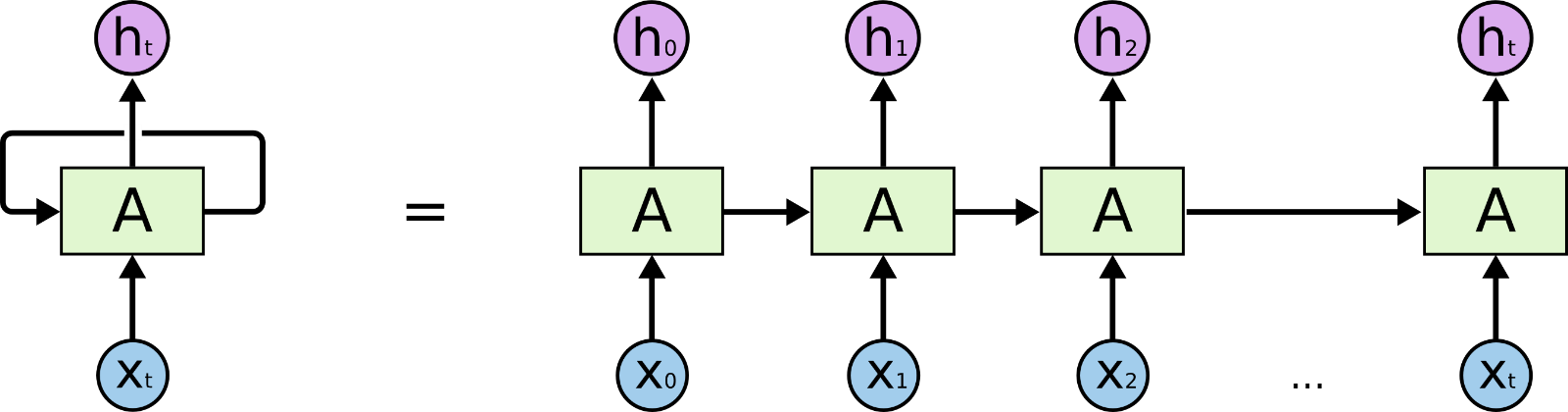
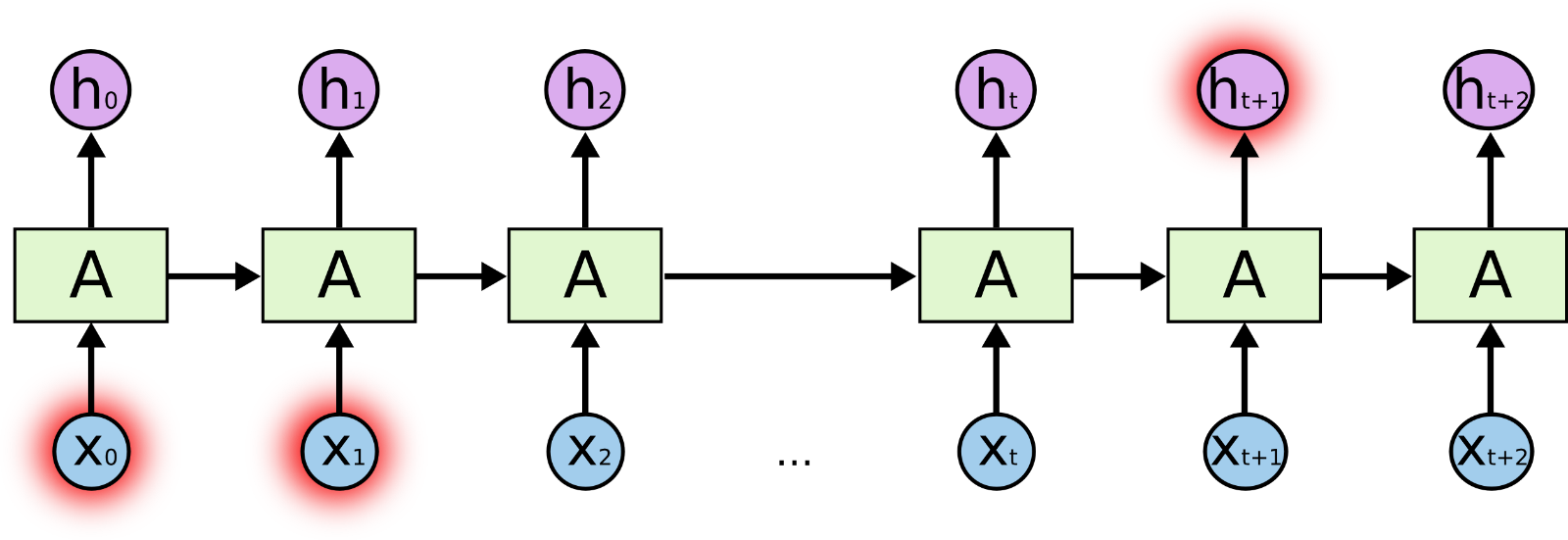
LSTM

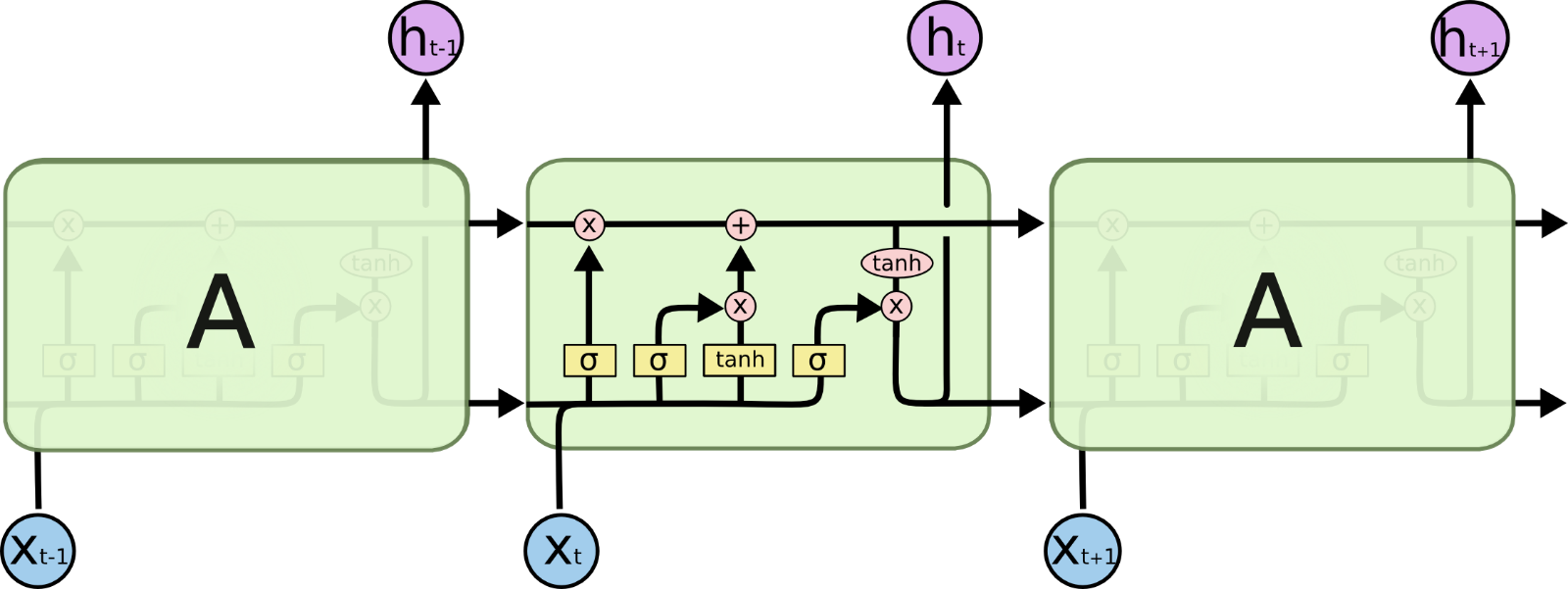
Mạng hồi quy RNN: Một mạng nơ-ron hồi quy có thể được coi là nhiều bản sao chép của cùng một mạng, trong đó mỗi đầu ra của mạng này là đầu vào của một mạng sao chép khác.



Tuy nhiên, với khoảng cách lớn dần thì RNN không học được các phụ thuộc xa.

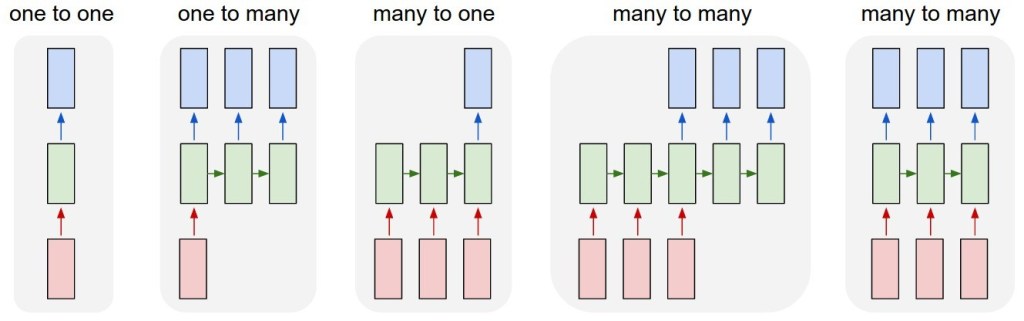


LSTM: là một dạng đặc biệt của RNN, nó có khả năng học được phụ thuộc xa.



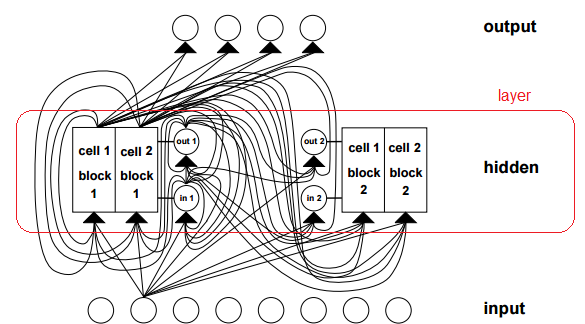


*Cấu trúc mạng lSTM*



*Các dạng bài toán*

(\* model many to many cuối là One to One Sequence Prediction Model Over Time, khác với One to One ở đầu là One to One Sequence Prediction Model)



*The computational unit of the LSTM network is called the memory cell, memory block, or just cell for short.*

*–(Devolop Sequence Prediction Models With Deep Learning– Jason Brownlee)*

LSTM01 : One to One Model

LSTM02 : One to One Model Upgrade version

(không có tác dụng nhiều lắm)

LSTM03 : Many to One Model

(cho kết quả khá tốt)

LSTM01

import numpy

import matplotlib.pyplot as plt

from pandas import read\_csv

import math

from keras.models import Sequential,load\_model,save\_model

from keras.layers import Dense

from keras.layers import LSTM

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error

numpy : Thư viện tính toán số học trong Python

matplotlib.pylot: Dùng để vẽ đồ thị

keras: Thư viện hỗ trợ deep learning

sklearn.preprocessing import MinMaxScaler: chuyển dữ liệu về khoảng (0;1)

# fix random seed for reproducibility

numpy.random.seed(7)

# load the dataset

path\_train = 'D:/MITECH/2020/DATA/NN\_data2/train/export\_data\_x\_center\_train\_40.csv'

data\_train = read\_csv(path\_train)

data\_train = data\_train.to\_numpy()

shape\_0 = data\_train.shape[0]

shape\_1 = data\_train.shape[1]

numpy.random.seed() : Khởi tạo các bộ sinh số ngẫu nhiên

path\_train : đường dẫn đến dữ liệu

.to\_numpy(): số hóa dữ liệu thu được

Shape\_0 : số cột của dữ liệu (ví dụ ở đây là ~ 700000 cột)

Shape\_1 : số hàng của dữ liệu (ví dụ ở đây là 41 hàng)

look\_back = 40

look\_back : một biến RẤT quan trọng trong chương trình, có thể coi là biểu diễn của số input trong 1 lần train , tức là số bước thời gian được xét đến trong 1 lần train

data\_train = data\_train.reshape(shape\_0\*shape\_1, 1)

scaler\_train = MinMaxScaler(feature\_range=(0, 1))

data\_train = scaler\_train.fit\_transform(data\_train)

data\_train = data\_train.reshape(shape\_0,shape\_1)

X\_train = data\_train[:,0:(shape\_1-1)]

Y\_train = data\_train[:,(shape\_1-1)]

X\_train = numpy.reshape(X\_train, (X\_train.shape[0], 1, X\_train.shape[1]))

Chuyển dữ liệu từ khoảng [-1 , 1600] về (0 , 1)

data\_train.reshape(shape\_0\*shape\_1, 1) : bước đầu tiên là phải chuyển mảng đang xét về dạng (N , 1) . Việc này là cần thiết vì lúc dạng dữ liệu để inverse\_tranform() cần giống với fit\_transform

MinMaxScaler(feature\_range=(0, 1)): Hàm chuyển dữ liệu về khoảng( 0 , 1)

fit\_transform(data\_train) : Sử dụng hàm . Muốn chuyển ngược lại sau khi sử dụng hàm , ta dùng lệnh inverse\_tranform()

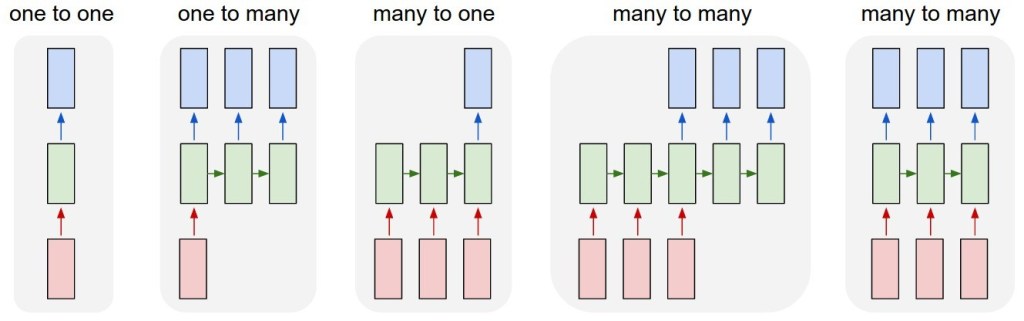
X\_train: là input

Y\_train: là output

numpy.reshape(X\_train, (X\_train.shape[0], 1, X\_train.shape[1]))

Dữ liệu vào của 1 mạng LSTM có dạng (samples ,time steps ,feature)

Ở đây, input 20 mà time steps =1 thì đây là One to One Sequence Prediction Model Over Time



model = Sequential()

model.add(LSTM(20, input\_shape=(1, look\_back)))

model.add(Dense(1,))

model.compile(loss='mean\_squared\_error', optimizer='adam')

model.fit(X\_train, Y\_train, epochs=200, batch\_size=32)

model.save('D:/MITECH/2020/LSTM/LSTM\_input\_x\_c\_40\_20\_1\_eps200\_bz32.h5')

Tạo model :

model.add(LSTM(20, input\_shape=(1, look\_back))):  
LSTM(20) thì 20 là số memory cell .

input\_shape(1,lookback) : là dạng input shape dùng cho trường hợp này

adam : tự động thay đổi learning rate, tránh việc điều chỉnh learning rate bằng tay

Đánh giá :

This is a poor use of the LSTM as it is not capable of learning across input or output time steps. This model does put all of the pressure on the internal state or memory.

Đây là việc sử dụng LSTM kém vì nó không có khả năng học qua các bước thời gian đầu vào hoặc đầu ra. Mô hình này không gây áp lực lên trạng thái bên trong hoặc bộ nhớ.

Thời gian : với batchsize = 1, epochs = 4 , thì khoảng 2 tiếng cho 1 lần train

LSTM02

import numpy

import matplotlib.pyplot as plt

from pandas import read\_csv

import math

from keras.models import Sequential,load\_model

from keras.layers import Dense

from keras.layers import LSTM

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error

numpy : Thư viện tính toán số học trong Python

matplotlib.pylot: Dùng để vẽ đồ thị

keras: Thư viện hỗ trợ deep learning

sklearn.preprocessing import MinMaxScaler: chuyển dữ liệu về khoảng (0;1)

def create\_dataset(dataset,look\_back):

    dataX,dataY = [],[]

    for j in range(len(dataset)-look\_back-1):

        a = dataset[j:(j+look\_back)]

        dataX.append(a)

        dataY.append(dataset[j+look\_back])

    return numpy.array(dataX) , numpy.array(dataY)

Chuyển dữ liệu từ 1 hàng ( ví dụ : gồm 40 phần tử) về dạng

|  |  |
| --- | --- |
| 1  2  3  4  ...  20 | X1 X2 ... X20 Y  112 118 ... 132 129  118 132 ... 129 121  132 129 ... 121 135  ... ... ... ... |
|  |  |

# fix random seed for reproducibility

numpy.random.seed(7)

# load the dataset

path\_train = 'D:/MITECH/2020/DATA/NN\_data2/train/export\_data\_x\_center\_train\_40.csv'

data\_train = read\_csv(path\_train)

data\_train = data\_train.to\_numpy()

shape\_0 = data\_train.shape[0]

shape\_1 = data\_train.shape[1]

look\_back = 20

data\_train = data\_train.reshape(shape\_0\*shape\_1, 1)

scaler\_train = MinMaxScaler(feature\_range=(0, 1))

data\_train = scaler\_train.fit\_transform(data\_train)

data\_train = data\_train.reshape(shape\_0,shape\_1)

model = Sequential()

model.add(LSTM(20, input\_shape=(1, look\_back)))

model.add(Dense(1,))

model.compile(loss='mean\_squared\_error', optimizer='adam')

for i in range(shape\_0):

    dataset\_train = data\_train[i,:]

    trainX,trainY = create\_dataset(dataset\_train,20)

    trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))

    model.fit(trainX, trainY, epochs=5, batch\_size=20, verbose=1)

    if i//100000 == 0:

        print(i)

model.save('D:/MITECH/2020/LSTM/LSTM\_missing\_data\_input\_x\_c\_20\_20\_1\_eps5\_batsz20.h5')

Tương tự như các model khác, nhưng ở đây, ta fit model nhiều lần.

Đánh giá: Kết quả của LSTM02 cũng không khá hơn LSTM01 nhiều lắm, thậm chí có thể còn tồi hơn. Chưa rõ là do model hay là do cách chọn các biến lúc fit ,nhưng với thời gian train dài hơn hẳn so với LSTM01 mà kết quả không tốt hơn hẳn thì không nên sử dụng LSTM02

Thời gian : với số liệu như trên thì khoảng 6,7 tiếng cho 1 lần train

LSTM03(many to one)

import numpy

import matplotlib.pyplot as plt

from pandas import read\_csv

import math

from keras.models import Sequential,load\_model,save\_model

from keras.layers import Dense, Activation

from keras.layers import LSTM

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error

numpy : Thư viện tính toán số học trong Python

matplotlib.pylot: Dùng để vẽ đồ thị

keras: Thư viện hỗ trợ deep learning

sklearn.preprocessing import MinMaxScaler: chuyển dữ liệu về khoảng (0;1)

# fix random seed for reproducibility

numpy.random.seed(7)

# load the dataset

path\_train = 'D:/MITECH/2020/DATA/NN\_data2/train/export\_data\_x\_center\_train\_20.csv'

data\_train = read\_csv(path\_train)

data\_train = data\_train.to\_numpy()

shape\_0 = data\_train.shape[0]

shape\_1 = data\_train.shape[1]

numpy.random.seed() : Khởi tạo các bộ sinh số ngẫu nhiên

path\_train : đường dẫn đến dữ liệu

.to\_numpy(): số hóa dữ liệu thu được

Shape\_0 : số cột của dữ liệu (ví dụ ở đây là ~ 700000 cột)

Shape\_1 : số hàng của dữ liệu (ví dụ ở đây là 21 hàng)

look\_back = 20

look\_back : một biến RẤT quan trọng trong chương trình, có thể coi là biểu diễn của số input trong 1 lần train , tức là số bước thời gian được xét đến trong 1 lần train

data\_train = data\_train.reshape(shape\_0\*shape\_1, 1)

scaler\_train = MinMaxScaler(feature\_range=(0, 1))

data\_train = scaler\_train.fit\_transform(data\_train)

data\_train = data\_train.reshape(shape\_0,shape\_1)

X\_train = data\_train[:,0:(shape\_1-1)]

Y\_train = data\_train[:,(shape\_1-1)]

X\_train = numpy.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1],1))

Chuyển dữ liệu từ khoảng [-1 , 1600] về (0 , 1)

data\_train.reshape(shape\_0\*shape\_1, 1) : bước đầu tiên là phải chuyển mảng đang xét về dạng (N , 1) . Việc này là cần thiết vì lúc dạng dữ liệu để inverse\_tranform() cần giống với fit\_transform

MinMaxScaler(feature\_range=(0, 1)): Hàm chuyển dữ liệu về khoảng( 0 , 1)

fit\_transform(data\_train) : Sử dụng hàm . Muốn chuyển ngược lại sau khi sử dụng hàm , ta dùng lệnh inverse\_tranform()

X\_train: là input

Y\_train: là output

numpy.reshape(X\_train, (X\_train.shape[0], 1, X\_train.shape[1]))

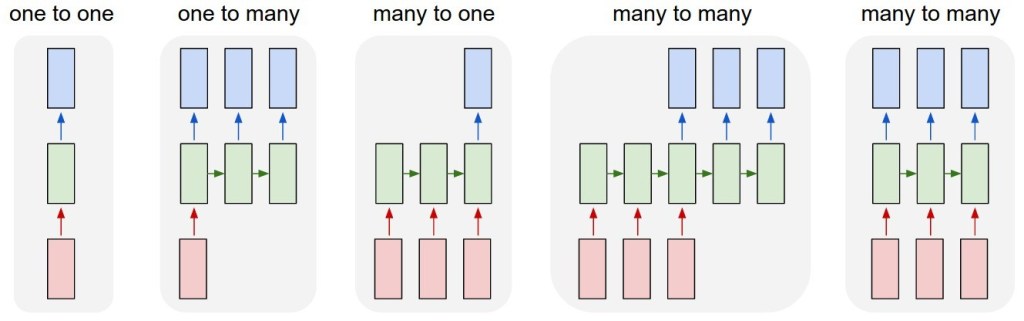
Dữ liệu vào của 1 mạng LSTM có dạng (samples ,time steps ,feature)

Ở đây, input 20 mà time steps =20 thì đây là Many to One Sequence Prediction Model Over Time

Số time step là điểm khác biệt cơ bản giữa LSTM03 và LSTM01

LSTM01: (X\_train.shape[0], 1 ,  X\_train.shape[1])

LSTM03: (X\_train.shape[0], X\_train.shape[1],1)



model = Sequential()

model.add(LSTM(4, input\_shape=(look\_back,1)))

model.add(Dense(1))

model.add(Activation('linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

model.fit(X\_train, Y\_train, epochs= 4, batch\_size=1)

model.save('D:/MITECH/2020/LSTM/LSTM03\_input\_x\_c\_20\_4\_1\_li\_eps6\_bz1.h5')

Tạo model :

model.add(LSTM(20, input\_shape=(look\_back,1 ))):  
LSTM(4) thì 4 là số memory cell .

input\_shape(lookback , 1) : là dạng input shape dùng cho trường hợp này

model.add(Activation('linear')):

Chọn activation funtion là việc quan trọng nhất cho output layer vì nó sẽ quyết định dạng của giá trị dự đoán.

Với Regression : nên chọn lỉnear

Với Binary Classification ( 2 class) : nên chọn sigmoid

Với Multiclass Classification(> 2 class) : nên chọn softmax

* ở bài toán này, ta chọn linear.

adam : tự động thay đổi learning rate, tránh việc điều chỉnh learning rate bằng tay

Đánh giá : Model LSTM03 cho kết quả tốt hơn LSTM01 vì nó không phụ thuộc quá nhiều vào giá trị gần cuối. Ở một số bộ test, nếu giá trị gần cuối là -1 hoặc 1 giá trị khác xa so với giá trị cuối thì LSTM01 sẽ cho kết quả với sai số lớn. Trong khi đó, LSTM03 đã khắc phục được nhược điểm trên.

Thời gian : Với epochs = 6, batchsize = 1 thì mất khoảng 8 tiếng cho 1 lần train