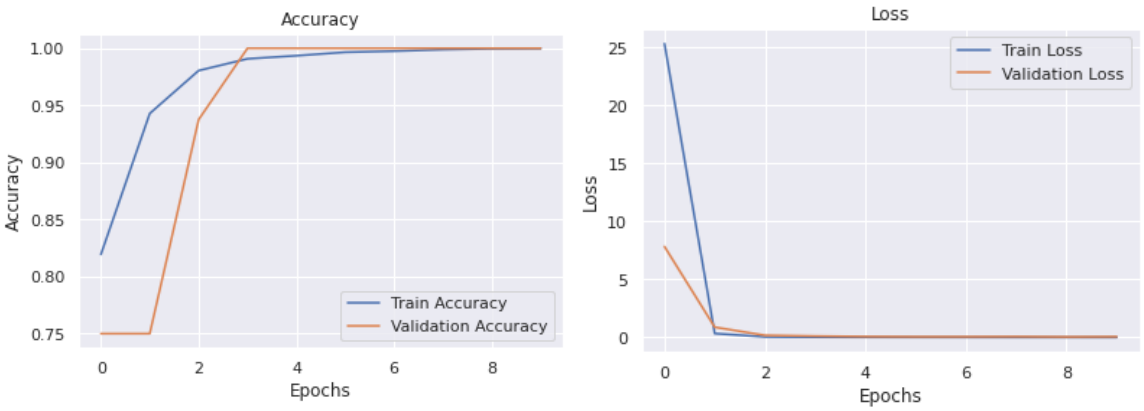
3. Visualization Loss & Accuracy training

```
plt.plot(history_3.history['accuracy'], label = 'Train Accuracy')
plt.plot(history_3.history['val_accuracy'], label = 'Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

– Output: Hình 3a

```
plt.plot(history_3.history['loss'], label = 'Train Loss')
plt.plot(history_3.history['val_loss'], label = 'Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 3b



(a) Accuracy

(b) Loss

Hình 3: Accuracy và Loss của model ResNet-50 Pre-trained bài 1

* **Nhận xét**:

– Accuracy trên cả 2 tập train và dev tăng nhanh và hội tụ sớm. Accuracy trên cả 2 tập đều rất cao (Accuracy train $\sim$ 99.99%, Accuracy dev = 100%).

– Loss trên cả 2 tập train và dev giảm rất nhanh và hội tụ rất sớm (ngay ở epoch đầu tiên).

4. Đánh giá mô hình

```
y_pred_total = []
y_true = []

for img, label in test_set_bai1:
  y_pred = model_3.predict(img)
  y_pred_total += [1 if i > 0.5 else 0 for i in y_pred]
  y_true += np.array(label).flatten().tolist()

accuracy = round(accuracy_score(y_true, y_pred_total)*100,2)
print('Accuracy test = {}%'.format(accuracy))
# Output: Accuracy test = 78.04%
precision = round(precision_score(y_true, y_pred_total)*100,2)
print('Precision test = {}%'.format(precision))
# Output: Precision test = 74.1%
recall = round(recall_score(y_true, y_pred_total)*100,2)
```

```
print('Recall test = {}%'.format(recall))
# Output: Recall test = 99.74%
f1 = round(f1_score(y_true, y_pred_total)*100,2)
print('F1-score test = {}%'.format(f1))
# Output: F1-score test = 85.03%
```

* **Nhận xét**:
  - Accuracy của mô hình khi kiểm tra trên tập test khá cao (Accuracy test = 78.04%). Có thể thấy đây là một mô hình khá tốt. Ngoài ra, với các độ đo khác model cũng cho kết quả khá cao (Precision = 74.1%, Recall = 99.74%, F1-score = 85.03%).
  - Có thể kết luận mô hình có khả năng phân lớp khá tốt.

* **Kết luận**:
  - Từ kết quả của 3 pre-trained model (VGG-16, VGG-19, ResNet-50) với bộ dữ liệu Chest X-ray thì có thể thấy VGG-16 pred-trained model cho kết quả tốt nhất, đứng thứ 2 là ResNet-50 pred-trained model và cuối cùng là VGG-19 pred-trained model.

Bài 2: Thực hiện các yêu cầu của Bài 1 với bộ dữ liệu Jewellery.

1. Chuẩn bị dữ liệu

```
!git clone https://github.com/princesegzy01/Jewellery-Classification.git

IMG_SIZE = 227
IMG_CHANNEL = 3
BATCH_SIZE = 256
COLOR_MODE = 'rgb'

IMG_SIZE = 227
IMG_CHANNEL = 3
BATCH_SIZE = 256
COLOR_MODE = 'rgb'

# Load train set
train_set_bai2=image_dataset_from_directory('/Jewellery-Classification/dataset/training'
                                            labels = 'inferred',
                                            label_mode = 'categorical',
                                            class_names = ['BRACELET',
                                                           'EARRINGS',
                                                           'NECKLACE',
                                                           'RINGS',
                                                           'WRISTWATCH'],
                                            color_mode = COLOR_MODE,
                                            batch_size = BATCH_SIZE,
                                            image_size = (IMG_SIZE, IMG_SIZE),
                                            interpolation = 'bilinear')
# Output: Found 1566 files belonging to 5 classes.
# Load dev set
dev_set_bai2=image_dataset_from_directory('/Jewellery-Classification/dataset/training',
                                          labels = 'inferred',
                                          label_mode = 'categorical',
                                          class_names = ['BRACELET',
                                                         'EARRINGS',
                                                         'NECKLACE',
                                                         'RINGS',
                                                         'WRISTWATCH'],
                                          color_mode = COLOR_MODE,
                                          batch_size = BATCH_SIZE,
                                          image_size = (IMG_SIZE, IMG_SIZE),
                                          interpolation = 'bilinear',
                                          subset="validation",
                                          seed=1, validation_split=0.1)
# Output: Found 1566 files belonging to 5 classes. Using 156 files for validation.
```

```
# Load test set
test_set_bai2=image_dataset_from_directory('/Jewellery-Classification/dataset/test',
                                           labels = 'inferred',
                                           label_mode = 'categorical',
                                           class_names = ['BRACELET',
                                                          'EARRINGS',
                                                          'NECKLACE',
                                                          'RINGS',
                                                          'WRISTWATCH'],
                                           color_mode = COLOR_MODE,
                                           batch_size = BATCH_SIZE,
                                           image_size = (IMG_SIZE, IMG_SIZE),
                                           interpolation = 'bilinear')
# Output: Found 250 files belonging to 5 classes.
```

Sử dụng VGG-16 pre-trained

1. Xây dựng mô hình

```
vgg = VGG16(include_top=False, input_shape = (IMG_SIZE, IMG_SIZE, IMG_CHANNEL))

for layer in vgg.layers:
  layer.trainable = False

flat = Flatten()(vgg.layers[-1].output)
fc1 = Dense(1024, activation='relu')(flat)
output = Dense(5, activation='softmax')(fc1)

model_4 = Model(inputs=vgg.inputs, outputs=output)

model_4.summary()
```

2. Huấn luyện mô hình

```
# Compile model
Optimizer = Adam(learning_rate=1e-3)
Loss = CategoricalCrossentropy()
model_4.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_4 = model_4.fit(train_set_bai2,
                        validation_data=dev_set_bai2, epochs=10)
```

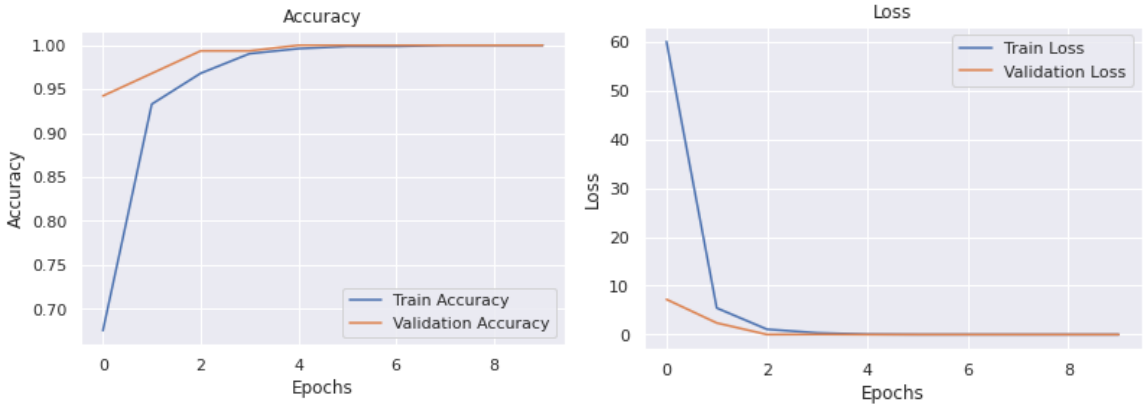
3. Visualization Loss & Accuracy training

```
plt.plot(history_4.history['accuracy'], label = 'Train Accuracy')
plt.plot(history_4.history['val_accuracy'], label = 'Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

– Output: Hình 4a

```
plt.plot(history_4.history['loss'], label = 'Train Loss')
plt.plot(history_4.history['val_loss'], label = 'Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 4b



(a) Accuracy

(b) Loss

Hình 4: Accuracy và Loss của model VGG-16 Pre-trained bài 2

* **Nhận xét**:

- Accuracy trên cả 2 tập train và dev tăng nhanh và hội tụ sớm. Accuracy trên cả 2 tập đều rất cao (Accuracy train = 100%, Accuracy dev = 100%).
- Loss trên 2 tập train và dev đều giảm rất nhanh và hội tụ rất sớm (ở epoch thứ 2).

4. Đánh giá mô hình

```
y_pred_total = []
y_true = []

for img, label in test_set_bai2:
  y_pred = model_4.predict(img)
  y_pred_total += np.argmax(y_pred, axis=-1).tolist()
  y_true += np.array(np.argmax(label, axis=-1)).flatten().tolist()

accuracy = round(accuracy_score(y_true, y_pred_total)*100,2)
print('Accuracy test = {}%'.format(accuracy))
# Output: F1-score test = 90.4%
precision = round(precision_score(y_true, y_pred_total, average='micro')*100,2)
print('Precision micro test = {}%'.format(precision))
# Output: Precision micro test = 90.4%
precision = round(precision_score(y_true, y_pred_total, average='macro')*100,2)
print('Precision macro test = {}%'.format(precision))
# Output: Precision macro test = 92.2%
recall = round(recall_score(y_true, y_pred_total, average='micro')*100,2)
print('Recall micro test = {}%'.format(recall))
# Output: Recall micro test = 90.4%
recall = round(recall_score(y_true, y_pred_total, average='macro')*100,2)
print('Recall macro test = {}%'.format(recall))
# Output: Recall macro test = 90.4%
f1 = round(f1_score(y_true, y_pred_total, average='micro')*100,2)
print('F1-score micro test = {}%'.format(f1))
# Output: F1-score micro test = 90.4%
f1 = round(f1_score(y_true, y_pred_total, average='macro')*100,2)
print('F1-score macro test = {}%'.format(f1))
# Output: F1-score macro test = 89.99%
```

* **Nhận xét**:

- Accuracy của mô hình khi kiểm tra trên tập test cao (Accuracy test = 90.4%). Có thể thấy đây là một mô hình rất tốt. Ngoài ra, với các độ đo khác model cũng cho kết quả cao (Precision micro = 90.4%, Precision macro = 90.2%, Recall micro = 90.4%, Recall macro = 90.4%, F1-score micro = 90.4%, F1-score macro = 89.99%).
- Có thể kết luận mô hình có khả năng phân lớp rất tốt.

Sử dụng VGG-19 pre-trained

1. Xây dựng mô hình

```
vgg = VGG19(include_top=False, input_shape = (IMG_SIZE, IMG_SIZE, IMG_CHANNEL))
for layer in vgg.layers:
  layer.trainable = False
flat = Flatten()(vgg.layers[-1].output)
fc1 = Dense(1024, activation='relu')(flat)
output = Dense(5, activation='softmax')(fc1)
model_5 = Model(inputs=vgg.inputs, outputs=output)
model_5.summary()
```

2. Huấn luyện mô hình

```
# Compile model
Optimizer = Adam(learning_rate=1e-3)
Loss = CategoricalCrossentropy()
model_5.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_5 = model_5.fit(train_set_bai2,
                        validation_data=dev_set_bai2, epochs=10)
```

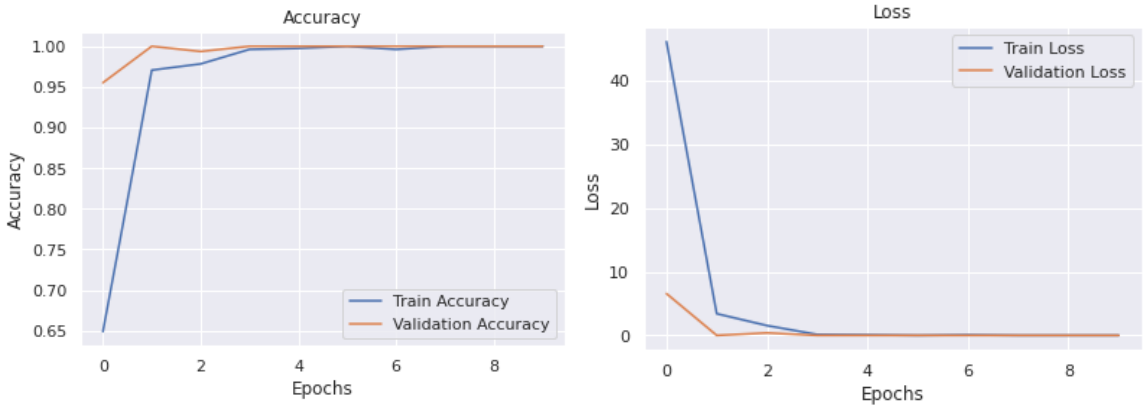
3. Visualization Loss & Accuracy training

```
plt.plot(history_5.history['accuracy'], label = 'Train Accuracy')
plt.plot(history_5.history['val_accuracy'], label = 'Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

– Output: Hình 5a

```
plt.plot(history_5.history['loss'], label = 'Train Loss')
plt.plot(history_5.history['val_loss'], label = 'Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 5b



(a) Accuracy

(b) Loss

Hình 5: Accuracy và Loss của model VGG-19 Pre-trained bài 2

* **Nhận xét**:
  - Accuracy trên cả 2 tập train và dev tăng nhanh và hội tụ sớm. Accuracy trên cả 2 tập đều rất cao (Accuracy train = 100%, Accuracy dev = 100%).
  - Loss trên 2 tập train và dev đều giảm rất nhanh và hội tụ rất sớm (ở epoch thứ 3).

4. Đánh giá mô hình

```
y_pred_total = []
y_true = []

for img, label in test_set_bai2:
  y_pred = model_5.predict(img)
  y_pred_total += np.argmax(y_pred, axis=-1).tolist()
  y_true += np.array(np.argmax(label, axis=-1)).flatten().tolist()

accuracy = round(accuracy_score(y_true, y_pred_total)*100,2)
print('Accuracy test = {}%'.format(accuracy))
# Output: F1-score test = 91.6%
precision = round(precision_score(y_true, y_pred_total, average='micro')*100,2)
print('Precision micro test = {}%'.format(precision))
# Output: Precision micro test = 91.6%
precision = round(precision_score(y_true, y_pred_total, average='macro')*100,2)
print('Precision macro test = {}%'.format(precision))
# Output: Precision macro test = 93.44%
recall = round(recall_score(y_true, y_pred_total, average='micro')*100,2)
print('Recall micro test = {}%'.format(recall))
# Output: Recall micro test = 91.6%
recall = round(recall_score(y_true, y_pred_total, average='macro')*100,2)
print('Recall macro test = {}%'.format(recall))
# Output: Recall macro test = 91.6%
f1 = round(f1_score(y_true, y_pred_total, average='micro')*100,2)
print('F1-score micro test = {}%'.format(f1))
# Output: F1-score micro test = 91.6%
f1 = round(f1_score(y_true, y_pred_total, average='macro')*100,2)
print('F1-score macro test = {}%'.format(f1))
# Output: F1-score macro test = 91.59%
```

* **Nhận xét**:
  - Accuracy của mô hình khi kiểm tra trên tập test cao (Accuracy test = 91.6%). Có thể thấy đây là một mô hình rất tốt. Ngoài ra, với các độ đo khác model cũng cho kết quả cao (Precision micro = 91.6%, Precision macro = 93.44%, Recall micro = 91.6%, Recall macro = 91.6%, F1-score micro = 91.6%, F1-score macro = 91.59%).
  - Có thể kết luận mô hình có khả năng phân lớp rất tốt.

Sử dụng ResNet-50 pre-trained

1. Xây dựng mô hình

```
resnet = ResNet50(include_top=False, input_shape = (IMG_SIZE, IMG_SIZE, IMG_CHANNEL))

for layer in resnet.layers:
  layer.trainable = False

flat = Flatten()(resnet.layers[-1].output)
fc1 = Dense(1024, activation='relu')(flat)
output = Dense(5, activation='softmax')(fc1)

model_6 = Model(inputs=resnet.inputs, outputs=output)

model_6.summary()
```

2. Huấn luyện mô hình

```
# Compile model
Optimizer = Adam(learning_rate=1e-3)
Loss = CategoricalCrossentropy()
model_6.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_6 = model_6.fit(train_set_bai2,
                        validation_data=dev_set_bai2, epochs=10)
```

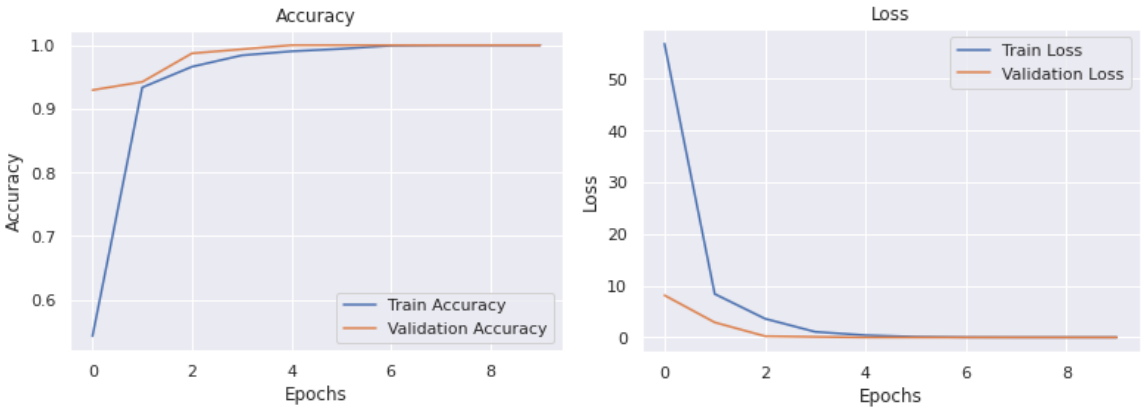
3. Visualization Loss & Accuracy training

```
plt.plot(history_6.history['accuracy'], label = 'Train Accuracy')
plt.plot(history_6.history['val_accuracy'], label = 'Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

– Output: Hình 6a

```
plt.plot(history_6.history['loss'], label = 'Train Loss')
plt.plot(history_6.history['val_loss'], label = 'Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 6b



(a) Accuracy

(b) Loss

Hình 6: Accuracy và Loss của model ResNet-50 Pre-trained bài 2

* **Nhận xét**:

– Accuracy trên cả 2 tập train và dev tăng nhanh và hội tụ sớm. Accuracy trên cả 2 tập đều rất cao (Accuracy train = 100%, Accuracy dev = 100%).

– Loss trên 2 tập train và dev đều giảm rất nhanh và hội tụ rất sớm (ở epoch thứ 3).

4. Đánh giá mô hình

```
y_pred_total = []
y_true = []

for img, label in test_set_bai2:
  y_pred = model_4.predict(img)
  y_pred_total += np.argmax(y_pred, axis=-1).tolist()
```

```
    y_true += np.array(np.argmax(label, axis=-1)).flatten().tolist()

accuracy = round(accuracy_score(y_true, y_pred_total)*100,2)
print('Accuracy test = {}%'.format(accuracy))
# Output: F1-score test = 97.2%
precision = round(precision_score(y_true, y_pred_total, average='micro')*100,2)
print('Precision micro test = {}%'.format(precision))
# Output: Precision micro test = 97.2%
precision = round(precision_score(y_true, y_pred_total, average='macro')*100,2)
print('Precision macro test = {}%'.format(precision))
# Output: Precision macro test = 97.25%
recall = round(recall_score(y_true, y_pred_total, average='micro')*100,2)
print('Recall micro test = {}%'.format(recall))
# Output: Recall micro test = 97.2%
recall = round(recall_score(y_true, y_pred_total, average='macro')*100,2)
print('Recall macro test = {}%'.format(recall))
# Output: Recall macro test = 97.2%
f1 = round(f1_score(y_true, y_pred_total, average='micro')*100,2)
print('F1-score micro test = {}%'.format(f1))
# Output: F1-score micro test = 97.2%
f1 = round(f1_score(y_true, y_pred_total, average='macro')*100,2)
print('F1-score macro test = {}%'.format(f1))
# Output: F1-score macro test = 97.21%
```

* **Nhận xét**:
  - Accuracy của mô hình khi kiểm tra trên tập test rất cao (Accuracy test = 97.2%). Có thể thấy đây là một mô hình rất tốt. Ngoài ra, với các độ đo khác model cũng cho kết quả rất cao (Precision micro = 97.2%, Precision macro = 97.25%, Recall micro = 97.2%, Recall macro = 97.2%, F1-score micro = 97.2%, F1-score macro = 97.21%).
  - Có thể kết luận mô hình có khả năng phân lớp rất tốt.

* **Kết luận**:
  - Từ kết quả của 3 pre-trained model (VGG-16, VGG-19, ResNet-50) với bộ dữ liệu Jewellery thì có thể thấy ResNet-50 pred-trained model cho kết quả tốt nhất, đứng thứ 2 là VGG–19 pred-trained model và cuối cùng là VGG–16 pred-trained model.