

Deep Learning trong khoa học dữ liệu

Bài tập thực hành 4 - DS201.M11.2

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

Thứ 4, ngày 10 tháng 11 năm 2021

Bài tập: MẠNG NEURAL HỒI QUY

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 keras.datasets import imdb
from keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import SimpleRNN, LSTM, Bidirectional, GRU
from keras.layers import Dense, Dropout, Input, Embedding
from tensorflow.keras.optimizers import Adam
from keras.losses import BinaryCrossentropy
from sklearn.metrics import accuracy_score

sn.set() # Set theme
```

2. Load bộ dữ liệu IMDB

```
(train_data, train_targets), (test_data, test_targets) = imdb.load_data(num_words=10000)
```

3. Chuẩn bị dữ liệu

```
# Lay cac tu vung va chi so tuong ung
word_index = imdb.get_word_index()
# Chuyen cau du lieu ve dang van ban
inverted_word_index = dict((i, word) for (word, i) in word_index.items())
decoded_sequence = " ".join(inverted_word_index[i] for i in train_data[0])
# Padding chieu dai cua sequence la 200
maxlen = 200
X_train = pad_sequences(train_data, maxlen=maxlen)
X_test = pad_sequences(test_data, maxlen=maxlen)
# categorical targets
y_train = to_categorical(train_targets, num_classes = 2)
y_test = test_targets
# Mo rong chieu cua du lieu
X_train_new = np.expand_dims(X_train, axis=2)
X_test_new = np.expand_dims(X_test, axis=2)
```

Bài 1: Tăng số lượng epoch lên 20 và thực hiện huấn luyện mô hình. Cho biết độ chính xác của mô hình là bao nhiêu? Vẽ đồ thị học.

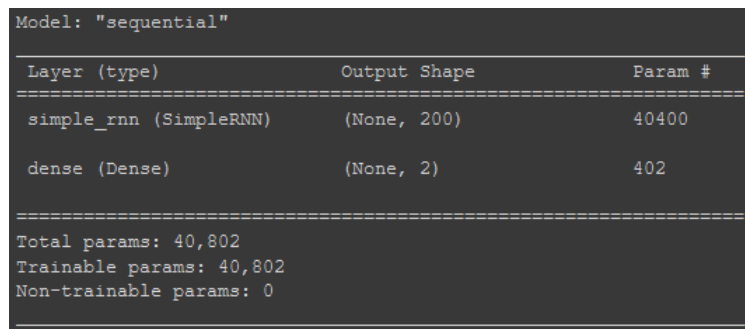
1. Xây dựng mô hình SimpleRNN bằng Keras

```
# Khởi tạo model
model_1 = Sequential()

# Thêm một lớp Input
model_1.add(Input(shape=(None, 1), dtype="float64"))
# Thêm một lớp SimpleRNN với activation là relu
model_1.add(SimpleRNN(200, return_sequences=False,
                      return_state=False, activation='relu'))
# Thêm một lớp Dense với activation là sigmoid
model_1.add(Dense(2, activation='sigmoid'))

model_1.summary()
```

– Output: Hình 1



```
Model: "sequential"
Layer (type)                Output Shape         Param #
=====
simple_rnn (SimpleRNN)       (None, 200)          40400
dense (Dense)                (None, 2)            402
=====
Total params: 40,802
Trainable params: 40,802
Non-trainable params: 0
```

Hình 1: Summary model

2. Huấn luyện mô hình

```
# Compile model
Optimizer = Adam(learning_rate=1e-4)
Loss = BinaryCrossentropy()
model_1.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_1 = model_1.fit(X_train_new, y_train,
                       validation_split=0.1,
                       batch_size=128, epochs=20)
```

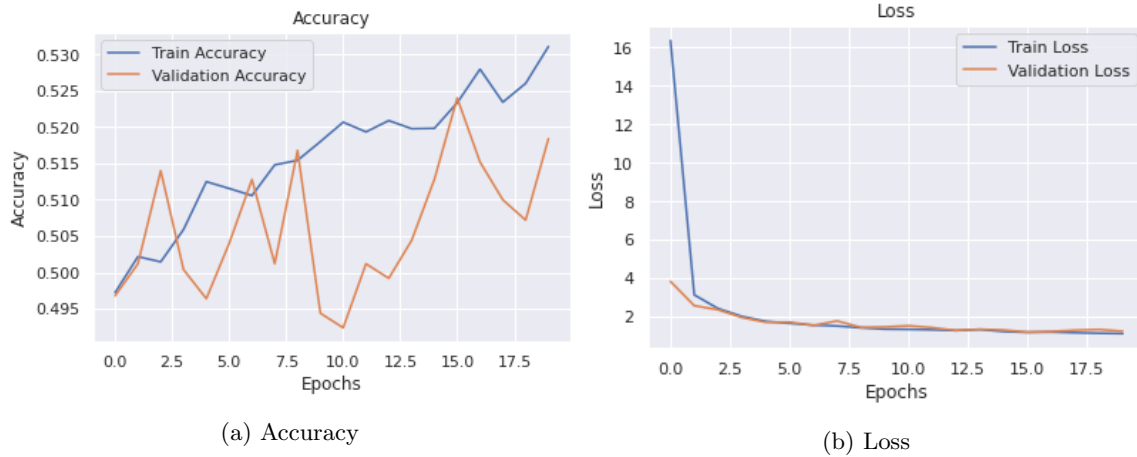
3. Visualization Loss & Accuracy training

```
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 2a

```
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 2b



Hình 2: Accuracy và Loss của model bài 1

* **Nhận xét:**

- Accuracy trên tập train và tập validation đều thấp và dao động rất lớn (49% – 53%). Accuracy tập train vẫn có xu hướng tăng nhưng ở tập validation thì không (dao động mạnh lúc tăng lúc giảm).
- Loss trên cả 2 tập train và validation đều giảm rất nhanh và hội tụ rất sớm (khoảng epoch thứ 5).

4. Đánh giá mô hình

```
y_pred = model_1.predict(X_test_new)
y_pred = np.argmax(y_pred, axis = -1)
accuracy = round(accuracy_score(y_test, y_pred)*100,2)
print('Accuracy test = {}'.format(accuracy))
# Output: Accuracy test = 49.97%
```

* **Nhận xét:**

- Accuracy của mô hình khi kiểm tra trên tập test thấp (Accuracy test = 49.97%). Có thể thấy đây là một mô hình không tốt.

Bài 2: Hãy thử thêm 1 lớp Simple RNN nữa vào mô hình và cho biết kết quả độ chính xác của mô hình như thế nào? Ghi chú: Thêm vào trước lớp SimpleRNN hiện tại, đặt `return_sequence = True`.

Ghi chú: Thêm vào trước lớp SimpleRNN hiện tại, đặt `return_sequence = True`.

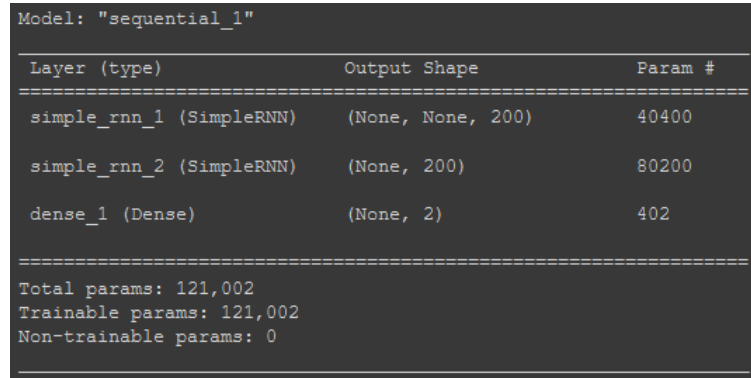
1. Xây dựng mô hình SimpleRNN bằng Keras

```
# Khởi tạo model
model_2 = Sequential()

# Thêm một lớp Input
model_2.add(Input(shape=(None, 1), dtype="float64"))
# Thêm một lớp SimpleRNN với activation là relu
model_2.add(SimpleRNN(200, return_sequences=True,
                      return_state=False, activation='relu'))
# Thêm một lớp SimpleRNN với activation là relu
model_2.add(SimpleRNN(200, return_sequences=False,
                      return_state=False, activation='relu'))
# Thêm một lớp Dense với activation là sigmoid
model_2.add(Dense(2, activation='sigmoid'))

model_2.summary()
```

– Output: Hình 3



```
Model: "sequential_1"
Layer (type)                Output Shape              Param #
=====
simple_rnn_1 (SimpleRNN)      (None, None, 200)        40400
simple_rnn_2 (SimpleRNN)      (None, 200)              80200
dense_1 (Dense)              (None, 2)                402
=====
Total params: 121,002
Trainable params: 121,002
Non-trainable params: 0
```

Hình 3: Summary model

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(X_train_new, y_train,
                        validation_split=0.1,
                        batch_size=128, epochs=20)
```

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 4a

```
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 4b

* Nhận xét:

- Accuracy trên tập train và tập validation đều thấp và dao động rất lớn (48% – 53%). Accuracy tập train vẫn có xu hướng tăng nhưng ở tập validation thì không (dao động mạnh lúc tăng lúc giảm).
- Loss trên cả 2 tập train và validation đều giảm rất nhanh và hội tụ rất sớm (khoảng epoch thứ 10).



Hình 4: Accuracy và Loss của model bài 2

4. Đánh giá mô hình

```

y_pred = model_1.predict(X_test_new)
y_pred = np.argmax(y_pred, axis = -1)
accuracy = round(accuracy_score(y_test, y_pred)*100,2)
print('Accuracy test = {}'.format(accuracy))
# Output: Accuracy test = 49.98%

```

* Nhận xét:

- Accuracy của mô hình khi kiểm tra trên tập test thấp (Accuracy test = 49.98%). Có thể thấy đây là một mô hình không tốt.

Bài 3: Xây dựng mô hình gồm 1 lớp Embedding và 1 lớp LSTM kết nối với nhau. Cho biết độ chính xác của mô hình là bao nhiêu? Vẽ đồ thị học.

1. Xây dựng mô hình LSTM bằng Keras

```

# Khởi tạo model
model_3 = Sequential()
# Thêm một lớp Input
model_3.add(Input(shape=(None, ), dtype="float64"))
# Thêm một lớp Embedding
model_3.add(Embedding(len(word_index), 128))
# Thêm một lớp Dropout
model_3.add(Dropout(0.3))
# Thêm một lớp LSTM
model_3.add(LSTM(200, return_sequences=False,
                 kernel_regularizer=l2(0.01),
                 recurrent_regularizer=l2(0.01),
                 dropout=0.33))
# Thêm một lớp Dropout
model_3.add(Dropout(0.3))
# Thêm một lớp Dense với activation là sigmoid
model_3.add(Dense(2, activation='sigmoid'))

model_3.summary()

```

- Output: Hình 5

Model: "sequential_2"		
Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, None, 128)	11338752
dropout (Dropout)	(None, None, 128)	0
lstm (LSTM)	(None, 200)	263200
dropout_1 (Dropout)	(None, 200)	0
dense_2 (Dense)	(None, 2)	402
Total params: 11,602,354		
Trainable params: 11,602,354		
Non-trainable params: 0		

Hình 5: Summary model

2. Huấn luyện mô hình

```
# Compile model
Optimizer = Adam(learning_rate=1e-4)
Loss = BinaryCrossentropy()
model_3.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_3 = model_3.fit(X_train_new, y_train,
                        validation_split=0.1,
                        batch_size=128, 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()
```

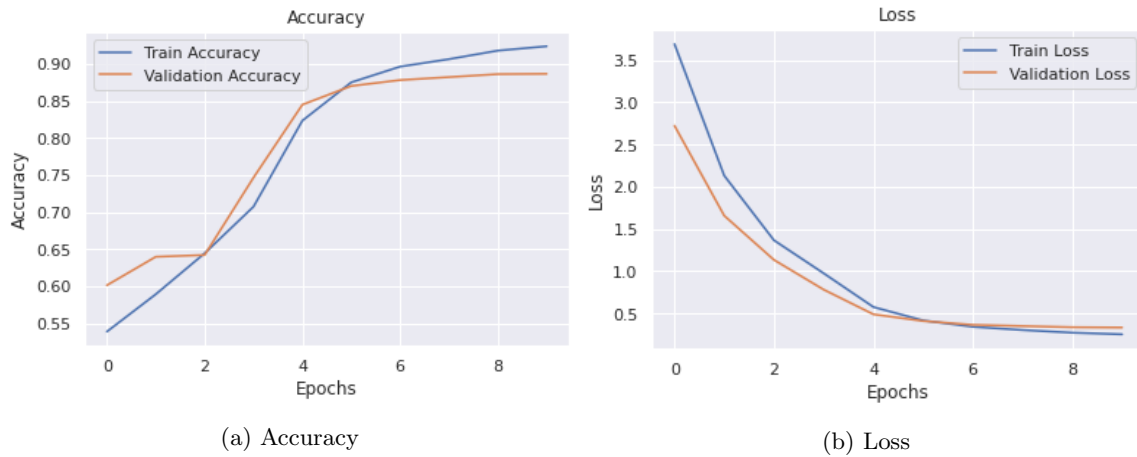
– Output: Hình 6a

```
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 6b

* Nhận xét:

- Accuracy trên tập train và tập validation đều cao (accuracy train $\sim 92.5\%$, accuracy validation $\sim 88.7\%$).
- Loss trên cả 2 tập train và validation đều giảm rất nhanh và hội tụ rất sớm (khoảng epoch thứ 6).



Hình 6: Accuracy và Loss của model bài 3

4. Đánh giá mô hình

```

y_pred = model_3.predict(X_test_new)
y_pred = np.argmax(y_pred, axis = -1)
accuracy = round(accuracy_score(y_test, y_pred)*100,2)
print('Accuracy test = {}%'.format(accuracy))
# Output: Accuracy test = 87.57%

```

* Nhận xét:

- Accuracy của mô hình khi kiểm tra trên tập test cao (Accuracy test = 87.57%). Có thể thấy đây là một mô hình tốt.

Bài 4: Xây dựng mô hình gồm 1 lớp Embedding và 2 lớp LSTM kết nối với nhau. Cho biết độ chính xác của mô hình là bao nhiêu? Vẽ đồ thị học.

1. Xây dựng mô hình LSTM bằng Keras

```

# Khởi tạo model
model_4 = Sequential()
# Thêm một lớp Input
model_4.add(Input(shape=(None, ), dtype="float64"))
# Thêm một lớp Embedding
model_4.add(Embedding(len(word_index), 128))
# Thêm một lớp Dropout
model_4.add(Dropout(0.3))
# Thêm một lớp LSTM
model_4.add(LSTM(200, return_sequences=True,
                 kernel_regularizer=l2(0.01),
                 recurrent_regularizer=l2(0.01),
                 dropout=0.33))
# Thêm một lớp Dropout
model_4.add(Dropout(0.3))
# Thêm một lớp LSTM
model_4.add(LSTM(64, return_sequences=False))
# Thêm một lớp Dropout
model_4.add(Dropout(0.3))
# Thêm một lớp Dense với activation là sigmoid
model_4.add(Dense(2, activation='sigmoid'))

model_4.summary()

```

- Output: Hình 7

Model: "sequential_3"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, None, 128)	11338752
dropout_2 (Dropout)	(None, None, 128)	0
lstm_1 (LSTM)	(None, None, 200)	263200
dropout_3 (Dropout)	(None, None, 200)	0
lstm_2 (LSTM)	(None, 64)	67840
dropout_4 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 2)	130

=====
Total params: 11,669,922
Trainable params: 11,669,922
Non-trainable params: 0
=====

Hình 7: Summary model

2. Huấn luyện mô hình

```
# Compile model
Optimizer = Adam(learning_rate=1e-4)
Loss = BinaryCrossentropy()
model_4.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_4 = model_4.fit(X_train_new, y_train,
                        validation_split=0.1,
                        batch_size=128, 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()
```

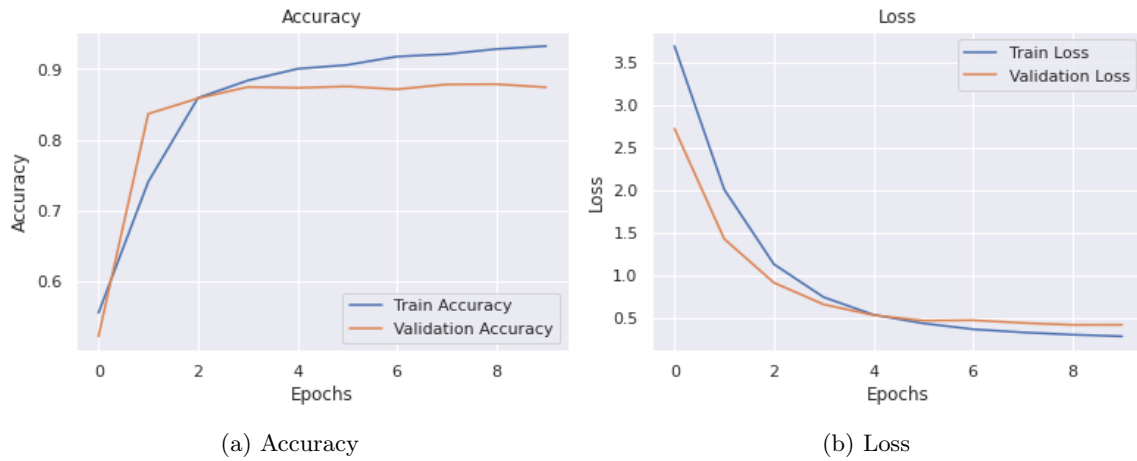
– Output: Hình 8a

```
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 8b

* Nhận xét:

- Accuracy trên tập train và tập validation đều cao (accuracy train $\sim 93.3\%$, accuracy validation $\sim 87.5\%$).
- Loss trên cả 2 tập train và validation đều giảm rất nhanh và hội tụ rất sớm (khoảng epoch thứ 6).



Hình 8: Accuracy và Loss của model bài 4

4. Đánh giá mô hình

```

y_pred = model_4.predict(X_test_new)
y_pred = np.argmax(y_pred, axis = -1)
accuracy = round(accuracy_score(y_test, y_pred)*100,2)
print('Accuracy test = {}'.format(accuracy))
# Output: Accuracy test = 86.19%

```

* Nhận xét:

- Accuracy của mô hình khi kiểm tra trên tập test cao (Accuracy test = 86.19%). Có thể thấy đây là một mô hình tốt.

Bài 5: Tìm hiểu GRU và xây dựng mô hình với lớp GRU (mô hình giống Bài 3, nhưng thay lớp LSTM thành GRU). So sánh kết quả của 2 mô hình.

1. Xây dựng mô hình GRU bằng Keras

```

# Khởi tạo model
model_5 = Sequential()
# Thêm một lớp Input
model_5.add(Input(shape=(None, ), dtype="float64"))
# Thêm một lớp Embedding
model_5.add(Embedding(len(word_index), 128))
# Thêm một lớp Dropout
model_5.add(Dropout(0.3))
# Thêm một lớp GRU
model_5.add(GRU(200, return_sequences=False,
                kernel_regularizer=l2(0.01),
                recurrent_regularizer=l2(0.01),
                dropout=0.33))
# Thêm một lớp Dropout
model_5.add(Dropout(0.3))
# Thêm một lớp Dense với activation là sigmoid
model_5.add(Dense(2, activation='sigmoid'))

model_5.summary()

```

- Output: Hình 9

Model: "sequential_4"

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, None, 128)	11338752
dropout_5 (Dropout)	(None, None, 128)	0
gru (GRU)	(None, 200)	198000
dropout_6 (Dropout)	(None, 200)	0
dense_4 (Dense)	(None, 2)	402

=====
Total params: 11,537,154
Trainable params: 11,537,154
Non-trainable params: 0

Hình 9: Summary model

2. Huấn luyện mô hình

```
# Compile model
Optimizer = Adam(learning_rate=1e-4)
Loss = BinaryCrossentropy()
model_5.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_5 = model_5.fit(X_train_new, y_train,
                        validation_split=0.1,
                        batch_size=128, 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()
```

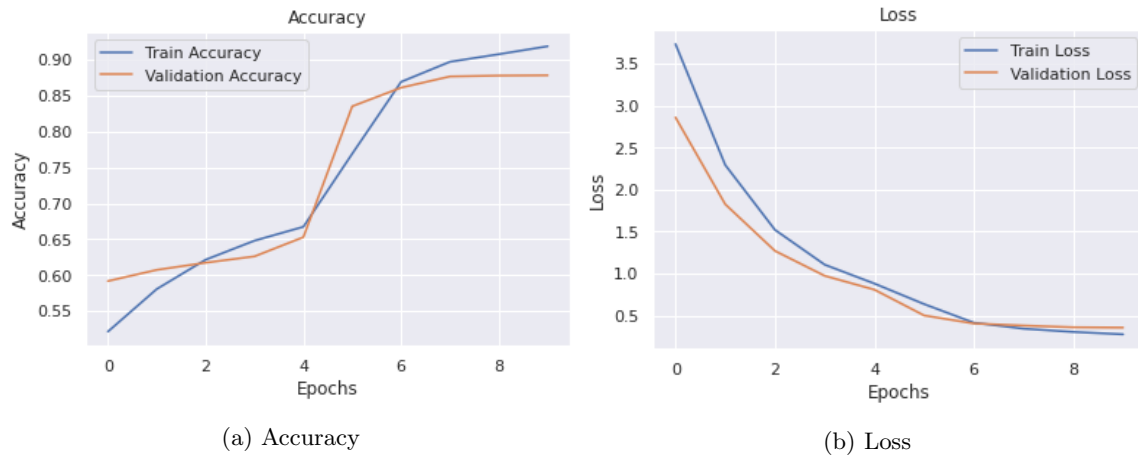
– Output: Hình 10a

```
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 10b

* Nhận xét:

- Accuracy trên tập train và tập validation đều cao (accuracy train $\sim 91.9\%$, accuracy validation $\sim 87.8\%$).
- Loss trên cả 2 tập train và validation đều giảm rất nhanh và hội tụ rất sớm (khoảng epoch thứ 6).



Hình 10: Accuracy và Loss của model bài 3

4. Đánh giá mô hình

```

y_pred = model_5.predict(X_test_new)
y_pred = np.argmax(y_pred, axis = -1)
accuracy = round(accuracy_score(y_test, y_pred)*100,2)
print('Accuracy test = {}'.format(accuracy))
# Output: Accuracy test = 87.29%

```

* **Nhận xét:**

- Accuracy của mô hình khi kiểm tra trên tập test cao (Accuracy test = 87.29%). Có thể thấy đây là một mô hình tốt.

* **So sánh:**

- Kết quả của model dùng GRU (bài 5) nhìn chung cũng xấp xỉ như kết quả của model dùng LSTM (bài 3) (bài 3: accuracy = 87.57%, bài 5: accuracy = 87.29%).