312-3302 ARTIFICIAL INTELLIGENCE

Lecture 8

Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN)

ในบทเรียนที่ผ่านมาโครงสร้างของ NN และ CNN นั้นมีขนาด Input ที่ตายตัวเท่านั้น ทำให้ไม่สามารถรองรับข้อมูลที่มีลักษณะสัมพันธ์ตามลำดับก่อนหลัง (Sequence) ได้

- ข้อความต่างๆ ประโยคภาษาต่างๆ
- ข้อมูลที่มีลำดับเวลา เช่น สภาพอากาศ การเปลี่ยนแปลงค่าเงิน
- ข้อมูลประเภทเสียง สัญญาณที่ได้รับจาก Sensor ในช่วงเวลาต่างๆ

Recurrent Neural Network (RNN)

ข้อมูลต่างๆ ในลักษณะนี้ ถ้าเราสามารถนำมาสร้าง Model ได้ จะทำให้เราสามารถนำผลลัพธ์ที่ได้ไปประยุกต์ใช้ให้เกิดการใช้งานได้หลากหลายรูปแบบ ตัวอย่างเช่น

- การแปลภาษา (Translation)
- การวิเคราะห์ทำนายความรู้สึกจากข้อความ (Sentimental Analysis)
- การทำนายหรือพยากรณ์เหตุการณ์ในอนาคต เช่น สภาพอากาศ การเกิดแผ่นดินไหว
- การวิเคราะห์และทำนายปริมาณการเข้าใช้บริการ หรือการทำนายราคาหุ้น

Image Captioning

RNNs are used to caption an image by analyzing the activities present.

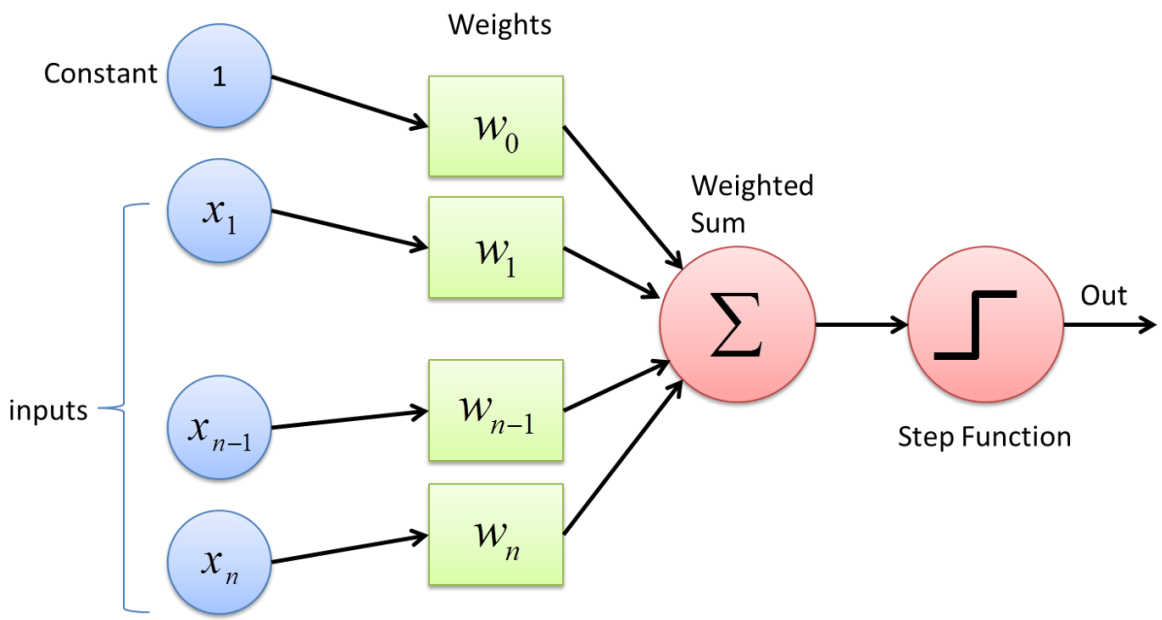
"A Dog catching a ball in mid air"

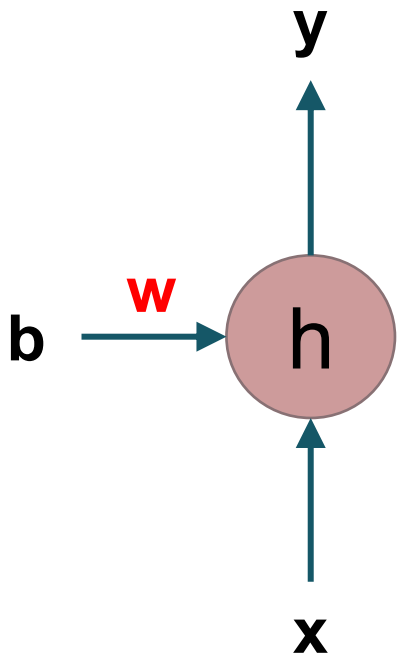
When it rains, look for rainbows.
When it's dark, look for stars.

Positive Sentiment

Recurrent Neural Network (RNN)

การปรับรูปแบบการเขียนโครงสร้างเพื่อให้สะดวกต่อการทำความเข้าใจ





- x คือ Input
- h คือ Hidden node
- y คือ Output
- w คือ Weight
- b คือ Bias

Recurrent Neural Network (RNN)

ตัวอย่าง ถ้าพิจารณาข้อมูลต่อไปนี้

1, 2, 3, 4, 1, 2, 3, 4, ...

A B C D E F ...

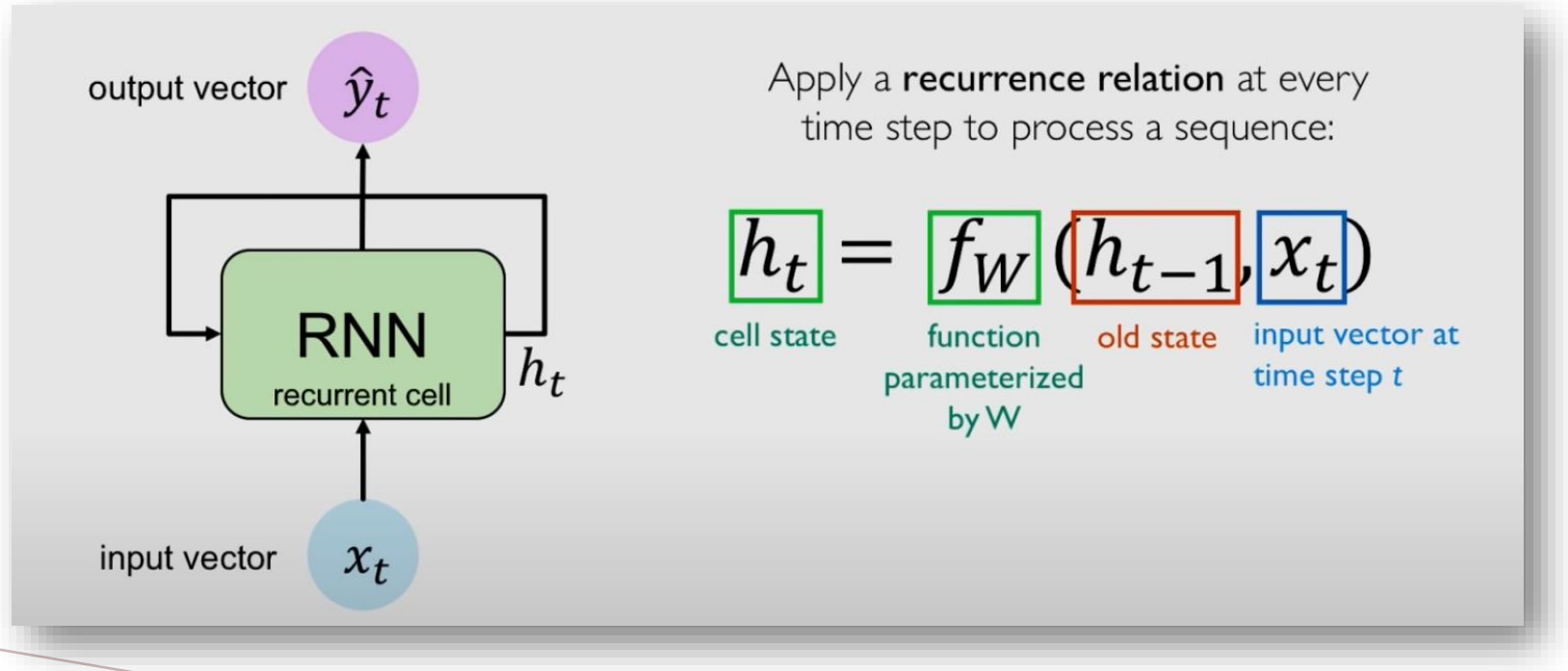
G F E D C B ...

- อ่านข้อมูลทีละครั้ง (Step) แล้วเก็บข้อมูลที่เป็นประโยชน์ไว้ประมวลผลในรอบถัดไป (State) หรือเรียกว่า (Hidden State)
- ความสัมพันธ์ของข้อมูลเชิงลำดับ (Sequence) นั้น มีความสำคัญอย่างมาก เราจะพิจารณาข้อมูลปัจจุบันจาก “ความรู้ก่อนหน้า” หรือ State ซึ่งแตกต่างจาก NN ปกติ ที่จะไม่มีการเก็บ State ไว้ (หรือจะมองว่าเป็นระบบที่มี Memory ก็ได้)

Recurrent Neural Network (RNN)

สถาปัตยกรรมของ RNN

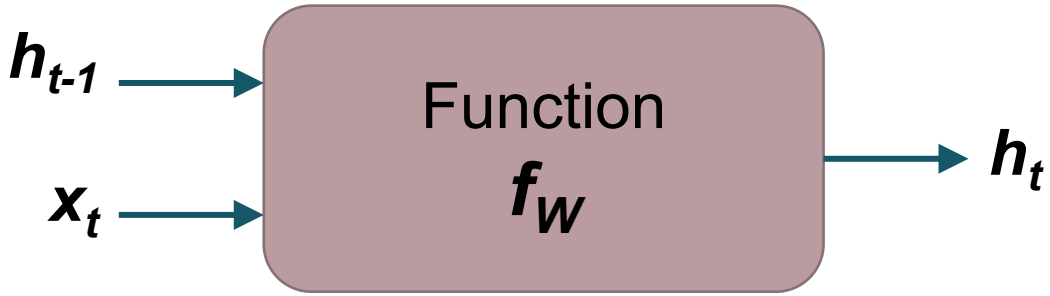
เนื่องจาก RNN มีการเก็บค่าสถานะ ดังนั้นจึงจำเป็นต้องมีตัวแปรเรื่องของเวลา (t) เข้ามาเกี่ยวข้อง

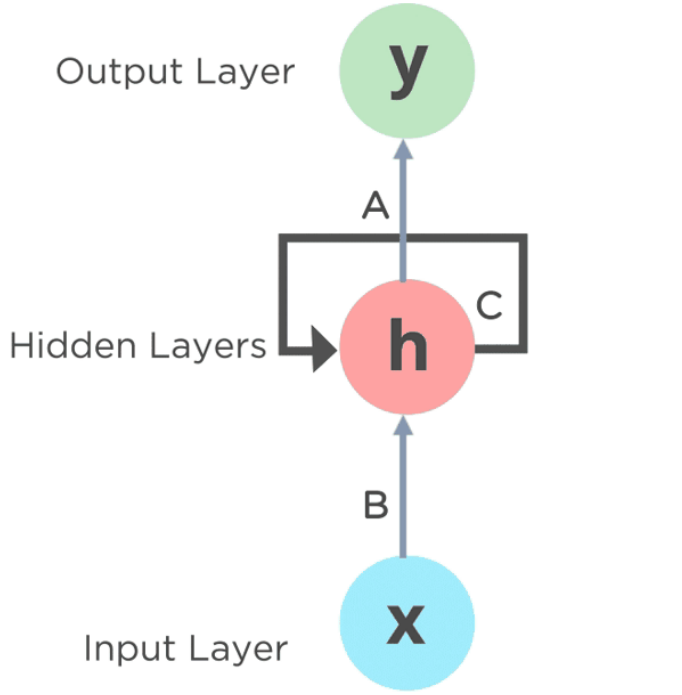


Recurrent Neural Network (RNN)

สถาปัตยกรรมของ RNN

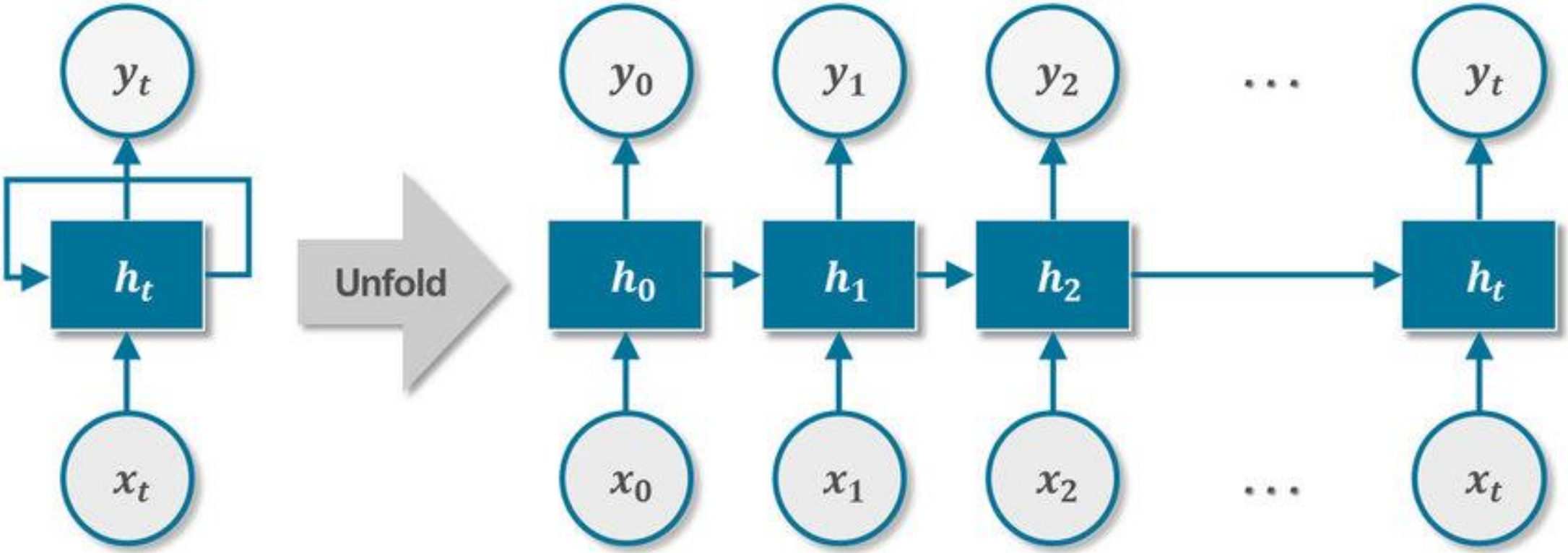
เนื่องจาก RNN มีการเก็บค่าสถานะ ดังนั้นจึงจำเป็นต้องมีตัวแปรเรื่องของเวลา (t) เข้ามาเกี่ยวข้อง





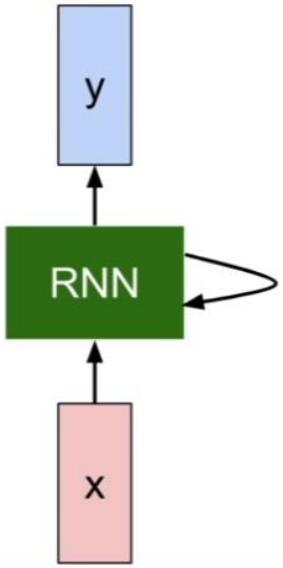
Recurrent Neural Network (RNN)

สถาปัตยกรรมของ RNN



Recurrent Neural Network (RNN)

สถาปัตยกรรมของ RNN



$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

W_{xh} คือ Weight ที่นำมาคูณกับค่า Input

W_{hh} คือ Weight จาก State ก่อนหน้า

h_t คือ ค่าสถานะของ State ปัจจุบัน

x_t คือ Input ที่เวลา State ปัจจุบัน

y_t คือ Output ที่เวลา State ปัจจุบัน

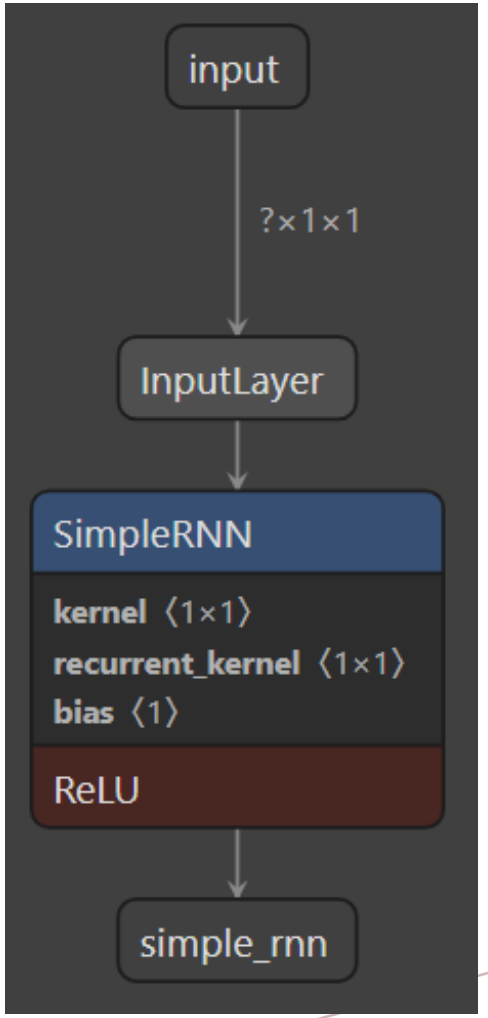
Recurrent Neural Network (RNN)

การเขียนโครงสร้าง RNN

```
import keras.api.models as mod
import keras.api.layers as lay

model = mod.Sequential()
model.add(lay.SimpleRNN(units=1,
                        input_shape=(1,1),
                        activation="relu"))

model.summary()
model.save("RNN.h5")
```



Recurrent Neural Network (RNN)

ทดลองสร้างข้อมูลสำหรับใส่ใน RNN

Example2.py > ...

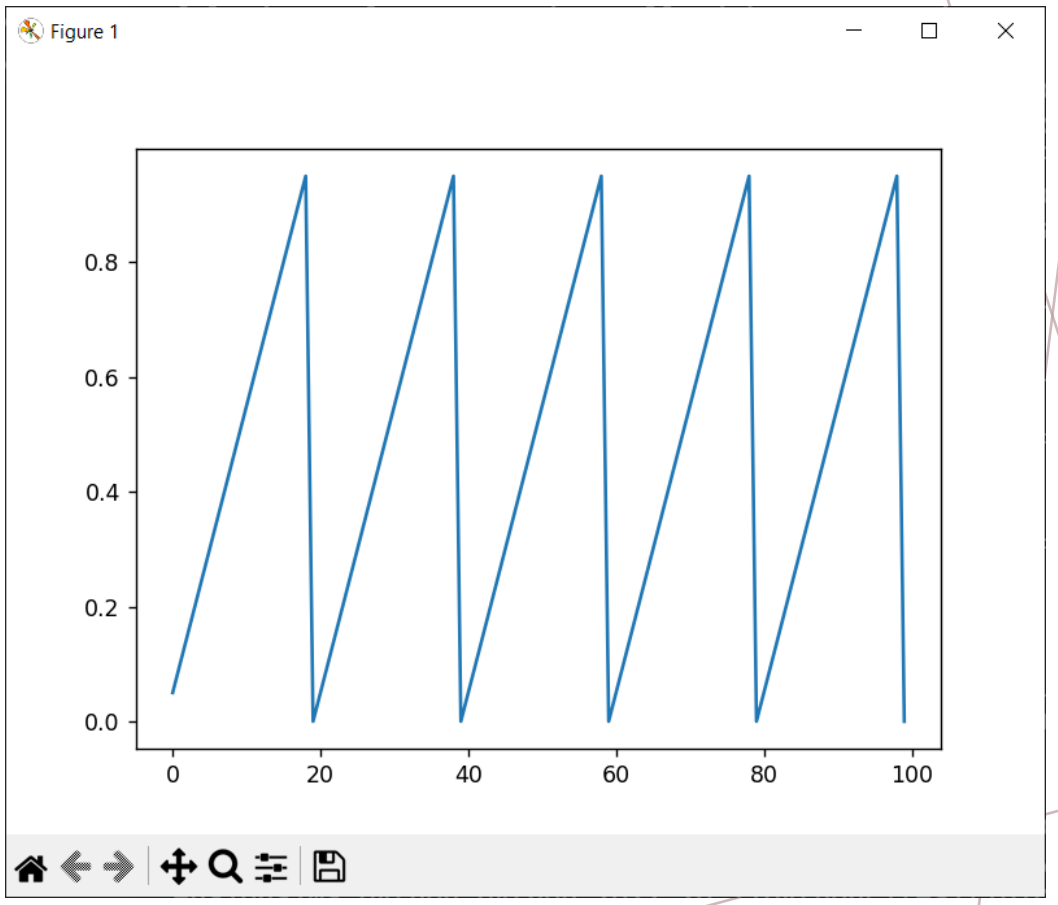
```python
import numpy as np
import matplotlib.pyplot as plt

pitch = 20
step = 1
N = 100
n_train = int(N*0.7)    # 70% for Training set

def gen_data(x):
    return (x%pitch)/pitch

t = np.arange(1, N+1)
y = [gen_data(i) for i in t]
y = np.array(y)

plt.figure()
plt.plot(y)
plt.show()
```

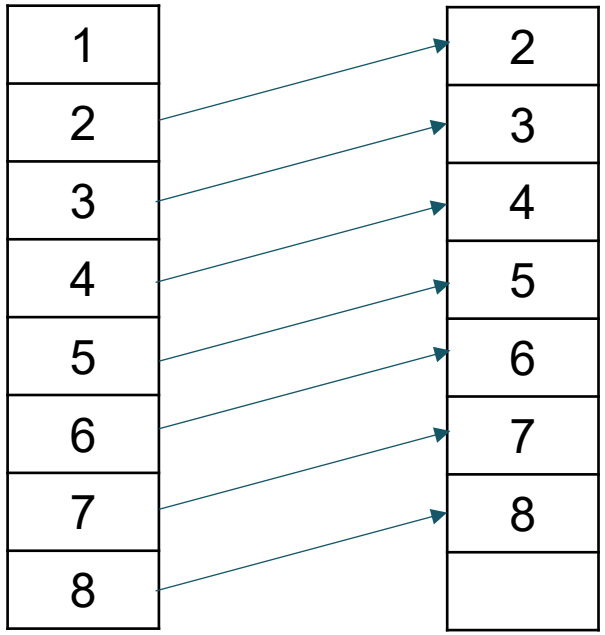


Recurrent Neural Network (RNN)

เตรียมข้อมูลสำหรับใส่ใน RNN

Dataset

1
2
3
4
5
6
7
8



x_train

```
[ [ 1 ]
  [ 2 ]
  [ 3 ]
  [ 4 ]
  [ 5 ]
  [ 6 ] ]
```

y_train

```
[ 2 3 4 5 6 7 ]
```

Step = 1

Recurrent Neural Network (RNN)

เตรียมข้อมูลสำหรับใส่ใน RNN

Dataset

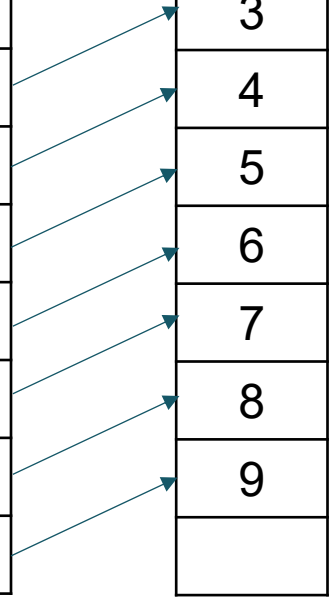
1
2
3
4
5
6
7
8

x_train

1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9

y_train

3
4
5
6
7
8
9



x_train

```
[[ 1  2 ]
 [ 2  3 ]
 [ 3  4 ]
 [ 4  5 ]
 [ 5  6 ]
 [ 6  7 ] ]
```

y_train

```
[ 3 4 5 6 7 8 ]
```

Step = 2

Recurrent Neural Network (RNN)

เตรียมข้อมูลสำหรับใส่ใน RNN

```
23  def convertToMatrix(data, step=1):
24      X, Y = [], []
25      for i in range(len(data)-step):
26          d = i + step
27          X.append(data[i:d,])
28          Y.append(data[d,])
29      return np.array(X), np.array(Y)
30
31  train, test = y[0:n_train], y[n_train:N]
32
33  x_train, y_train = convertToMatrix(train, step)
34  x_test, y_test = convertToMatrix(test, step)
35
36  print("Dimension (Before): ", train.shape, test.shape)
37  print("Dimension (After) : ", x_train.shape, x_test.shape)
```

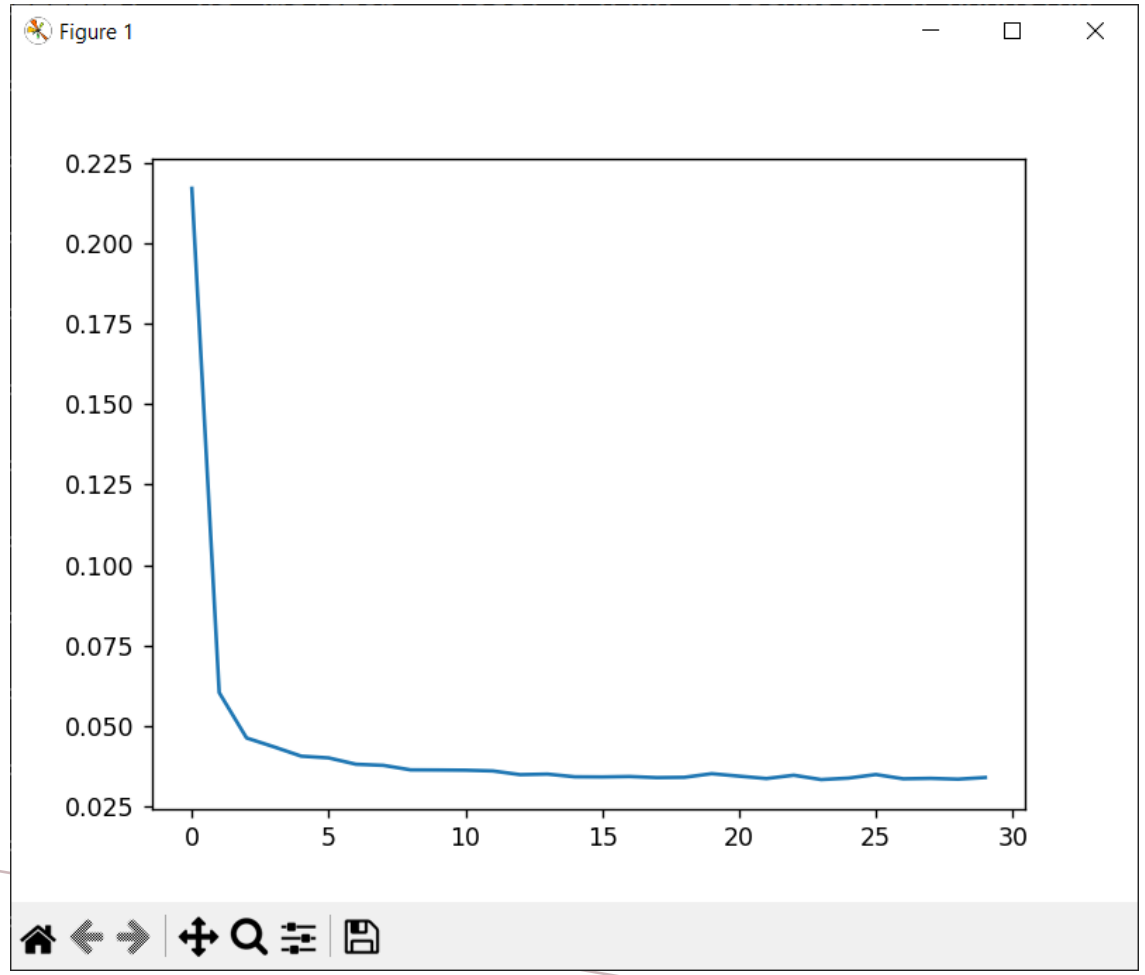
Recurrent Neural Network (RNN)

สร้าง Model และทำการ Training

```
45  model = mod.Sequential()
46  model.add(lay.SimpleRNN(units=32,
47                          input_shape=(step,1),
48                          activation="relu"))
49  model.add(lay.Dense(units=1))
50
51  model.compile(optimizer="adam", loss="mse", metrics=["accuracy"])
52  hist = model.fit(x_train, y_train, epochs=30, batch_size=1, verbose=1)
53
54  plt.plot(hist.history['loss'])
55  plt.show()
```

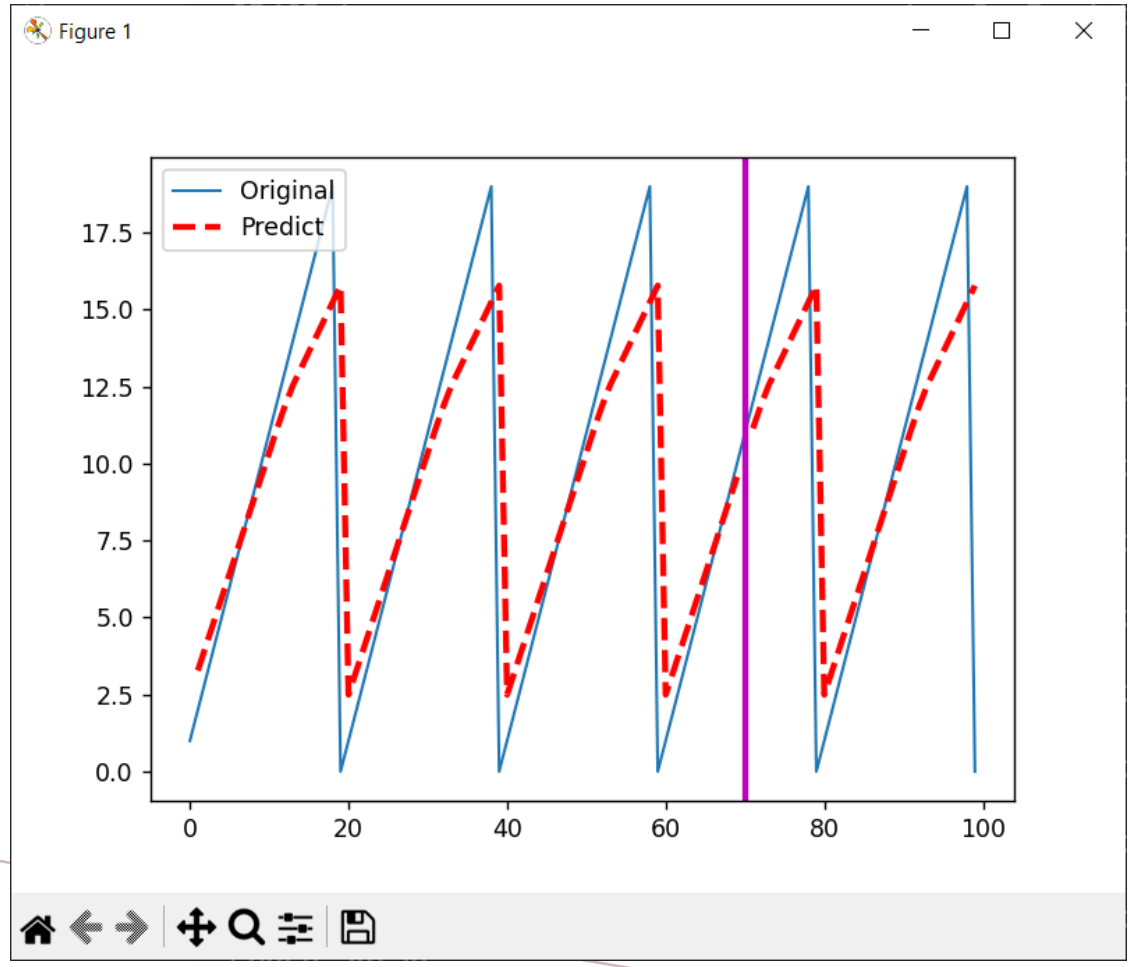
Recurrent Neural Network (RNN)

สร้าง Model และทำการ Training



Recurrent Neural Network (RNN)

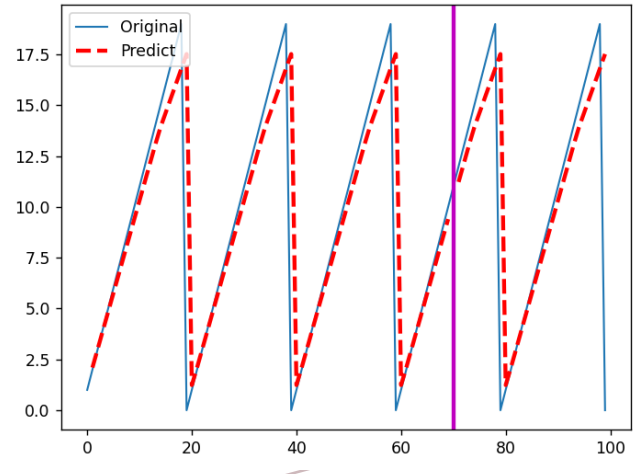
เปรียบเทียบผลลัพธ์การ Predict กับข้อมูลจริง



Classwork

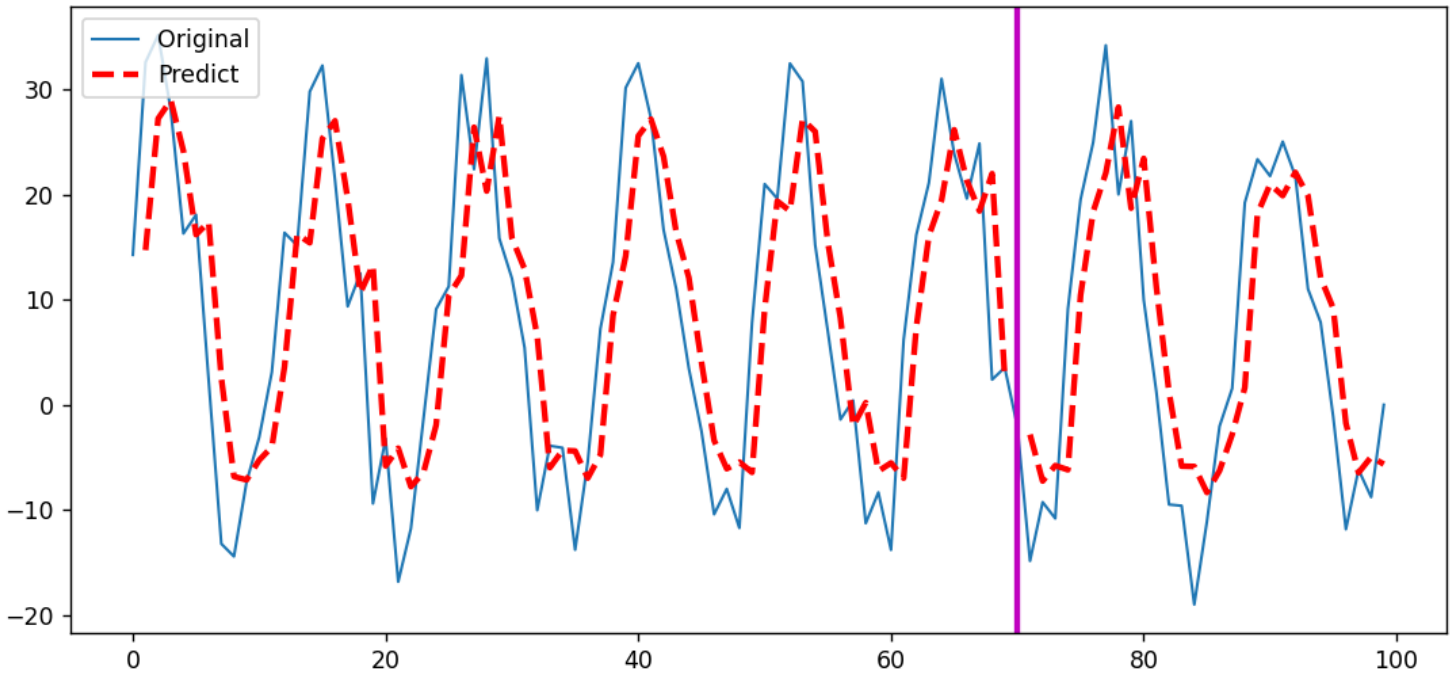
[1] สร้างฟังก์ชั่น Plot เปรียบเทียบผลลัพธ์การ Predict กับต้นฉบับ

[2] ทดลองปรับค่า Parameter ของ Model เพื่อให้ได้ผลลัพธ์ใกล้เคียงกับต้นฉบับที่สุด



Recurrent Neural Network (RNN)

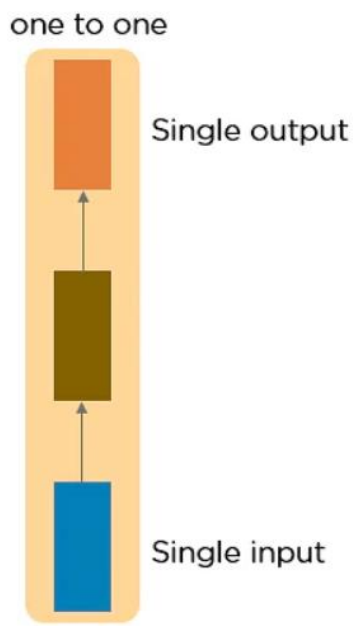
[3] ทดลองเปลี่ยนสัญญาณ Input ให้เป็นลักษณะต่างๆ




```
y = np.sin(0.05*t*10) + 0.8 * np.random.rand(N)
```

Types of Recurrent Neural Networks

RNN มีโครงสร้างทั้งหมด 4 รูปแบบ



Vanilla
Neural Network

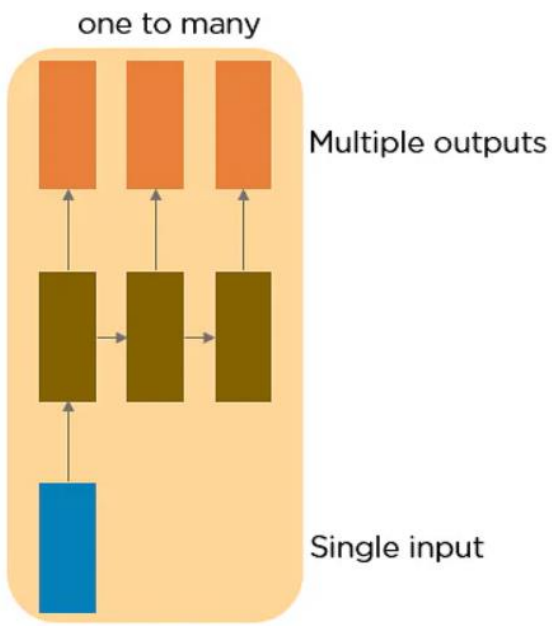
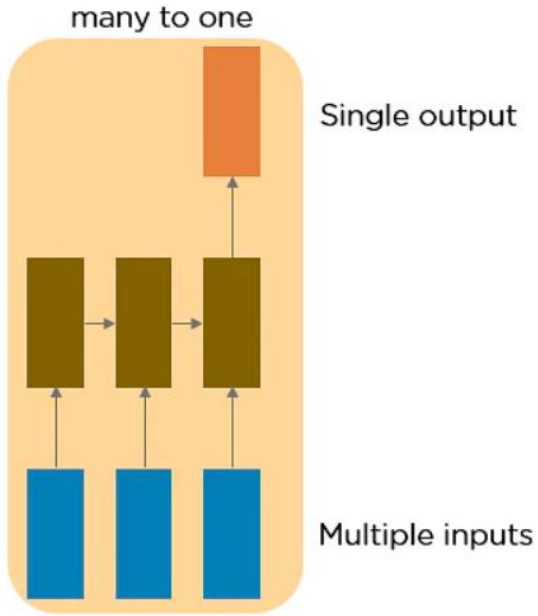
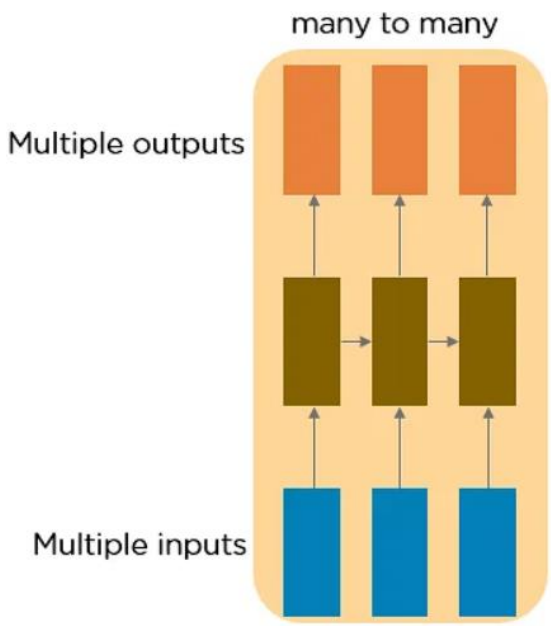


Image caption



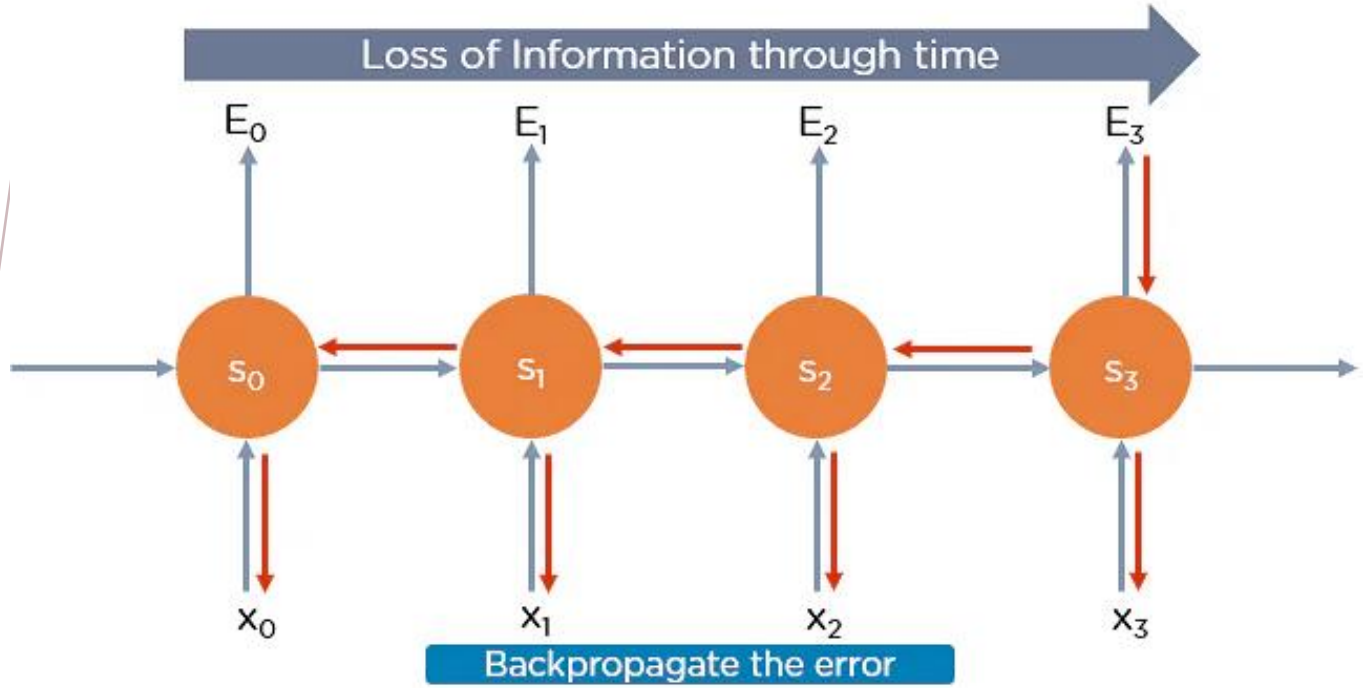
Sentiment analysis

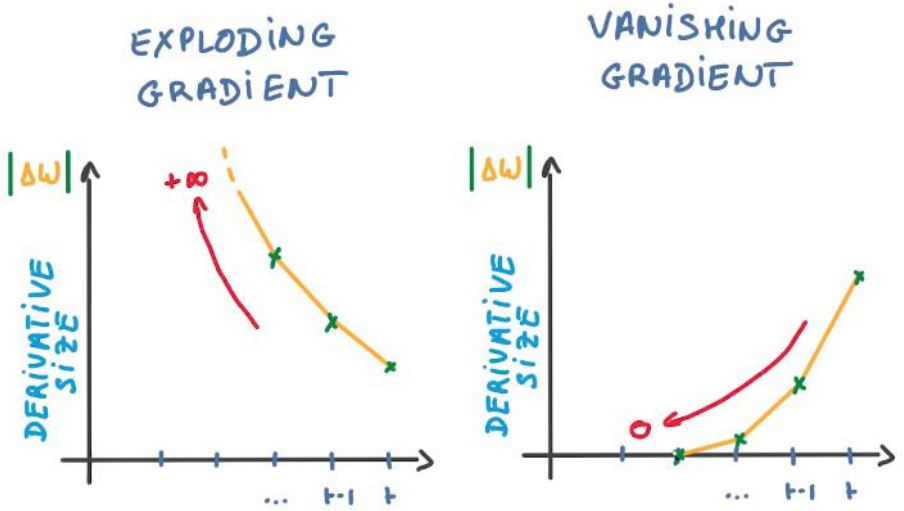


Machine translation

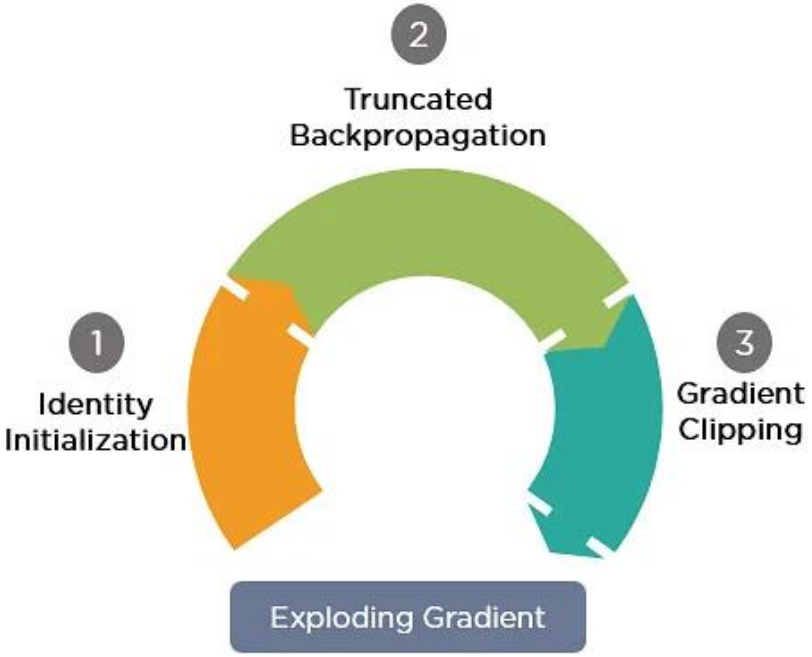
Two Issues of Standard RNNs

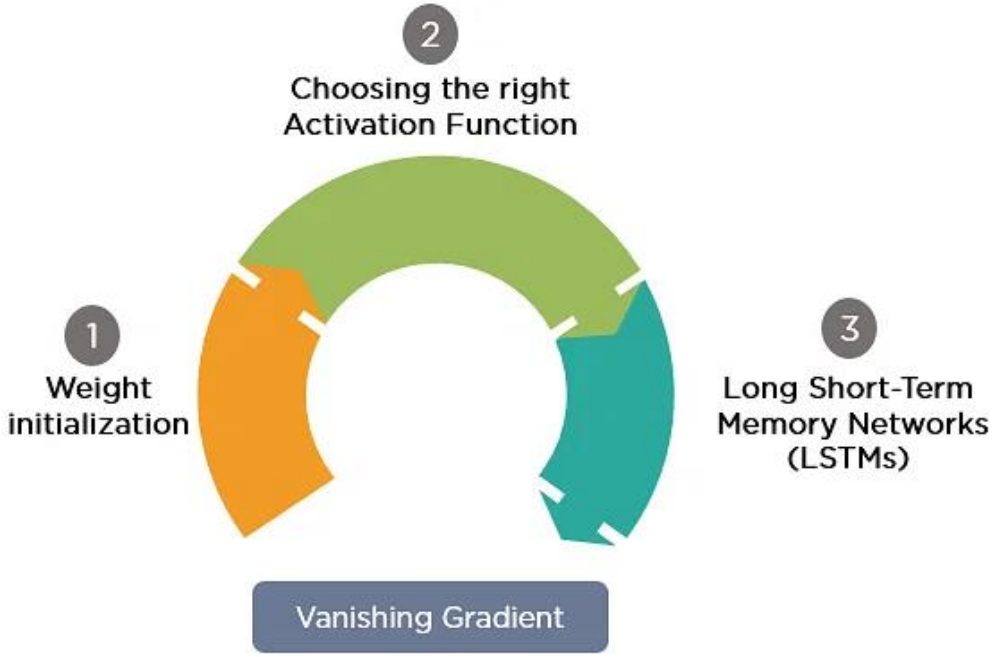
Vanishing Gradient Problem & Exploding Gradient Problem

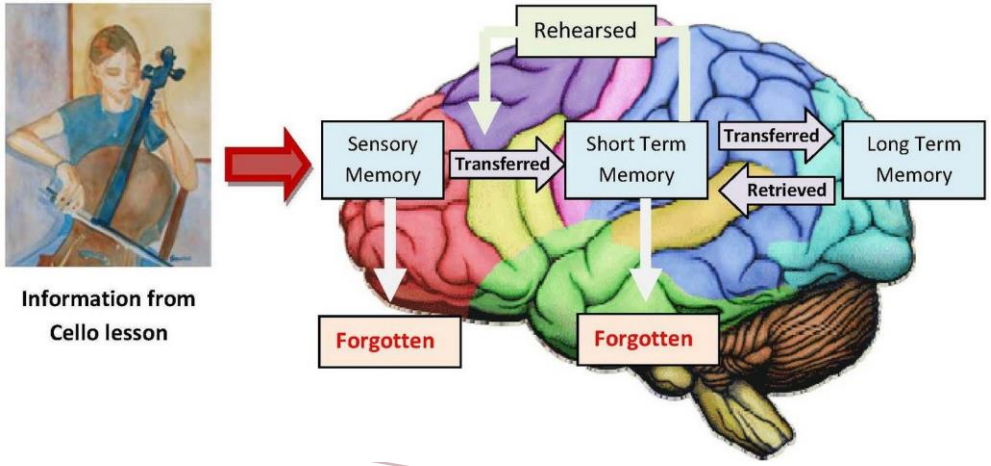




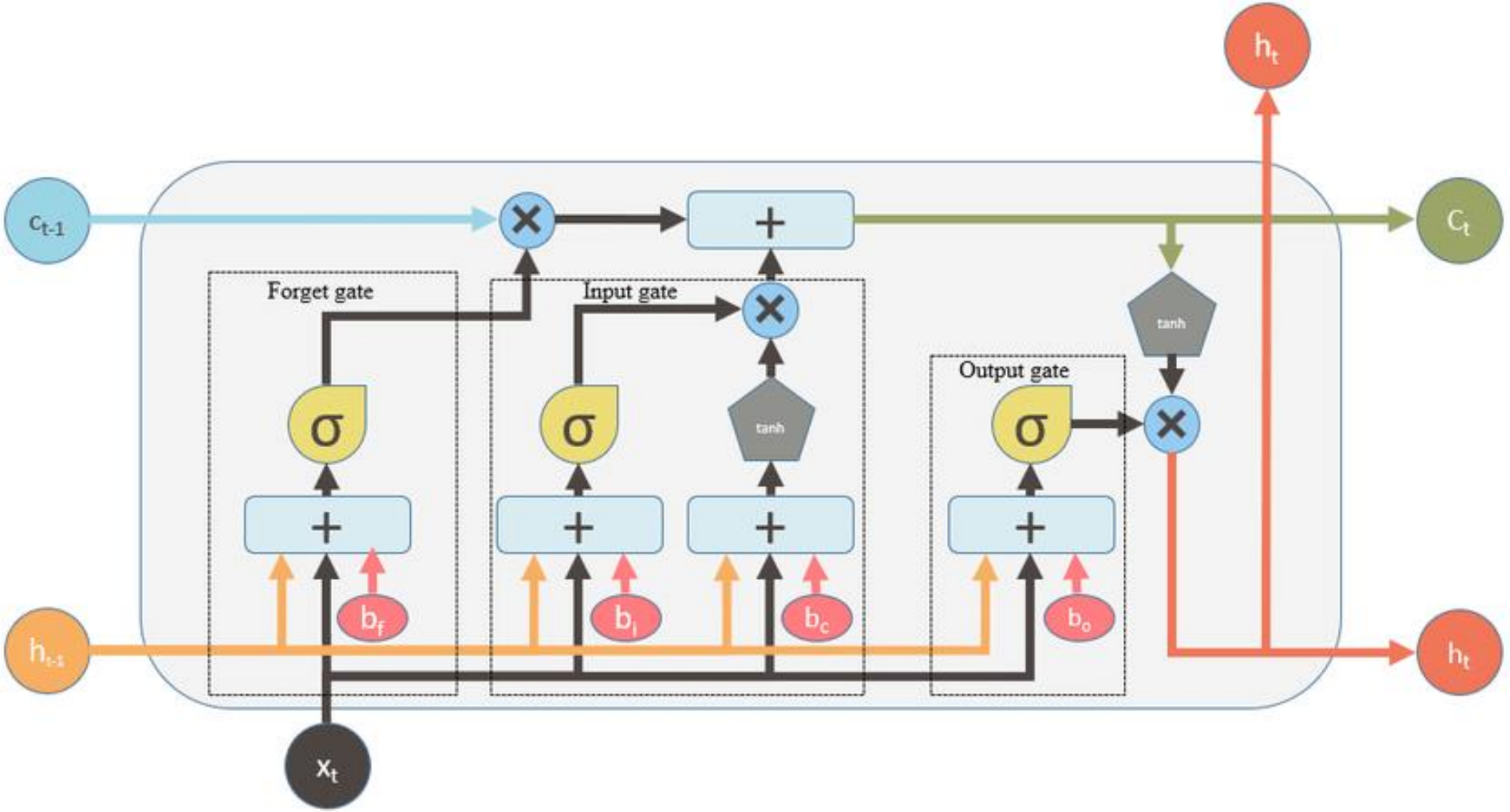
Gradient Problem Solutions







Long short-term memory (LSTM)



Long short-term memory (LSTM)

Example

https://keras.io/examples/nlp/bidirectional_lstm_imdb/

https://huggingface.co/spaces/keras-io/bidirectional_lstm_imdb

https://github.com/MRYingLEE/Stock-Price-Specific-LSTM