THỰC HÀNH 7 : LONG SHORT-TERM MEMMORY

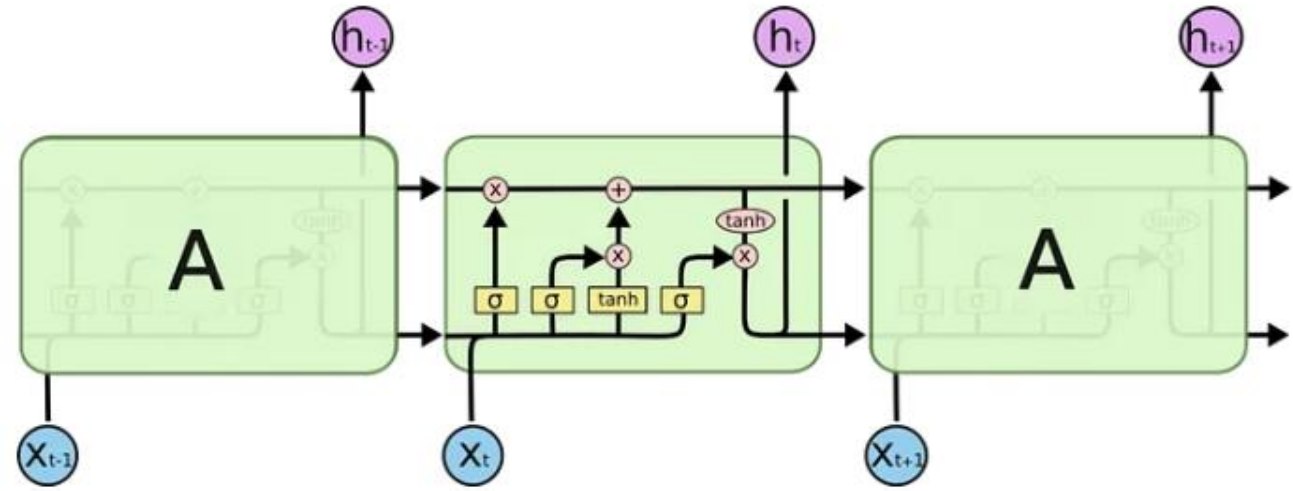
Sau bài thực hành này, sinh viên có khả năng:

- Xây dựng được kiến trúc Long Short-Term Memmory (LSTM)
- Cài đặt mô hình Long Short-Term Memmory bằng tensorflow
- Cài đặt ứng dụng thực tế sử dụng Long Short-Term Memmory

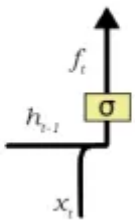
1. GIỚI THIỆU

Recurrent Neural Network (RNN) có thể sử dụng thông tin thời điểm trước để dự báo thông tin tiếp theo. Tuy nhiên, RNN gặp vấn đề exploding và vanishing gradient. Vanishing gradient xảy ra khi giá trị của gradient quá nhỏ làm cho model dừng hoặc huấn luyện quá lâu. Ngoài ra, RNN chưa thể lấy tất cả thông tin trước đó (nhớ lâu) để dự báo. RNN không phân biệt thông tin nào quan trọng và không quan trọng. LSTM giải quyết được các vấn đề trên.

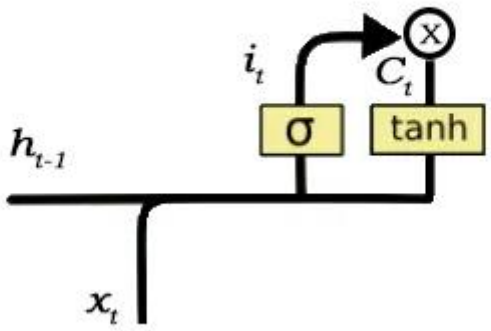
LSTM sử dụng các gate để điều khiển dòng thông tin từ input cho đến output. Các gate này quyết định dữ liệu nào có ích trong tương lai nên giữ lại hoặc cần xóa đi.



Cổng Forget (Forget Gate): cổng này quyết định thông tin nào quan trọng, cần được lưu trữ hoặc thông tin nào cần quên. Điều này làm tối ưu hiệu năng của mô hình. Cổng này nhận 2 input: hidden state từ bước trước và input x của neuron hiện hành. Kèm theo đó là weight, bias và hàm sigmoid được áp dụng cho các giá trị. Nếu đầu ra của cổng này là 0 thì loại bỏ thông tin, còn bằng 1 thì nhớ thông tin này.



Cổng Input (Input Gate): Cổng này dùng để thêm thông tin vào neuron bằng cách sử dụng hàm sigmoid. Cổng này tạo một mảng thông tin và hoàn tất bằng cách kết hợp hàm tanh. Cổng này xác định thông tin nào cần thêm vào.



Cổng Output (Output Gate): Cổng này có nhiệm vụ chọn thông tin quan trọng từ neuron hiện hành và hiển thị output. Cổng này tạo vector giá trị sử dụng hàm tanh. Cổng này sử dụng output của neuron trước và input neuron hiện hành và hàm sigmoid để quyết định thông tin nào sẽ được output.

2. CÀI ĐẶT LSTM

Phần này sẽ hướng dẫn cài đặt LSTM trên bộ dữ liệu MNIST nhận diện chữ viết tay.

2.1. Nạp thư viện

```
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
import keras
from keras import Sequential
from keras.layers import LSTM, BatchNormalization, Dense
```

2.2. Nạp dữ liệu MNIST

```
[10] mnist = keras.datasets.mnist
     (x_train, y_train), (x_test, y_test) = mnist.load_data()
     x_train, y_train = x_train/255.0, y_train/255.0
     x_validate, y_validate = x_test[:-10], y_test[:-10]
     x_test, y_test = x_test[-10:], y_test[-10:]
```

2.3. Định nghĩa mô hình LSTM

```
[32] model = Sequential()
     model.add(LSTM(64, input_shape=(x_train.shape[1:]), return_sequences=True))
     model.add(Dropout(0.2))
     model.add(LSTM(64))
     model.add(Dense(32, activation='relu'))
     model.add(Dropout(0.2))
     model.add(Dense(10, activation='softmax'))
     print(model.summary())
```

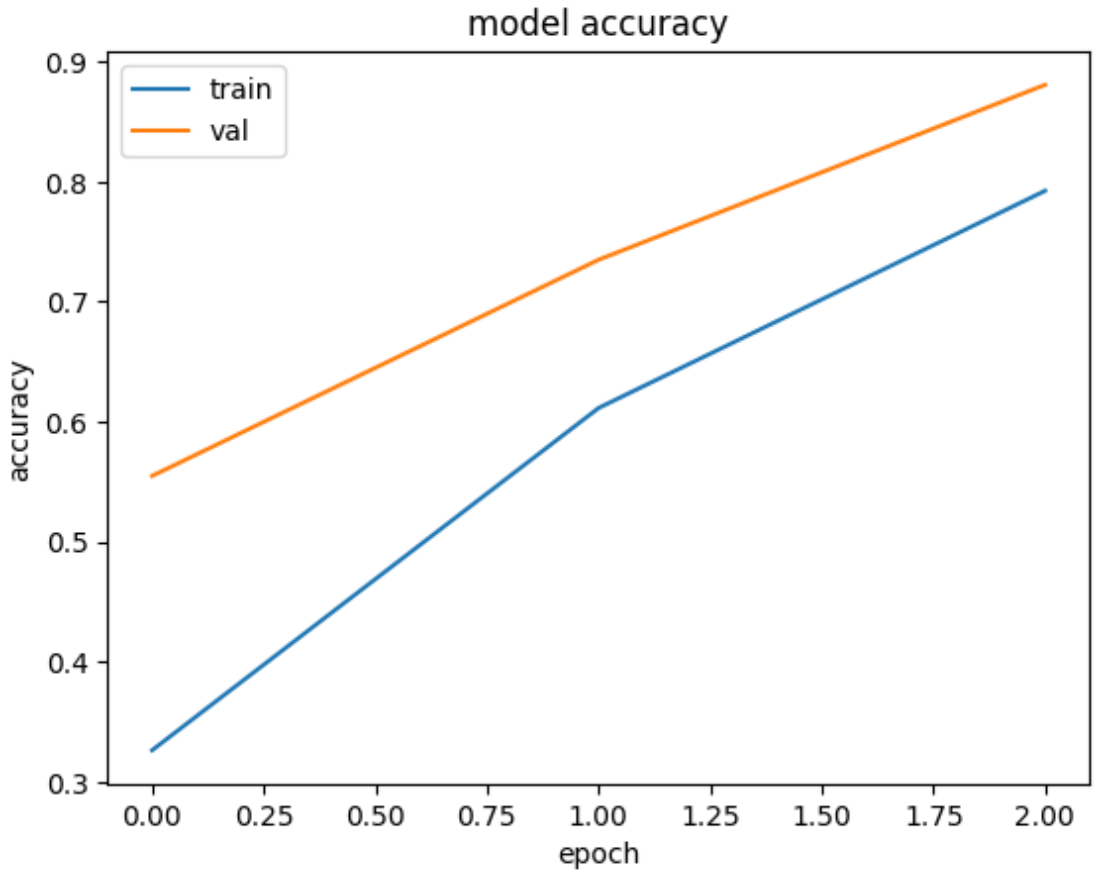
```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_7 (LSTM)               (None, 28, 64)            23808

 dropout_3 (Dropout)         (None, 28, 64)            0

 lstm_8 (LSTM)               (None, 64)                33024

 dense_3 (Dense)             (None, 32)                2080

 dropout_4 (Dropout)         (None, 32)                0

 dense_4 (Dense)             (None, 10)                330

=================================================================
Total params: 59,242
Trainable params: 59,242
Non-trainable params: 0
_________________________________________________________________
```

2.4. Huấn luyện mô hình LSTM

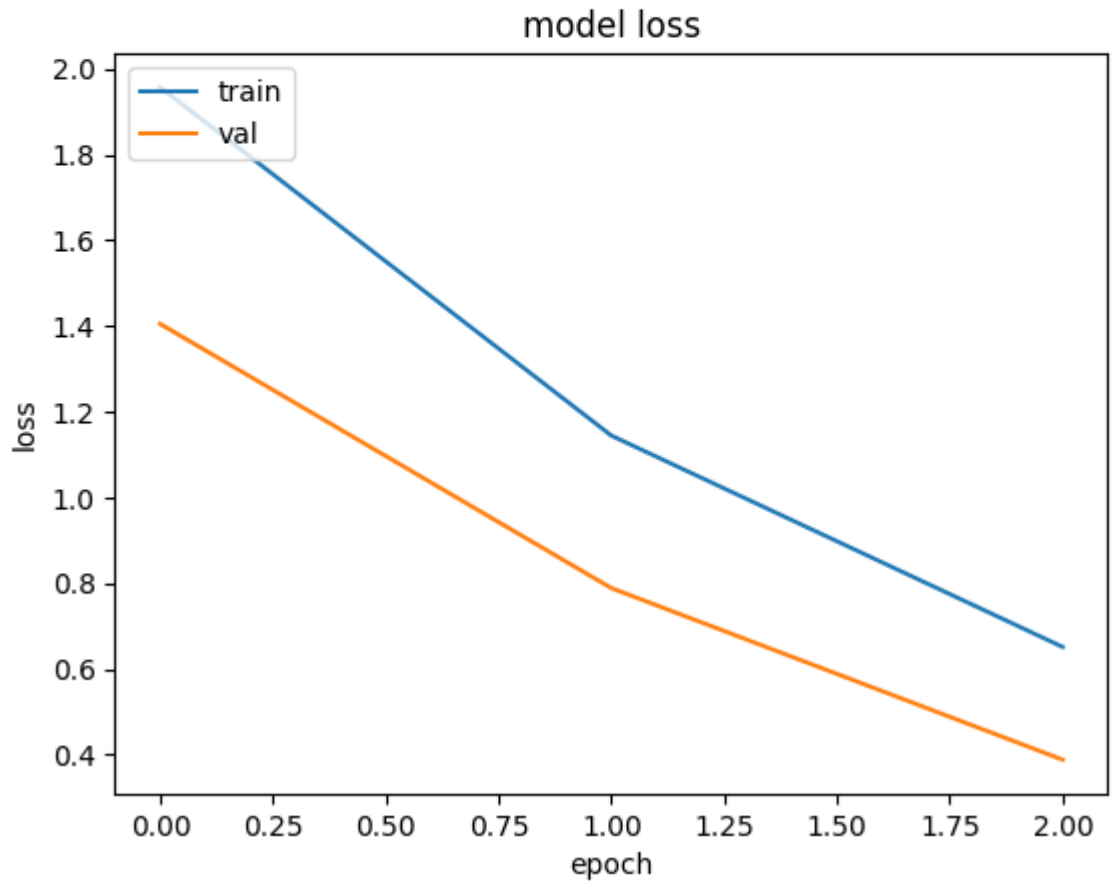
```
[33] model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
     model_fit = model.fit(x_train, y_train, validation_split=0.1, epochs=3, verbose=1)
```

2.5. Đánh giá mô hình

```
[34] plt.plot(model_fit.history['accuracy'])
     plt.plot(model_fit.history['val_accuracy'])
     plt.title('model accuracy')
     plt.ylabel('accuracy')
     plt.xlabel('epoch')
     plt.legend(['train', 'val'], loc='upper left')
     plt.show()
```



```
[35] plt.plot(model_fit.history['loss'])
     plt.plot(model_fit.history['val_loss'])
     plt.title('model loss')
     plt.ylabel('loss')
     plt.xlabel('epoch')
     plt.legend(['train', 'val'], loc='upper left')
     plt.show()
```



2.6. Dự báo của mô hình

```
for i in range(10):
    result = tf.argmax(model.predict(tf.expand_dims(x_test[i], 0)), axis=1)
    print(result.numpy(), y_test[i])
```

```
1/1 [==============================] - 1s 875ms/step
[7] 7
1/1 [==============================] - 0s 28ms/step
[2] 2
1/1 [==============================] - 0s 29ms/step
[1] 1
1/1 [==============================] - 0s 27ms/step
[0] 0
1/1 [==============================] - 0s 26ms/step
[4] 4
1/1 [==============================] - 0s 28ms/step
[1] 1
1/1 [==============================] - 0s 31ms/step
[4] 4
1/1 [==============================] - 0s 38ms/step
[9] 9
1/1 [==============================] - 0s 36ms/step
[6] 5
1/1 [==============================] - 0s 39ms/step
[9] 9
```

3. SỬ DỤNG LSTM ĐỂ SINH TỪ TRONG VĂN BẢN

3.1. Giới thiệu

Sinh văn bản là việc dự báo từ tiếp theo trong một chuỗi đầu vào.

3.2. Nạp thư viện

```
import pandas as pd
import numpy as np
import string, os

from keras.utils import pad_sequences
from keras.layers import Embedding, LSTM, Dense, Dropout
from keras.preprocessing.text import Tokenizer
from keras.callbacks import EarlyStopping
from keras.models import Sequential
import keras.utils as ku
```

3.3. Nạp dữ liệu

Ta dùng dữ liệu New York Times comments để minh họa. Dữ liệu này chứa các comment của độc giả bình luận về các bản tin của tờ New York Times

3.3.1. Làm sạch dữ liệu

Chúng ta cần loại bỏ các dấu chấm câu, làm lowercase các từ như N.F.L → nfl

```
[12] def clean_text(txt):
       txt = "".join(t for t in txt if t not in string.punctuation).lower()
       return txt
     corpus = [clean_text(txt) for txt in df['headline']]
     print(corpus[:10])

     ['finding an expansive view  of a forgotten people in niger', 'and now  the dreaded trump curse', 'venezuela's descent into dictatorship',
```

3.3.2. Sinh chuỗi n-gram cho huấn luyện

```
[16] tokenizer = Tokenizer()
     def get_sequence_of_tokens(corpus):
         ## tokenization
         #create word - index : I am a student --> word_index['I'] = 1; word_index['am'] = 2
         tokenizer.fit_on_texts(corpus)
         total_words = len(tokenizer.word_index) + 1


         ## convert data to a token sequence
         input_sequences = []
         for line in corpus[:2]:
             #convert word into index according to word_index
             #[[169, 21, 653, 359, 4, 2, 654, 170, 5, 655]]
             token_list = tokenizer.texts_to_sequences([line])[0]
             for i in range(1, len(token_list)):
                 n_gram_sequence = token_list[:i+1]
                 input_sequences.append(n_gram_sequence)
         return input_sequences, total_words
     inp_sequences, total_words = get_sequence_of_tokens(corpus)
     print(inp_sequences[:10])
     print(total_words)

     [[169, 21, 653, 359, 4, 2, 654, 170, 5, 655]]
     [[6, 80, 1, 656, 11, 657]]
     [[169, 21], [169, 21, 653], [169, 21, 653, 359], [169, 21, 653, 359, 4], [169, 21, 653, 359, 4, 2],
     2484
```

3.3.3. Padding sequences

Ngram	Sequence of Tokens
i stand	[30, 507]
i stand with	[30, 507, 11]
i stand with the	[30, 507, 11, 1]
i stand with the shedevils	[30, 507, 11, 1, 975]

```
def generate_padded_sequences(input_sequences):
    max_sequence_len = max([len(x) for x in input_sequences])
    input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre'))

    predictors, label = input_sequences[:,:-1],input_sequences[:,-1]
    label = ku.to_categorical(label, num_classes=total_words)
    return predictors, label, max_sequence_len

predictors, label, max_sequence_len = generate_padded_sequences(inp_sequences)
print(predictors)
print(label)
print(max_sequence_len)
```

```
[[  0   0   0   0   0   0   0   0 169]
 [  0   0   0   0   0   0   0 169  21]
 [  0   0   0   0   0   0 169  21 653]
 [  0   0   0   0   0 169  21 653 359]
 [  0   0   0   0 169  21 653 359   4]
 [  0   0   0 169  21 653 359   4   2]
 [  0   0 169  21 653 359   4   2 654]
 [  0 169  21 653 359   4   2 654 170]
 [169  21 653 359   4   2 654 170   5]
 [  0   0   0   0   0   0   0   0   6]
 [  0   0   0   0   0   0   0   6  80]
 [  0   0   0   0   0   0   6  80   1]
 [  0   0   0   0   0   6  80   1 656]
 [  0   0   0   0   6  80   1 656  11]]
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
10
```

3.4. Định nghĩa mô hình LSTM

Ta dùng Predictors như tập mẫu huấn luyện, labels là nhãn cần dự báo

```
def create_model(max_sequence_len, total_words):
    input_len = max_sequence_len - 1
    model = Sequential()
    # ----------Add Input Embedding Layer
    model.add(Embedding(total_words, 10, input_length=input_len))
    # ----------Add Hidden Layer 1 - LSTM Layer
    model.add(LSTM(100))
    model.add(Dropout(0.1))
    # ----------Add Output Layer
    model.add(Dense(total_words, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam')
    return model
model = create_model(max_sequence_len, total_words)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 9, 10)             24840

 lstm (LSTM)                 (None, 100)               44400

 dropout (Dropout)           (None, 100)               0

 dense (Dense)               (None, 2484)              250884

=================================================================
Total params: 320,124
Trainable params: 320,124
Non-trainable params: 0
```

3.5. Huấn luyện mô hình

```
model.fit(predictors, label, epochs=100, verbose=1)
```

```
Epoch 1/100
1/1 [==============================] - 0s 21ms/step - loss: 2.6404
Epoch 2/100
1/1 [==============================] - 0s 21ms/step - loss: 2.6675
Epoch 3/100
1/1 [==============================] - 0s 22ms/step - loss: 2.6754
Epoch 4/100
1/1 [==============================] - 0s 21ms/step - loss: 2.6505
Epoch 5/100
1/1 [==============================] - 0s 21ms/step - loss: 2.6586
```

3.6. Dự báo từ tiếp theo

```
def generate_text(seed_text, next_words, model, max_sequence_len):
    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = pad_sequences([token_list],maxlen=max_sequence_len-1, padding='pre')
        predicted = model.predict(token_list, verbose=0)
        predicted_idx = np.argmax(predicted)

        output_word = ""
        for word,index in tokenizer.word_index.items():
            if index == predicted_idx:
                output_word = word
                break
        seed_text += " "+ output_word


    return seed_text.title()

print (generate_text("india and pakistan", 3, model, max_sequence_len))
print (generate_text("president trump", 3, model, max_sequence_len))
print (generate_text("united states", 4, model, max_sequence_len))
print (generate_text("donald trump", 2, model, max_sequence_len))
print (generate_text("new york", 3, model, max_sequence_len))
print (generate_text("science and technology", 5, model, max_sequence_len))
```

```
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
India And Pakistan Fury’ Puts Rubin
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 26ms/step
President Trump He’D Storm In
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 46ms/step
United States In Kiev Fuels War
1/1 [==============================] - 0s 52ms/step
```

4. BÀI TẬP

1. Viết chương trình cài đặt LSTM để nhận dạng ảnh trên bộ dataset CIFAR10 có sẵn trong tensorflow với các nhãn sau

Label	Description
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

2. Hãy viết chương trình cài đặt LSTM để nhận dạng ảnh Cat hoặc Dog. Dữ liệu do giảng viên cung cấp

3. Hãy viết chương trình cài đặt LSTM để nhận dạng ảnh Fashion-MNIST. Dữ liệu do giảng viên cung cấp.

4. Hãy viết chương trình cài đặt LSTM để nhận dạng ảnh khuôn mặt {Nam, Nữ}. Dữ liệu do giảng viên cung cấp.

5. Hãy viết chương trình cài đặt LSTM để sinh văn bản từ tập thơ Truyện Kiều của đại thi hào Nguyễn Du.

6. Hãy viết chương trình cài đặt LSTM để sinh văn bản từ bộ dữ liệu Twitter, lấy cột twitter content. Yêu cầu LSTM trả lời tròn câu (có dấu chấm cuối câu)

7. Triển khai câu 1 - 7 trên nền tảng WEB sử dụng Flask