

Deep Learning trong khoa học dữ liệu

Bài tập thực hành 2 - 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 13 tháng 10 năm 2021

Bài tập: CÁC KỸ THUẬT TỐI ƯU HOÁ MÔ HÌNH

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.mnist import load_data
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import layers
from keras.layers import Dense, Activation
from keras.models import Sequential
from sklearn.metrics import accuracy_score, confusion_matrix
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
from tensorflow.keras.losses import BinaryCrossentropy
from keras.models import load_model
from sklearn.model_selection import train_test_split
from keras.regularizers import l2
from keras.initializers import GlorotUniform, Zeros, Ones
from keras.callbacks import EarlyStopping

sn.set() # Set theme
```

2. Load bộ dữ liệu MNIST

```
(X_train, y_train), (X_test, y_test) = load_data()
```

3. Chuẩn bị dữ liệu

```
X_train, X_dev, y_train, y_dev = train_test_split(X_train, y_train,
                                                    test_size=0.1)

print('X_train = {}'.format(X_train.shape))
# Output: X_train = (54000, 28, 28)
print('y_train = {}'.format(y_train.shape))
# Output: y_train = (54000,)
print('X_dev = {}'.format(X_dev.shape))
# Output: X_dev = (6000, 28, 28)
print('y_dev = {}'.format(y_dev.shape))
# Output: y_dev = (6000,)
print('X_test = {}'.format(X_test.shape))
# Output: X_test = (10000, 28, 28)
print('y_test = {}'.format(y_test.shape))
# Output: y_test = (10000,)
```

- X_train gồm có 60,000 bức ảnh trắng đen (channel = 1) về chữ số viết tay với kích thước mỗi bức ảnh là 28×28 pixel dùng để train model, y_train là các nhãn tương ứng X_train.
- X_test gồm có 10,000 bức ảnh trắng đen (channel = 1) về chữ số viết tay với kích thước mỗi bức ảnh là 28×28 pixel dùng để test model, y_test là các nhãn tương ứng của X_test.

```
# Reshape kích thước của dữ liệu
X_train_reshaped = X_train.reshape(-1, 28*28)
X_dev_reshaped = X_dev.reshape(-1, 28*28)
X_test_reshaped = X_test.reshape(-1, 28*28)

y_train_new = to_categorical(y_train, num_classes = 10)
y_dev_new = to_categorical(y_dev, num_classes = 10)

# Xem kích thước của dữ liệu sau khi reshape
print('X_train_reshaped = {}'.format(X_train_reshaped.shape))
# Output: X_train_reshaped = (54000, 784)
print('y_train_new = {}'.format(y_train_new.shape))
# Output: y_train_new = (54000, 10)
print('X_dev_reshaped = {}'.format(X_dev_reshaped.shape))
# Output: X_dev_reshaped = (6000, 784)
print('y_dev_new = {}'.format(y_dev_new.shape))
# Output: y_dev_new = (6000, 10)
print('X_test_reshaped = {}'.format(X_test_reshaped.shape))
# Output: X_test_reshaped = (10000, 784)
print('y_test = {}'.format(y_test.shape))
# Output: y_test = (10000,)
```

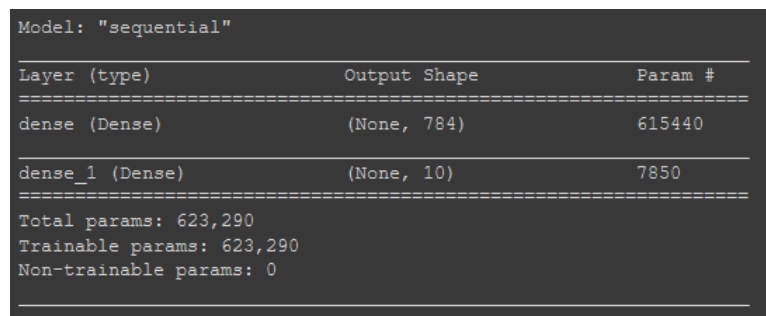
Bài 1: Thực hiện huấn luyện mô hình mạng neural ở Mục 3 trên bộ dữ liệu MNIST. Về đồ thị học của mô hình với thông số accuracy và loss.

1. Xây dựng mạng neural bằng Keras

```
# Khởi tạo model
model_1 = Sequential()
# Add layer Dense
model_1.add(Dense(784, input_shape=(784, ), activation='relu'))
model_1.add(Dense(10, input_shape=(10, ), activation='sigmoid'))

model_1.summary()
```

- Output: Hình 1



| Layer (type) | Output Shape | Param # |
|---------------------------|--------------|---------|
| dense (Dense) | (None, 784) | 615440 |
| dense_1 (Dense) | (None, 10) | 7850 |
| Total params: 623,290 | | |
| Trainable params: 623,290 | | |
| Non-trainable params: 0 | | |

Hình 1: Summary model

* Nhận xét:

- Model có 1 hidden layer Dense, có shape là (None, 784) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model (tổng số điểm dữ liệu training), 784 là shape dữ liệu đầu vào của model.
- Tổng số params mà model cần phải học là 623,290 params.

2. Huấn luyện mô hình

```
# Compile model
Optimizer = SGD(learning_rate=0.01)
Loss = BinaryCrossentropy()
model_1.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_1 = model_1.fit(X_train_resaped, y_train_new,
                        validation_data=(X_dev_resaped, y_dev_new),
                        batch_size=128, epochs=30)
```

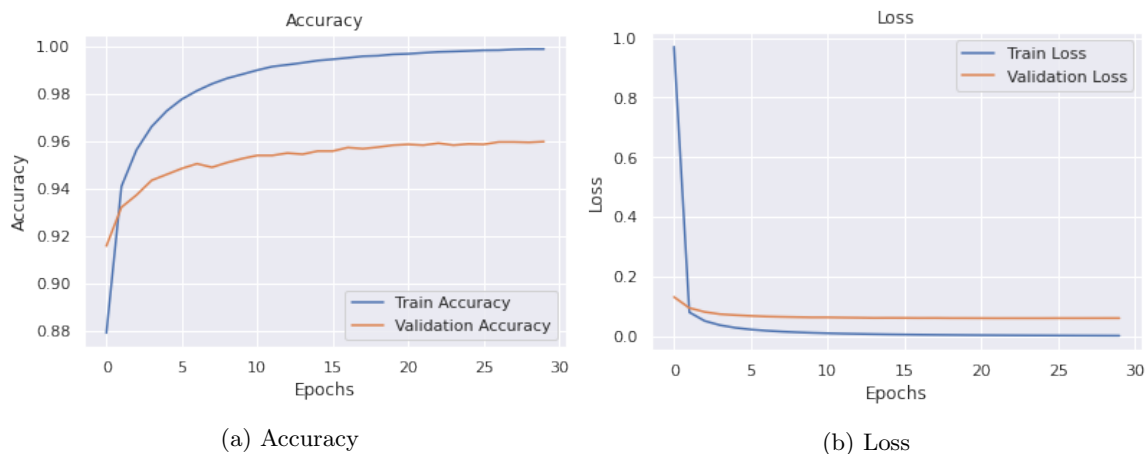
3. Visualization Loss & Accuracy training

```
plt.figure(0)
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.figure(0)
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 vẫn tiếp tục tăng đến epoch thứ 15 mới bắt đầu hội tụ, nhưng accuracy trên tập dev đã bắt đầu hội tụ ở epoch thứ 10. Accuracy trên tập train rất cao ($\sim 99.9\%$), accuracy trên tập dev cao ($\sim 96.0\%$), chênh lệch khoảng 4.0%. Nên có thể đã xảy ra hiện tượng Overfitting.
- Loss giảm rất nhanh và hội tụ rất sớm.

4. Đánh giá mô hình

```
y_pred = model_1.predict(X_test_reshaped)
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 = 96.18%
```

* **Nhận xét:**

- Accuracy của mô hình khi kiểm tra trên tập test cao (Accuracy test = 96.18%).

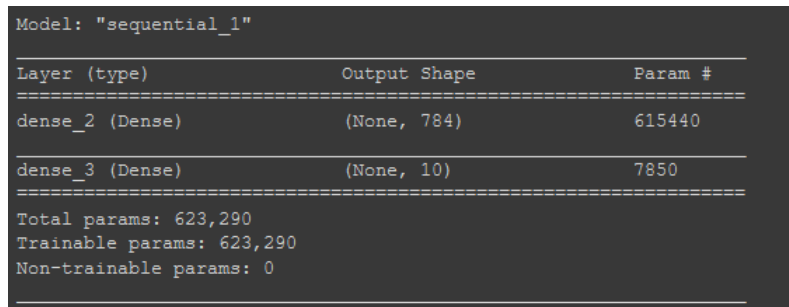
Bài 2: Thực hiện huấn luyện mô hình mạng neural ở bài 1 với kỹ thuật regularization cho tham số W và b với λ là 0.01. Cho biết kết quả huấn luyện mô hình trước và sau khi áp dụng kỹ thuật regularization (Dựa vào đồ thị học).

1. Xây dựng mạng neural bằng Keras

```
Khoi tao model
model_2 = Sequential()
# Áp dụng kỹ thuật regularization với lambda = 0.01
model_2.add(Dense(784, input_shape=(784, ),
                  kernel_regularizer = l2 (0.01),
                  bias_regularizer = l2 (0.01),
                  activation='relu'))
model_2.add(Dense(10, input_shape=(10, ),
                  kernel_regularizer = l2 (0.01),
                  bias_regularizer = l2 (0.01),
                  activation='sigmoid'))

model_2.summary()
```

– Output: Hình 4



| Layer (type) | Output Shape | Param # |
|---------------------------|--------------|---------|
| dense_2 (Dense) | (None, 784) | 615440 |
| dense_3 (Dense) | (None, 10) | 7850 |
| Total params: 623,290 | | |
| Trainable params: 623,290 | | |
| Non-trainable params: 0 | | |

Hình 3: Summary model

* **Nhận xét:**

- Model có 1 hidden layer Dense, có shape là (None, 784) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model (tổng số điểm dữ liệu training), 784 là shape dữ liệu đầu vào của model.
- Tổng số params mà model cần phải học là 623,290 params.

2. Huấn luyện mô hình

```
# Compile model
Optimizer = SGD(learning_rate=0.01)
Loss = BinaryCrossentropy()
model_2.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])
```

```
# Training model
history_2 = model_2.fit(X_train_resaped, y_train_new,
                        validation_data=(X_dev_resaped, y_dev_new),
                        batch_size=128, epochs=30)
```

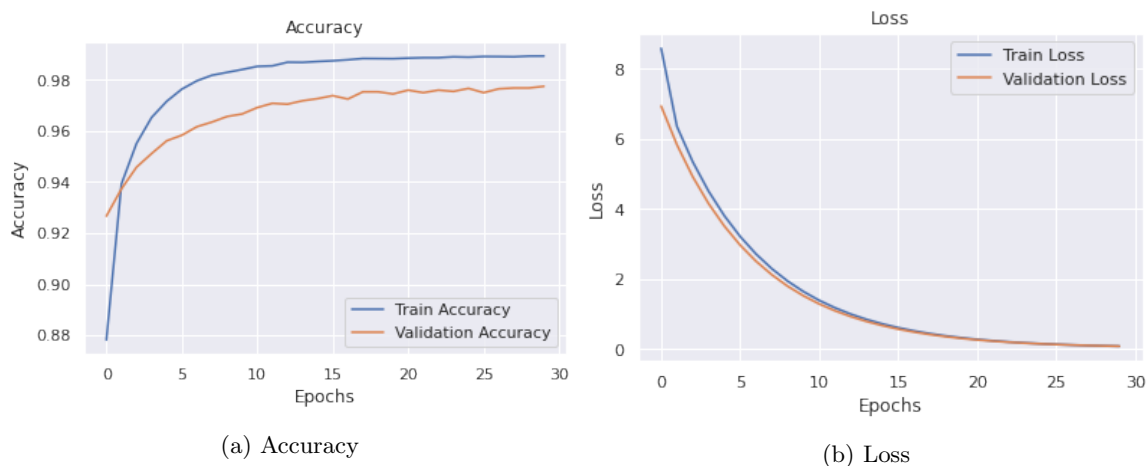
3. Visualization Loss & Accuracy training

```
plt.figure(0)
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.figure(0)
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



Hình 4: Accuracy và Loss của model bài 2

* Nhận xét:

- Accuracy trên tập train bắt đầu hội tụ ở epoch thứ 10, accuracy trên tập dev bắt đầu hội tụ ở epoch thứ 20. Accuracy trên tập train cao ($\sim 98.9\%$), accuracy trên tập dev cao ($\sim 98.0\%$), chênh lệch khoảng 1.0%. Khoảng cách chênh lệch đã giảm đi đáng kể, hay nói cách khác là đã khắc phục được hiện tượng Overfitting ở bài 1 bằng kỹ thuật regularization.
- Loss giảm nhanh và đều qua các epoch, không có hiện tượng giảm đột ngột như bài 1. Giá trị loss của train và dev gần như bằng nhau qua các epoch. Loss của train và dev hội tụ cùng nhau (khoảng epoch thứ 25).

4. Đánh giá mô hình

```
y_pred = model_2.predict(X_test_resaped)
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 = 97.98%
```

* **Nhận xét:**

- Accuracy của mô hình khi kiểm tra trên tập test cao (Accuracy test = 97.98%), cao hơn đáng kể so với accuracy test của mô hình ở bài 1.

Bài 3: Thực hiện huấn luyện mô hình mạng neural ở bài 1 với kỹ thuật khởi tạo tham số cho tham số W. Nếu sử dụng kỹ thuật khởi tạo tham số là 1 cho tham số W thì kết quả độ chính xác mô hình như thế nào?

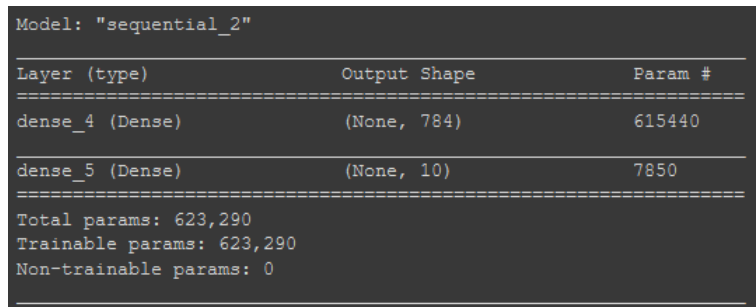
Câu a: Khởi tạo với: kernel_initializer = Zeros, bias_initializer = Zeros

1. Xây dựng mạng neural bằng Keras

```
# Khởi tạo model
model_3a = Sequential()
# Áp dụng kỹ thuật khởi tạo tham số cho W, với W = 0
model_3a.add(Dense(784, input_shape=(784, ),
                    kernel_initializer = Zeros,
                    bias_initializer= Zeros,
                    activation='relu'))
model_3a.add(Dense(10, input_shape=(10, ),
                    activation='sigmoid'))

model_3a.summary()
```

– Output: Hình 5



The image shows a terminal window displaying the summary of a Keras model named 'sequential_2'. The summary is as follows:

| Layer (type) | Output Shape | Param # |
|---------------------------|--------------|---------|
| dense_4 (Dense) | (None, 784) | 615440 |
| dense_5 (Dense) | (None, 10) | 7850 |
| Total params: 623,290 | | |
| Trainable params: 623,290 | | |
| Non-trainable params: 0 | | |

Hình 5: Summary model

* **Nhận xét:**

- Model có 1 hidden layer Dense, có shape là (None, 784) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model (tổng số điểm dữ liệu training), 784 là shape dữ liệu đầu vào của model.
- Tổng số params mà model cần phải học là 623,290 params.

2. Huấn luyện mô hình

```
# Compile model
Optimizer = SGD(learning_rate=0.01)
Loss = BinaryCrossentropy()
model_3a.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_3a = model_3a.fit(X_train_reshaped, y_train_new,
                          validation_data=(X_dev_reshaped, y_dev_new),
                          batch_size=128, epochs=30)
```

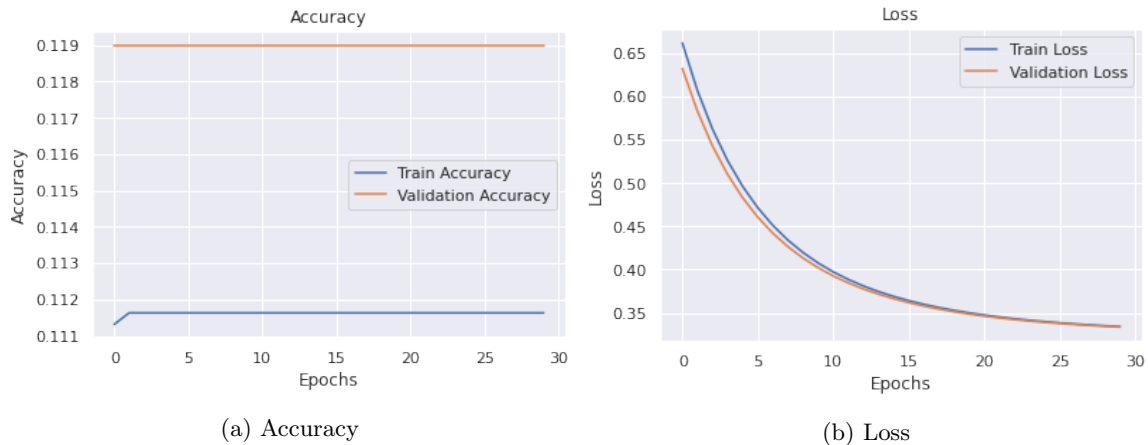
3. Visualization Loss & Accuracy training

```
plt.figure(0)
plt.plot(history_3a.history['accuracy'], label = 'Train Accuracy')
plt.plot(history_3a.history['val_accuracy'], label = 'Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

– Output: Hình 6a

```
plt.figure(0)
plt.plot(history_3a.history['loss'], label = 'Train Loss')
plt.plot(history_3a.history['val_loss'], label = 'Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 6b



Hình 6: Accuracy và Loss của model bài 3a

* Nhận xét:

- Accuracy trên tập dev không thay đổi từ epoch 1 đến epoch 30, accuracy trên tập train thì có thay đổi ở epoch đầu nhưng nhau đó lại không thay đổi. Có thể thấy model bị Underfitting nghiêm trọng, tức là không học được gì từ dữ liệu.

4. Đánh giá mô hình

```
y_pred = model_3a.predict(X_test_reshaped)
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 = 11.35%
```

* Nhận xét:

- Accuracy của mô hình khi kiểm tra trên tập test rất thấp (Accuracy test = 11.35%), mô hình bị Underfitting.

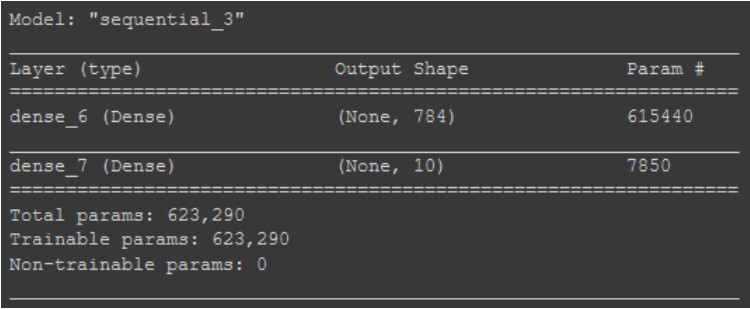
Câu b: Khởi tạo với: `kernel_initializer = Ones`, `bias_initializer = Ones`

1. Xây dựng mạng neural bằng Keras

```
# Khởi tạo model
model_3b = Sequential()
# Áp dụng kỹ thuật khởi tạo tham số cho W, với W = 1
model_3b.add(Dense(784, input_shape=(784, ),
                    kernel_initializer = Ones,
                    bias_initializer= Ones,
                    activation='relu'))
model_3b.add(Dense(10, input_shape=(10, ),
                    activation='sigmoid'))

model_3b.summary()
```

– Output: Hình 7



The image shows the output of the `model_3b.summary()` command in a terminal window. It displays the architecture of the model 'sequential_3', which consists of two dense layers: 'dense_6' with 784 units and 'dense_7' with 10 units. The total number of parameters is 623,290, all of which are trainable.

| Layer (type) | Output Shape | Param # |
|---------------------------|--------------|---------|
| dense_6 (Dense) | (None, 784) | 615440 |
| dense_7 (Dense) | (None, 10) | 7850 |
| Total params: 623,290 | | |
| Trainable params: 623,290 | | |
| Non-trainable params: 0 | | |

Hình 7: Summary model

* Nhận xét:

- Model có 1 hidden layer Dense, có shape là (None, 784) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model (tổng số điểm dữ liệu training), 784 là shape dữ liệu đầu vào của model.
- Tổng số params mà model cần phải học là 623,290 params.

2. Huấn luyện mô hình

```
# Compile model
Optimizer = SGD(learning_rate=0.01)
Loss = BinaryCrossentropy()
model_3b.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_3b = model_3b.fit(X_train_reshaped, y_train_new,
                          validation_data=(X_dev_reshaped, y_dev_new),
                          batch_size=128, epochs=30)
```

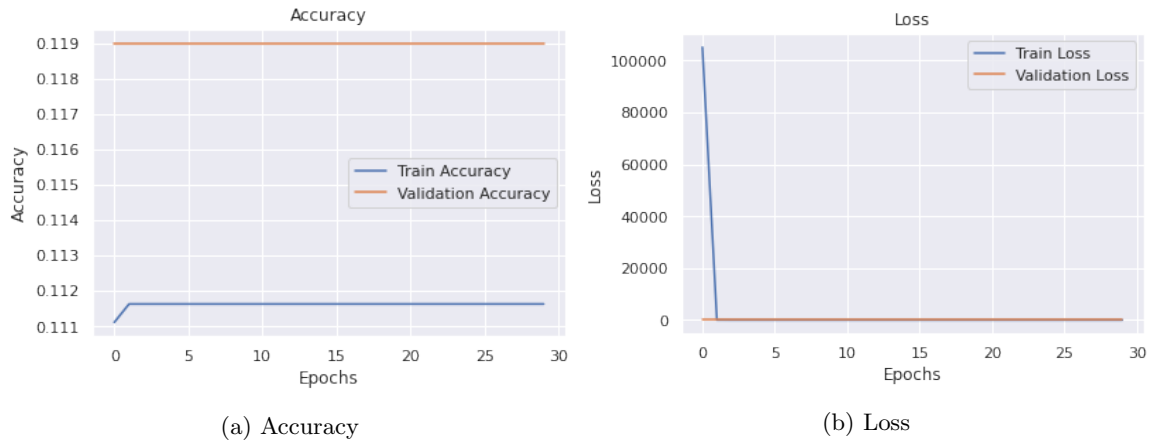
3. Visualization Loss & Accuracy training

```
plt.figure(0)
plt.plot(history_3b.history['accuracy'], label = 'Train Accuracy')
plt.plot(history_3b.history['val_accuracy'], label = 'Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

– Output: Hình 8a

```
plt.figure(0)
plt.plot(history_3b.history['loss'], label = 'Train Loss')
plt.plot(history_3b.history['val_loss'], label = 'Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 8b



Hình 8: Accuracy và Loss của model bài 3b

* **Nhận xét:**

- Accuracy trên tập dev không thay đổi từ epoch 1 đến epoch 30, accuracy trên tập train thì có thay đổi ở epoch đầu nhưng nhau đó lại không thay đổi. Có thể thấy model bị Underfitting nghiêm trọng, tức là không học được gì từ dữ liệu.

4. Đánh giá mô hình

```
y_pred = model_3a.predict(X_test_resaped)
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 = 11.35%
```

* **Nhận xét:**

- Accuracy của mô hình khi kiểm tra trên tập test rất thấp (Accuracy test = 11.35%), mô hình bị Underfitting.

Câu c: Khởi tạo với: kernel_initializer = GlorotUniform, bias_initializer = Ones

1. Xây dựng mạng neural bằng Keras

```
# Khởi tạo model
model_3c = Sequential()
# Áp dụng kỹ thuật khởi tạo tham số cho W, với W = 1
model_3c.add(Dense(784, input_shape=(784, ),
                    kernel_initializer = GlorotUniform,
                    bias_initializer= Ones,
                    activation='relu'))
model_3c.add(Dense(10, input_shape=(10, ),
                    activation='sigmoid'))

model_3c.summary()
```

– Output: Hình 9

```
Model: "sequential_4"
```

| Layer (type) | Output Shape | Param # |
|-----------------|--------------|---------|
| dense_8 (Dense) | (None, 784) | 615440 |
| dense_9 (Dense) | (None, 10) | 7850 |

```
Total params: 623,290  
Trainable params: 623,290  
Non-trainable params: 0
```

Hình 9: Summary model

* **Nhận xét:**

- Model có 1 hidden layer Dense, có shape là (None, 784) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model (tổng số điểm dữ liệu training), 784 là shape dữ liệu đầu vào của model.
- Tổng số params mà model cần phải học là 623,290 params.

2. Huấn luyện mô hình

```
# Compile model  
Optimizer = SGD(learning_rate=0.01)  
Loss = BinaryCrossentropy()  
model_3c.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])  
  
# Training model  
history_3c = model_3c.fit(X_train_resaped, y_train_new,  
                           validation_data=(X_dev_resaped, y_dev_new),  
                           batch_size=128, epochs=30)
```

3. Visualization Loss & Accuracy training

```
plt.figure(0)  
plt.plot(history_3c.history['accuracy'], label = 'Train Accuracy')  
plt.plot(history_3c.history['val_accuracy'], label = 'Validation Accuracy')  
plt.title('Accuracy')  
plt.xlabel('Epochs')  
plt.ylabel('Accuracy')  
plt.legend()  
plt.show()
```

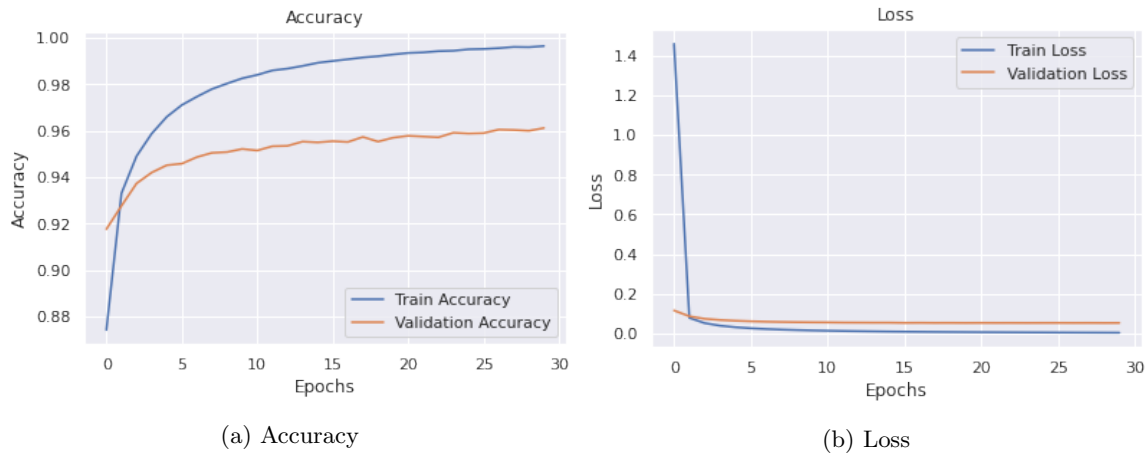
– Output: Hình 10a

```
plt.figure(0)  
plt.plot(history_3c.history['loss'], label = 'Train Loss')  
plt.plot(history_3c.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 vẫn tiếp tục tăng đến epoch thứ 25 mới bắt đầu hội tụ, nhưng accuracy trên tập dev đã bắt đầu hội tụ ở epoch thứ 15. Accuracy trên tập train rất cao ($\sim 99.6\%$), accuracy trên tập dev cao ($\sim 96.0\%$), chênh lệch khoảng 4.0%. Nên có thể đã xảy ra hiện tượng Overfitting (kết quả của mô hình này giống với kết quả của mô hình ở bài 1).



Hình 10: Accuracy và Loss của model bài 3c

4. Đánh giá mô hình

```

y_pred = model_3c.predict(X_test_resaped)
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 = 96.21%

```

* Nhận xét:

- Accuracy của mô hình khi kiểm tra trên tập test cao (Accuracy test = 96.21%).

Kết luận

- Khi khởi tạo `kernel_initializer=Zeros`, `bias_initializer=Zeros` hoặc `kernel_initializer=Ones`, `bias_initializer=Ones` thì mô hình thu được sẽ có hiện tượng bị underfitting. Độ chính xác của mô hình khi đánh giá trên tập test rất thấp (11.35%), nên mô hình rất tệ.
- Khi khởi tạo `kernel_initializer=GlorotUniform`, `bias_initializer=Ones` thì mô hình thu được sẽ có hiện tượng bị overfitting (giống với bài 1). Độ chính xác của mô hình khi đánh giá trên tập test rất cao (96.21%).
- Vậy, bước khởi tạo tham số cho mô hình có ảnh hưởng lớn đến kết quả của mô hình thu được, khởi tạo hợp lý thì sẽ thu được mô hình có kết quả cao, còn khởi tạo với 0 hoặc 1 thì thu được mô hình có kết quả tệ như bài trên.

Bài 4: Thực hiện huấn luyện mô hình mạng neural ở bài 2 (đã áp dụng regularization) với thuật toán optimizer RMSProp và Adam. So sánh kết quả giữa 2 thuật toán.

Câu a: Xây dựng model với optimizer=SGD, momentum=0.9

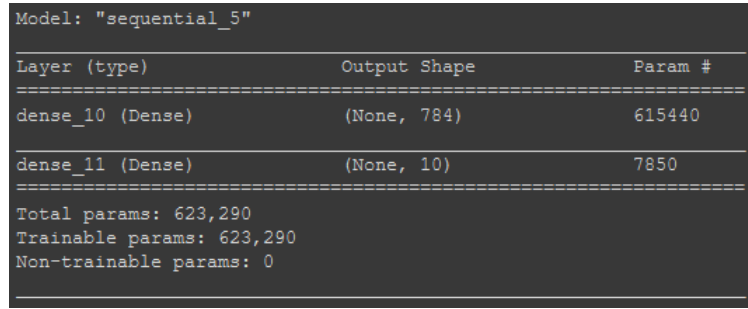
1. Xây dựng mạng neural bằng Keras

```

# Khởi tạo model
model_4a = Sequential()
# Áp dụng kỹ thuật regularization với lambda = 0.01
model_4a.add(Dense(784, input_shape=(784, ),
                    kernel_regularizer = l2 (0.01),
                    bias_regularizer = l2 (0.01),
                    activation='relu'))
model_4a.add(Dense(10, input_shape=(10, ),
                    kernel_regularizer = l2 (0.01),
                    bias_regularizer = l2 (0.01),
                    activation='sigmoid'))
model_4a.summary()

```

– Output: Hình 11



| Layer (type) | Output Shape | Param # |
|---------------------------|--------------|---------|
| dense_10 (Dense) | (None, 784) | 615440 |
| dense_11 (Dense) | (None, 10) | 7850 |
| Total params: 623,290 | | |
| Trainable params: 623,290 | | |
| Non-trainable params: 0 | | |

Hình 11: Summary model

* **Nhận xét:**

- Model có 1 hidden layer Dense, có shape là (None, 784) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model (tổng số điểm dữ liệu training), 784 là shape dữ liệu đầu vào của model.
- Tổng số params mà model cần phải học là 623,290 params.

2. Huấn luyện mô hình

```
# Compile model
Optimizer = SGD(learning_rate=0.01, momentum=0.9)
Loss = BinaryCrossentropy()
model_4a.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_4a = model_4a.fit(X_train_reshaped, y_train_new,
                          validation_data=(X_dev_reshaped, y_dev_new),
                          batch_size=128, epochs=30)
```

3. Visualization Loss & Accuracy training

```
plt.figure(0)
plt.plot(history_4a.history['accuracy'], label = 'Train Accuracy')
plt.plot(history_4a.history['val_accuracy'], label = 'Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

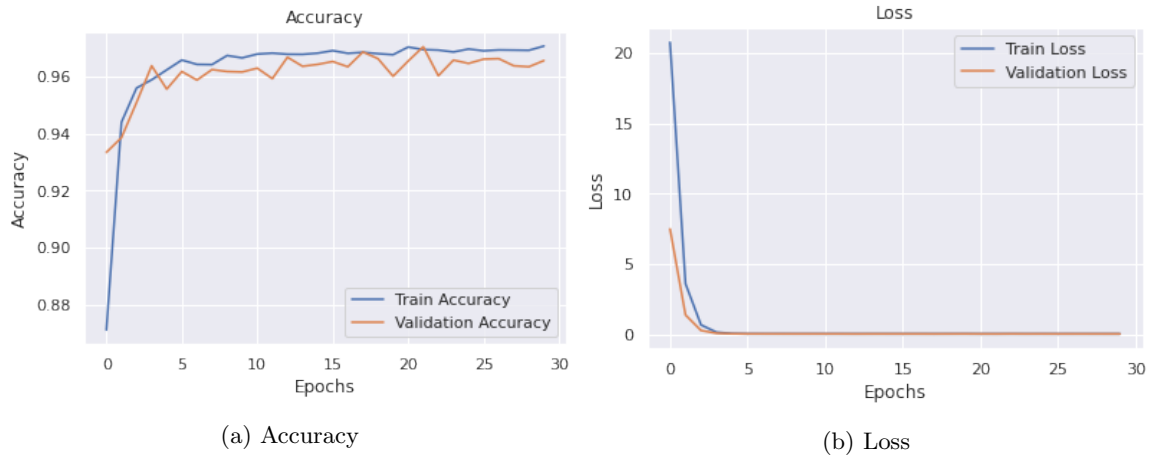
– Output: Hình 12a

```
plt.figure(0)
plt.plot(history_4a.history['loss'], label = 'Train Loss')
plt.plot(history_4a.history['val_loss'], label = 'Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 12b

* **Nhận xét:**

- Accuracy trên tập dev và tập train của mô hình rất cao. Mô hình không bị underfitting hay overfitting, nên mô hình này rất tốt.
- Loss giảm rất nhanh và hội tụ rất sớm (khoảng epoch thứ 3).



Hình 12: Accuracy và Loss của model bài 4a

4. Đánh giá mô hình

```

y_pred = model_4a.predict(X_test_resaped)
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 = 96.93%

```

* Nhận xét:

- Accuracy của mô hình khi kiểm tra trên tập test rất cao (Accuracy test = 96.93%), mô hình rất tốt.

Câu b: Xây dựng model với optimizer=RMSPProp

1. Xây dựng mạng neural bằng Keras

```

# Khởi tạo model
model_4b = Sequential()

model_4b.add(Dense(784, input_shape=(784, ),
                        activation='relu'))
model_4b.add(Dense(10, input_shape=(10, ),
                        activation='sigmoid'))

model_4b.summary()

```

– Output: Hình 13

```

Model: "sequential_6"
_____
Layer (type)                 Output Shape              Param #
_____
dense_12 (Dense)             (None, 784)               615440
_____
dense_13 (Dense)             (None, 10)                7850
_____
Total params: 623,290
Trainable params: 623,290
Non-trainable params: 0

```

Hình 13: Summary model

* **Nhận xét:**

- Model có 1 hidden layer Dense, có shape là (None, 784) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model (tổng số điểm dữ liệu training), 784 là shape dữ liệu đầu vào của model.
- Tổng số params mà model cần phải học là 623,290 params.

2. Huấn luyện mô hình

```
# Compile model
Optimizer = RMSprop(learning_rate=1e-4)
Loss = BinaryCrossentropy()
model_4b.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_4b = model_4b.fit(X_train_resaped, y_train_new,
                           validation_data=(X_dev_resaped, y_dev_new),
                           batch_size=128, epochs=30)
```

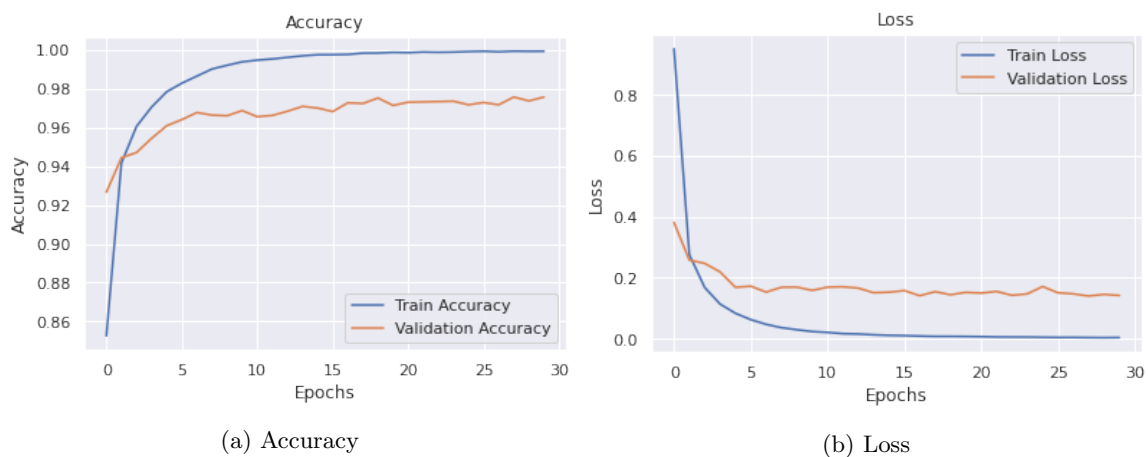
3. Visualization Loss & Accuracy training

```
plt.figure(0)
plt.plot(history_4b.history['accuracy'], label='Train Accuracy')
plt.plot(history_4b.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

– Output: Hình 14a

```
plt.figure(0)
plt.plot(history_4b.history['loss'], label='Train Loss')
plt.plot(history_4b.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 14b



Hình 14: Accuracy và Loss của model bài 4b

* **Nhận xét:**

- Accuracy trên tập dev và tập train của mô hình rất cao. Mô hình có thể bị overfitting nhẹ, nhưng không ảnh hưởng nhiều đến kết quả nên mô hình này vẫn tốt.

- Loss giảm rất nhanh và hội tụ rất sớm (khoảng epoch thứ 5 với dev và epoch thứ 10 với train).

4. Đánh giá mô hình

```
y_pred = model_4b.predict(X_test_reshaped)
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 = 97.54%
```

*** Nhận xét:**

- Accuracy của mô hình khi kiểm tra trên tập test rất thấp (Accuracy test = 97.54%), mô hình tốt.

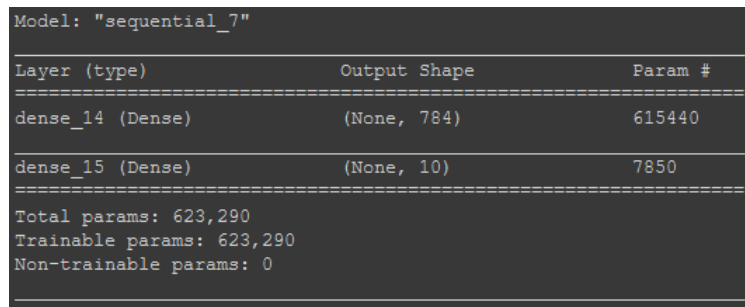
Câu c: Xây dựng model với optimizer=Adam

1. Xây dựng mạng neural bằng Keras

```
# Khởi tạo model
model_4c = Sequential()
# Áp dụng kỹ thuật regularization với lambda = 0.01
model_4c.add(Dense(784, input_shape=(784, ),
                    kernel_regularizer = l2 (0.01),
                    bias_regularizer = l2 (0.01),
                    activation='relu'))
model_4c.add(Dense(10, input_shape=(10, ),
                    kernel_regularizer = l2 (0.01),
                    bias_regularizer = l2 (0.01),
                    activation='sigmoid'))

model_4c.summary()
```

– Output: Hình 15



```
Model: "sequential_7"
Layer (type)                 Output Shape                 Param #
=====
dense_14 (Dense)              (None, 784)                  615440
dense_15 (Dense)              (None, 10)                   7850
=====
Total params: 623,290
Trainable params: 623,290
Non-trainable params: 0
```

Hình 15: Summary model

*** Nhận xét:**

- Model có 1 hidden layer Dense, có shape là (None, 784) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model (tổng số điểm dữ liệu training), 784 là shape dữ liệu đầu vào của model.
- Tổng số params mà model cần phải học là 623,290 params.

2. Huấn luyện mô hình

```
# Compile model
Optimizer = Adam(learning_rate=1e-4)
Loss = BinaryCrossentropy()
model_4c.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])
```

```
# Training model
history_4c = model_4c.fit(X_train_resaped, y_train_new,
                           validation_data=(X_dev_resaped, y_dev_new),
                           batch_size=128, epochs=30)
```

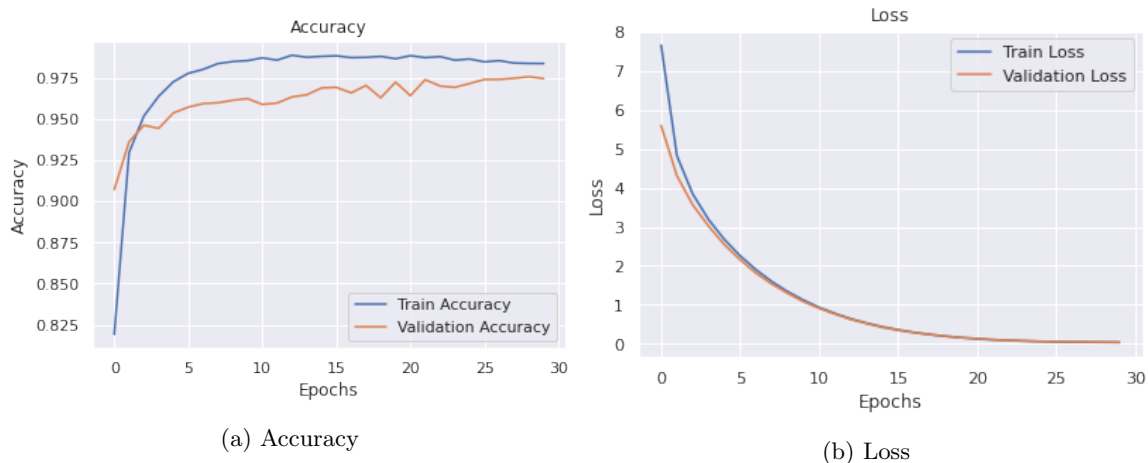
3. Visualization Loss & Accuracy training

```
plt.figure(0)
plt.plot(history_4c.history['accuracy'], label = 'Train Accuracy')
plt.plot(history_4c.history['val_accuracy'], label = 'Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

– Output: Hình 16a

```
plt.figure(0)
plt.plot(history_4c.history['loss'], label = 'Train Loss')
plt.plot(history_4c.history['val_loss'], label = 'Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 16b



Hình 16: Accuracy và Loss của model bài 4c

* Nhận xét:

- Accuracy trên tập dev và tập train của mô hình rất cao. Mô hình không bị underfitting hay overfitting, nên mô hình này rất tốt.
- Loss giảm nhanh, đều và hội tụ khoảng epoch thứ 20.

4. Đánh giá mô hình

```
y_pred = model_4c.predict(X_test_resaped)
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 = 97.57%
```

* Nhận xét:

- Accuracy của mô hình khi kiểm tra trên tập test cao (Accuracy test = 97.57%).

Kết luận

- Mô hình với hàm optimizer=Adam và mô hình với hàm optimizer=RMSProp có Accuracy test khá tương đương nhau. Nhưng nhìn chung mô hình với hàm optimizer=Adam tốt hơn mô hình với hàm optimizer=RMSProp vì mô hình với hàm optimizer=RMSProp có hơi bị overfitting nhẹ, còn mô hình với hàm optimizer=Adam không bị underfitting hay overfitting.

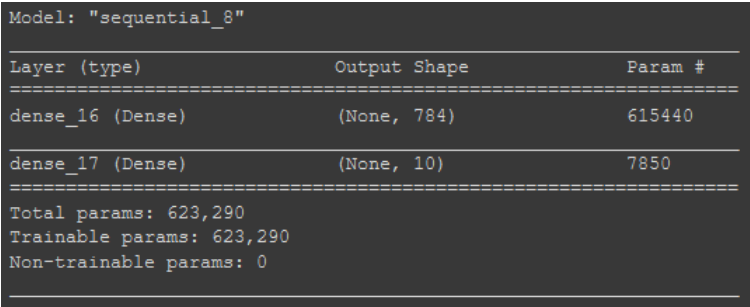
Bài 5: Hãy giảm batch size xuống 8 và huấn luyện mô hình mạng neural ở bài 2 (sử dụng optimizer SGD). Khi batch_size nhỏ thì có chuyện gì xảy ra (dựa vào đồ thị học).

- Xây dựng mạng neural bằng Keras

```
# Khởi tạo model
model_5 = Sequential()
# Áp dụng kỹ thuật regularization với lambda = 0.01
model_5.add(Dense(784, input_shape=(784, ),
                  kernel_regularizer = l2 (0.01),
                  bias_regularizer = l2 (0.01),
                  activation='relu'))
model_5.add(Dense(10, input_shape=(10, ),
                  kernel_regularizer = l2 (0.01),
                  bias_regularizer = l2 (0.01),
                  activation='sigmoid'))

model_5.summary()
```

– Output: Hình 17



| Layer (type) | Output Shape | Param # |
|---------------------------|--------------|---------|
| dense_16 (Dense) | (None, 784) | 615440 |
| dense_17 (Dense) | (None, 10) | 7850 |
| Total params: 623,290 | | |
| Trainable params: 623,290 | | |
| Non-trainable params: 0 | | |

Hình 17: Summary model

* Nhận xét:

- Model có 1 hidden layer Dense, có shape là (None, 784) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model (tổng số điểm dữ liệu training), 784 là shape dữ liệu đầu vào của model.
- Tổng số params mà model cần phải học là 623,290 params.

- Huấn luyện mô hình

```
# Compile model
Optimizer = SGD(learning_rate=0.01)
Loss = BinaryCrossentropy()
model_5.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_5 = model_5.fit(X_train_reshaped, y_train_new,
                        validation_data=(X_dev_reshaped, y_dev_new),
                        batch_size=8, epochs=30)
```

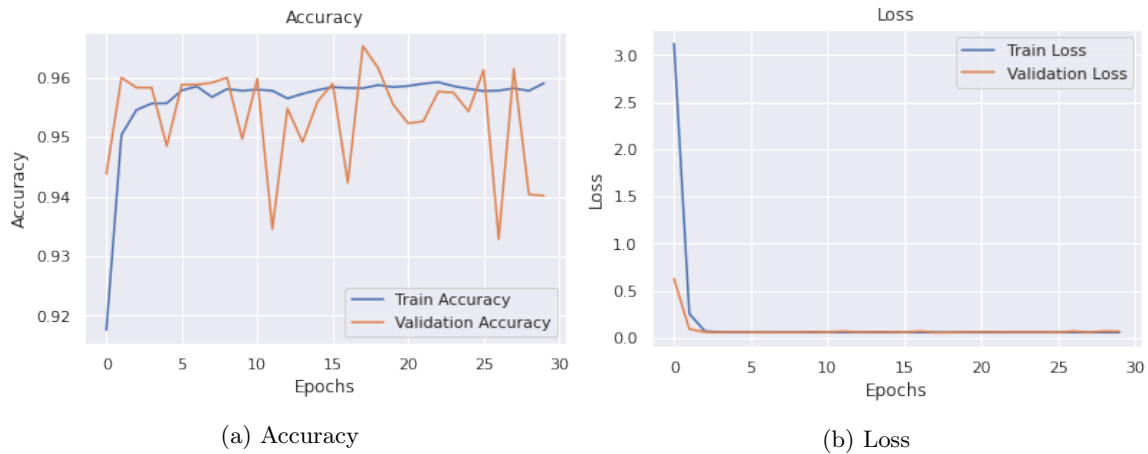
3. Visualization Loss & Accuracy training

```
plt.figure(0)
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 18a

```
plt.figure(0)
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 18b



Hình 18: Accuracy và Loss của model bài 2

* Nhận xét:

- Accuracy trên tập train tăng và có xu hướng hội tụ qua các epoch, còn accuracy trên tập dev lại dao động mạnh (lúc tăng lúc giảm) và chưa có xu hướng hội tụ.
- Loss giảm nhanh và hội tụ ngay những epoch đầu tiên (khoảng epoch thứ 3).
- Thời gian training model sẽ tăng lên khi giảm batch size.

4. Đánh giá mô hình

```
y_pred = model_5.predict(X_test_resaped)
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 = 94.59%
```

* Nhận xét:

- Accuracy của mô hình khi kiểm tra trên tập test cao (Accuracy test = 94.59%), thấp hơn đáng kể so với accuracy test của mô hình ở bài 2.

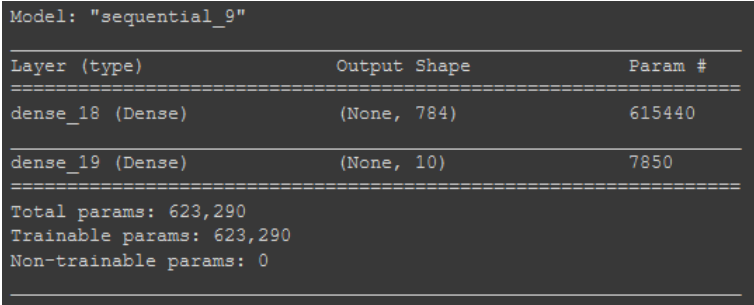
Bài 6: Hãy tăng epoch lên 100 và huấn luyện mô hình ở bài 5 với kỹ thuật Early Stopping (sử dụng monitor là val_loss). Cho biết kết quả.

1. Xây dựng mạng neural bằng Keras

```
# Khởi tạo model
model_6 = Sequential()
# Áp dụng kỹ thuật regularization với lambda = 0.01
model_6.add(Dense(784, input_shape=(784, ),
                  kernel_regularizer = l2 (0.01),
                  bias_regularizer = l2 (0.01),
                  activation='relu'))
model_6.add(Dense(10, input_shape=(10, ),
                  kernel_regularizer = l2 (0.01),
                  bias_regularizer = l2 (0.01),
                  activation='sigmoid'))

model_6.summary()
```

– Output: Hình 19



```
Model: "sequential_9"
Layer (type)                 Output Shape              Param #
-----
dense_18 (Dense)             (None, 784)              615440
dense_19 (Dense)             (None, 10)              7850
-----
Total params: 623,290
Trainable params: 623,290
Non-trainable params: 0
```

Hình 19: Summary model

* Nhận xét:

- Model có 1 hidden layer Dense, có shape là (None, 784) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model (tổng số điểm dữ liệu training), 784 là shape dữ liệu đầu vào của model.
- Tổng số params mà model cần phải học là 623,290 params.

2. Huấn luyện mô hình

```
# Compile model
Optimizer = SGD(learning_rate=0.01)
Loss = BinaryCrossentropy()
callback = EarlyStopping(monitor='val_loss', patience=3)
model_6.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_6 = model_6.fit(X_train_resaped, y_train_new,
                        validation_data=(X_dev_resaped, y_dev_new),
                        batch_size=8, epochs=100,
                        callbacks=[callback])
```

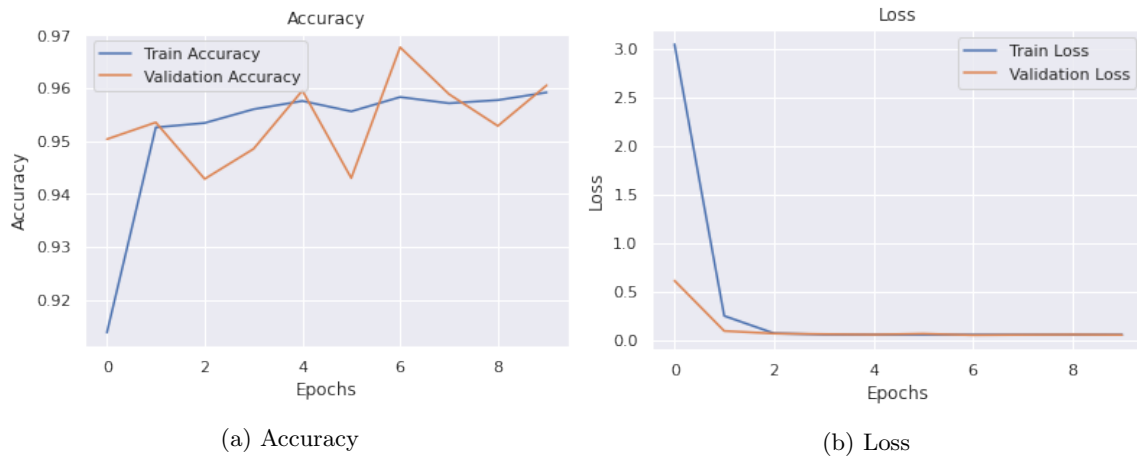
3. Visualization Loss & Accuracy training

```
plt.figure(0)
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 20a

```
plt.figure(0)
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 20b



Hình 20: Accuracy và Loss của model bài 2

* **Nhận xét:**

- Accuracy trên tập train tăng và có xu hướng hội tụ qua các epoch, còn accuracy trên tập dev lại dao động mạnh (lúc tăng lúc giảm) và chưa có xu hướng hội tụ.
- Loss giảm nhanh và hội tụ ngay những epoch đầu tiên (khoảng epoch thứ 3).
- Thời gian training model sẽ tăng lên khi giảm batch size.
- Mô hình đã dừng sớm ở epoch thứ 10 thay vì chạy hết cả 100 epoch. Nguyên nhân là vì val_loss đã tăng liên tiếp 3 lần nên mô hình bị dừng sớm để tránh hiện tượng bị underfitting.

4. Đánh giá mô hình

```
y_pred = model_6.predict(X_test_reshaped)
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 = 96.48%
```

* **Nhận xét:**

- Accuracy của mô hình khi kiểm tra trên tập test cao (Accuracy test = 96.48%), cao hơn đáng kể so với accuracy test của mô hình ở bài 5.

Bài 7:* Thực hiện các yêu cầu từ Bài 1, Bài 2, Bài 6 với bộ dữ liệu CIFAR10

Set up

1. Import các thư viện cần thiết
 - Các thư viện cần thiết sử dụng lại từ phần [Set up](#).

2. Load bộ dữ liệu MNIST

```
from keras.datasets.cifar10 import load_data

(X_train_bai7, y_train_bai7), (X_test_bai7, y_test_bai7) = load_data()
```

3. Chuẩn bị dữ liệu

```
X_train_bai7, X_dev_bai7, y_train_bai7, y_dev_bai7 = train_test_split(X_train_bai7,
                                                                      y_train_bai7,
                                                                      test_size=0.1)
```

```
print('X_train_bai7 = {}'.format(X_train_bai7.shape))
# Output: X_train_bai7 = (45000, 32, 32, 3)
print('y_train_bai7 = {}'.format(y_train_bai7.shape))
# Output: y_train_bai7 = (45000, 1)
print('X_dev_bai7 = {}'.format(X_dev_bai7.shape))
# Output: X_dev_bai7 = (5000, 32, 32, 3)
print('y_dev_bai7 = {}'.format(y_dev_bai7.shape))
# Output: y_dev_bai7 = (5000, 1)
print('X_test_bai7 = {}'.format(X_test_bai7.shape))
# Output: X_test_bai7 = (10000, 32, 32, 3)
print('y_test_bai7 = {}'.format(y_test_bai7.shape))
# Output: y_test_bai7 = (10000, 1)
```

– X_train_5 gồm có 50,000 bức ảnh màu (số channel = 3) với kích thước mỗi bức ảnh màu là 32×32 pixel dùng để train model, y_train_5 là các nhãn tương ứng X_train_5.

– X_test_5 gồm có 10,000 bức ảnh màu (số channel = 3) với kích thước mỗi bức ảnh màu là 32×32 pixel dùng để test model, y_test_5 là các nhãn tương ứng của X_test_5.

```
# Reshape kích thước của dữ liệu
X_train_bai7_resaped = X_train_bai7.reshape(-1, 32*32*3)
X_dev_bai7_resaped = X_dev_bai7.reshape(-1, 32*32*3)
X_test_bai7_resaped = X_test_bai7.reshape(-1, 32*32*3)

y_train_bai7_new = to_categorical(y_train_bai7, num_classes = 10)
y_dev_bai7_new = to_categorical(y_dev_bai7, num_classes = 10)

# Xem kích thước của dữ liệu sau khi reshape
print('X_train_bai7_resaped = {}'.format(X_train_bai7_resaped.shape))
# Output: X_train_bai7_resaped = (45000, 3072)
print('y_train_bai7_new = {}'.format(y_train_bai7_new.shape))
# Output: y_train_bai7_new = (45000, 10)
print('X_dev_bai7_resaped = {}'.format(X_dev_bai7_resaped.shape))
# Output: X_dev_bai7_resaped = (5000, 3072)
print('y_dev_bai7_new = {}'.format(y_dev_bai7_new.shape))
# Output: y_dev_bai7_new = (5000, 10)
print('X_test_bai7_resaped = {}'.format(X_test_bai7_resaped.shape))
# Output: X_test_bai7_resaped = (10000, 3072)
print('y_test_bai7 = {}'.format(y_test_bai7.shape))
# Output: y_test_bai7 = (10000, 1)
```

Câu a: Thực hiện huấn luyện mô hình mạng neural ở Mục 3 trên bộ dữ liệu CIFAR10. Vẽ đồ thị học của mô hình với thông số accuracy và loss

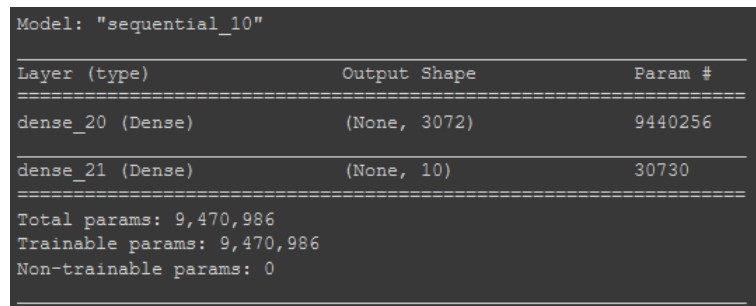
1. Xây dựng mạng neural bằng Keras

```
# Khởi tạo model
model_7a = Sequential()

model_7a.add(Dense(3072, input_shape=(3072, ),
                                activation='relu'))
model_7a.add(Dense(10, input_shape=(10, ),
                    activation='sigmoid'))

model_7a.summary()
```

– Output: Hình 21



```
Model: "sequential_10"
Layer (type)                 Output Shape                 Param #
=====
dense_20 (Dense)             (None, 3072)                9440256
dense_21 (Dense)             (None, 10)                  30730
=====
Total params: 9,470,986
Trainable params: 9,470,986
Non-trainable params: 0
```

Hình 21: Summary model

* **Nhận xét:**

- Model có 1 hidden layer Dense, có shape là (None, 3072) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model, 3072 là shape đầu vào của model.
- Tổng số params mà model cần phải học là 9,470,986 params.

2. Huấn luyện mô hình

```
# Compile model
Optimizer = SGD(learning_rate=1e-3)
Loss = BinaryCrossentropy()
model_7a.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_7a = model_7a.fit(X_train_bai7_resaped, y_train_bai7_new,
                          validation_data=(X_dev_bai7_resaped, y_dev_bai7_new),
                          batch_size=128, epochs=30)
```

3. Visualization Loss & Accuracy training

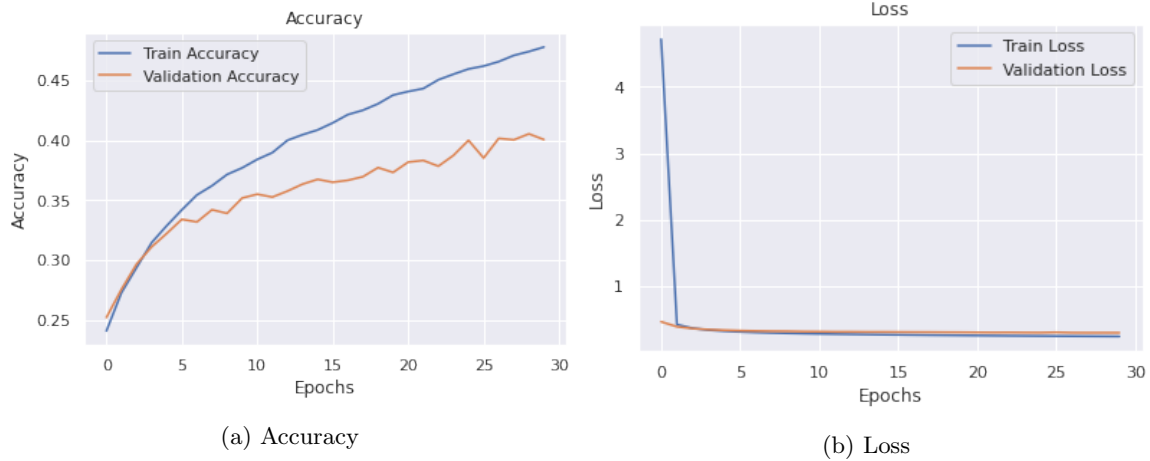
```
plt.figure(0)
plt.plot(history_7a.history['accuracy'], label = 'Train Accuracy')
plt.plot(history_7a.history['val_accuracy'], label = 'Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

– Output: Hình 22a

```
plt.figure(0)
plt.plot(history_7a.history['loss'], label = 'Train Loss')
plt.plot(history_7a.history['val_loss'], label = 'Validation Loss')
```

```
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 22b



Hình 22: Accuracy và Loss của model bài 4a

* **Nhận xét:**

- Accuracy trên tập train và trên tập dev thấp. Mô hình có học được nhưng học chậm, kết quả không được tốt. Accuracy trên tập train $\sim 47.8\%$, accuracy trên tập dev $\sim 40.0\%$, chênh lệch khoảng 7.0%. Nên có thể đã xảy ra hiện tượng Overfitting.
- Loss giảm rất nhanh và hội tụ rất sớm.

4. Đánh giá mô hình

```
y_pred_bai7 = model_7a.predict(X_test_bai7_reshaped)
y_pred_bai7 = np.argmax(y_pred_bai7, axis = -1)
accuracy = round(accuracy_score(y_test_bai7, y_pred_bai7)*100,2)
print('Accuracy test = {}%'.format(accuracy))
# Output: Accuracy test = 38.52%
```

* **Nhận xét:**

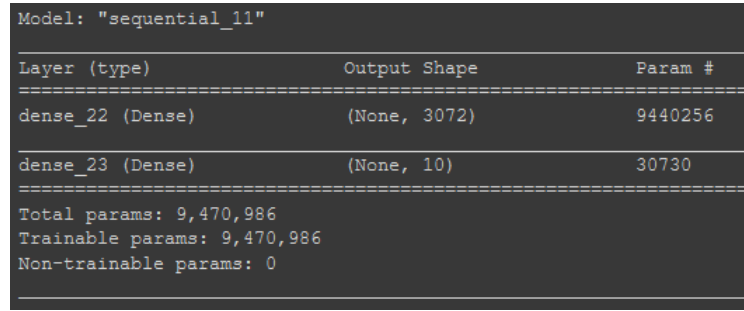
- Accuracy của mô hình khi kiểm tra trên tập test khá thấp (Accuracy test = 38.52%). Mô hình bị overfitting và accuracy trên tập test thấp nên mô hình này là một mô hình tệ.

Câu b: Thực hiện huấn luyện mô hình mạng neural ở bài 1 với kỹ thuật regularization cho tham số W và b với lamda là 0.01. Cho biết kết quả huấn luyện mô hình trước và sau khi áp dụng kỹ thuật regularization (Dựa vào đồ thị học).

1. Xây dựng mạng neural bằng Keras

```
# Khởi tạo model
model_7b = Sequential()
model_7b.add(Dense(3072, input_shape=(3072, ),
                                kernel_regularizer = l2 (0.01),
                                bias_regularizer = l2 (0.01),
                                activation='relu'))
model_7b.add(Dense(10, input_shape=(10, ),
                                kernel_regularizer = l2 (0.01),
                                bias_regularizer = l2 (0.01),
                                activation='sigmoid'))
model_7b.summary()
```

– Output: Hình 23



```
Model: "sequential_11"
Layer (type)                Output Shape              Param #
=====
dense_22 (Dense)            (None, 3072)              9440256
dense_23 (Dense)            (None, 10)                30730
=====
Total params: 9,470,986
Trainable params: 9,470,986
Non-trainable params: 0
```

Hình 23: Summary model

* **Nhận xét:**

- Model có 1 hidden layer Dense, có shape là (None, 3072) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model, 3072 là shape đầu vào của model.
- Tổng số params mà model cần phải học là 9,470,986 params.

2. Huấn luyện mô hình

```
# Compile model
Optimizer = SGD(learning_rate=1e-3)
Loss = BinaryCrossentropy()
model_7b.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_7b = model_7b.fit(X_train_bai7_resaped, y_train_bai7_new,
                          validation_data=(X_dev_bai7_resaped, y_dev_bai7_new),
                          batch_size=128, epochs=30)
```

3. Visualization Loss & Accuracy training

```
plt.figure(0)
plt.plot(history_7b.history['accuracy'], label = 'Train Accuracy')
plt.plot(history_7b.history['val_accuracy'], label = 'Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

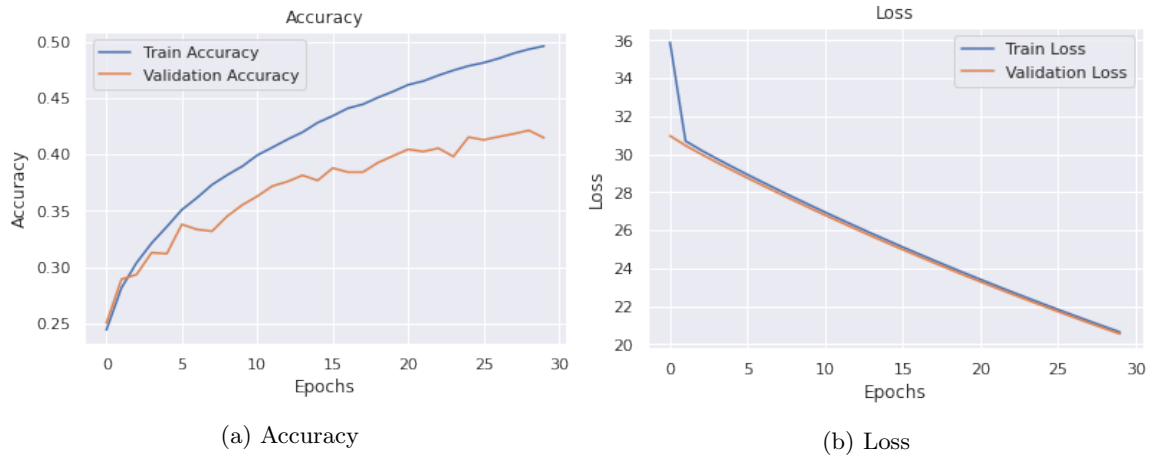
– Output: Hình 24a

```
plt.figure(0)
plt.plot(history_7b.history['loss'], label = 'Train Loss')
plt.plot(history_7b.history['val_loss'], label = 'Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 24b

* **Nhận xét:**

- Accuracy trên tập train và trên tập dev thấp. Mô hình có học được nhưng học chậm, kết quả không được tốt. Accuracy trên tập train $\sim 50.0\%$, accuracy trên tập dev $\sim 42.0\%$, chênh lệch khoảng 8.0%. Khoảng cách chênh lệch vẫn chưa giảm, hay nói cách khác là vẫn chưa khắc phục được hiện tượng Overfitting ở bài 7a bằng kỹ thuật regularization.



Hình 24: Accuracy và Loss của model bài 4b

- Loss giảm nhanh và đều qua các epoch, không có hiện tượng giảm đột ngột như bài 7a. Giá trị loss của train và dev gần như bằng nhau qua các epoch.

4. Đánh giá mô hình

```

y_pred_bai7 = model_7b.predict(X_test_bai7_resaped)
y_pred_bai7 = np.argmax(y_pred_bai7, axis = -1)
accuracy = round(accuracy_score(y_test_bai7, y_pred_bai7)*100,2)
print('Accuracy test = {}'.format(accuracy))
# Output: Accuracy test = 41.34%

```

* Nhận xét:

- Accuracy của mô hình khi kiểm tra trên tập test khá thấp (Accuracy test = 41.34%), cao hơn so với accuracy test của mô hình ở bài 7a. Nhưng mô hình vẫn còn bị overfitting và accuracy trên tập test thấp nên mô hình này là vẫn còn tệ.

Câu c: Hãy tăng epoch lên 100 và huấn luyện mô hình ở bài 5 với kỹ thuật Early Stopping (sử dụng monitor là val_loss). Cho biết kết quả.

1. Xây dựng mạng neural bằng Keras

```

# Khởi tạo model
model_7c = Sequential()
# Áp dụng kỹ thuật regularization với lambda = 0.01
model_7c.add(Dense(3072, input_shape=(3072, ),
                    kernel_regularizer = l2 (0.01),
                    bias_regularizer = l2 (0.01),
                    activation='relu'))
model_7c.add(Dense(10, input_shape=(10, ),
                    kernel_regularizer = l2 (0.01),
                    bias_regularizer = l2 (0.01),
                    activation='sigmoid'))

model_7c.summary()

```

– Output: Hình 25

* Nhận xét:

- Model có 1 hidden layer Dense, có shape là (None, 3072) trong đó : None sẽ được xác định trong quá trình đưa dữ liệu vào training model, 3072 là shape đầu vào của model.
- Tổng số params mà model cần phải học là 9,470,986 params.

| Model: "sequential_12" | | |
|-----------------------------|--------------|---------|
| Layer (type) | Output Shape | Param # |
| dense_24 (Dense) | (None, 3072) | 9440256 |
| dense_25 (Dense) | (None, 10) | 30730 |
| Total params: 9,470,986 | | |
| Trainable params: 9,470,986 | | |
| Non-trainable params: 0 | | |

Hình 25: Summary model

2. Huấn luyện mô hình

```
# Compile model
Optimizer = SGD(learning_rate=1e-3)
Loss = BinaryCrossentropy()
callback = EarlyStopping(monitor='val_loss', patience=2)
model_7c.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])

# Training model
history_7c = model_7c.fit(X_train_bai7_resaped, y_train_bai7_new,
                          validation_data=(X_dev_bai7_resaped, y_dev_bai7_new),
                          batch_size=8, epochs=100,
                          callbacks=[callback])
```

3. Visualization Loss & Accuracy training

```
plt.figure(0)
plt.plot(history_7c.history['accuracy'], label='Train Accuracy')
plt.plot(history_7c.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

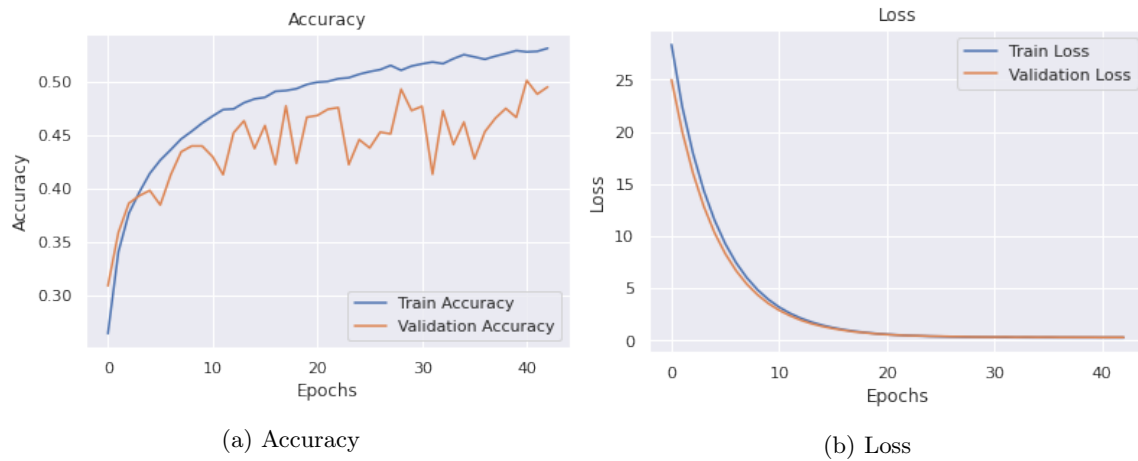
– Output: Hình 26a

```
plt.figure(0)
plt.plot(history_7c.history['loss'], label='Train Loss')
plt.plot(history_7c.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

– Output: Hình 26b

* Nhận xét:

- Accuracy trên tập train và trên tập dev thấp. Mô hình có học được nhưng học chậm, kết quả không được tốt. Accuracy trên tập train $\sim 53.0\%$, accuracy trên tập dev $\sim 50.0\%$, chênh lệch khoảng 3.0%. Khoảng cách chênh lệch đã giảm đi đáng kể, hay nói cách khác là đã phục được hiện tượng overfitting, nhưng vẫn còn overfitting nhẹ.
- Accuracy trên tập train tăng và có xu hướng hội tụ qua các epoch, còn accuracy trên tập dev lại dao động mạnh (lúc tăng lúc giảm) và chưa có xu hướng hội tụ.
- Loss giảm nhanh và đều qua các epoch. Giá trị loss của train và dev gần như bằng nhau qua các epoch. Loss của train và dev hội tụ cùng nhau (khoảng epoch thứ 25)
- Thời gian training model sẽ tăng lên khi giảm batch size.



Hình 26: Accuracy và Loss của model bài 4c

- Mô hình đã dừng sớm ở epoch thứ 43 thay vì chạy hết cả 100 epoch. Nguyên nhân là vì `val_loss` đã tăng liên tiếp 2 lần nên mô hình bị dừng sớm để tránh hiện tượng bị underfitting.

4. Đánh giá mô hình

```

y_pred_bai7 = model_7c.predict(X_test_bai7_resaped)
y_pred_bai7 = np.argmax(y_pred_bai7, axis = -1)
accuracy = round(accuracy_score(y_test_bai7, y_pred_bai7)*100,2)
print('Accuracy test = {}'.format(accuracy))
# Output: Accuracy test = 47.65%

```

* Nhận xét:

- Accuracy của mô hình khi kiểm tra trên tập test khá thấp (Accuracy test = 47.65%), cao hơn so với accuracy test của mô hình ở bài 7a, 7b. Mặc dù mô hình đã giảm đi hiện tượng bị overfitting nhưng accuracy trên tập test thấp nên mô hình này là vẫn còn tệ.

Kết luận

- Trong 3 mô hình bài 7a, 7b, 7c thì mô hình bài 7c là tốt nhất trong 3 mô hình (accuracy = 47.65%), nhưng nhìn chung thì mô hình vẫn chưa tốt.