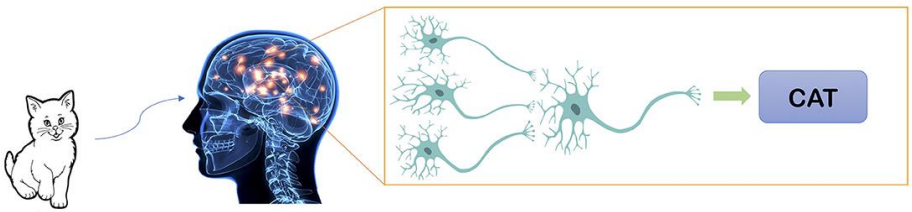
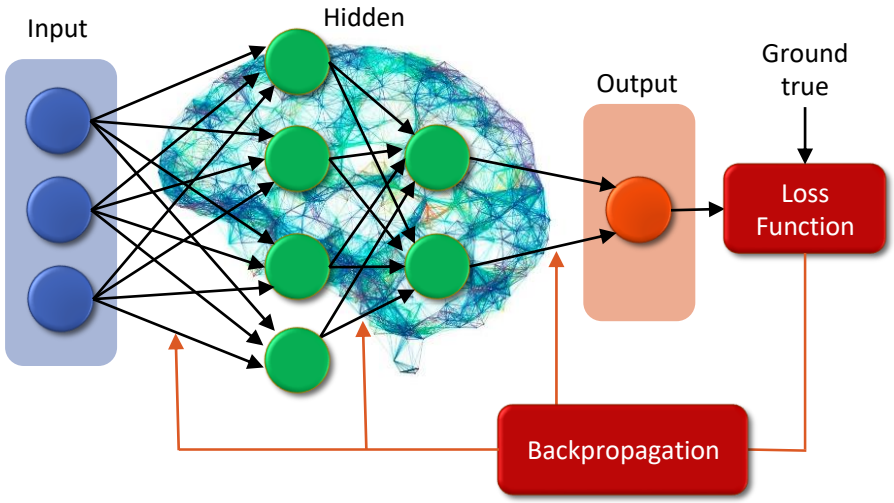
DEEP LEARNING

RECURRENT NEURAL NETWORK





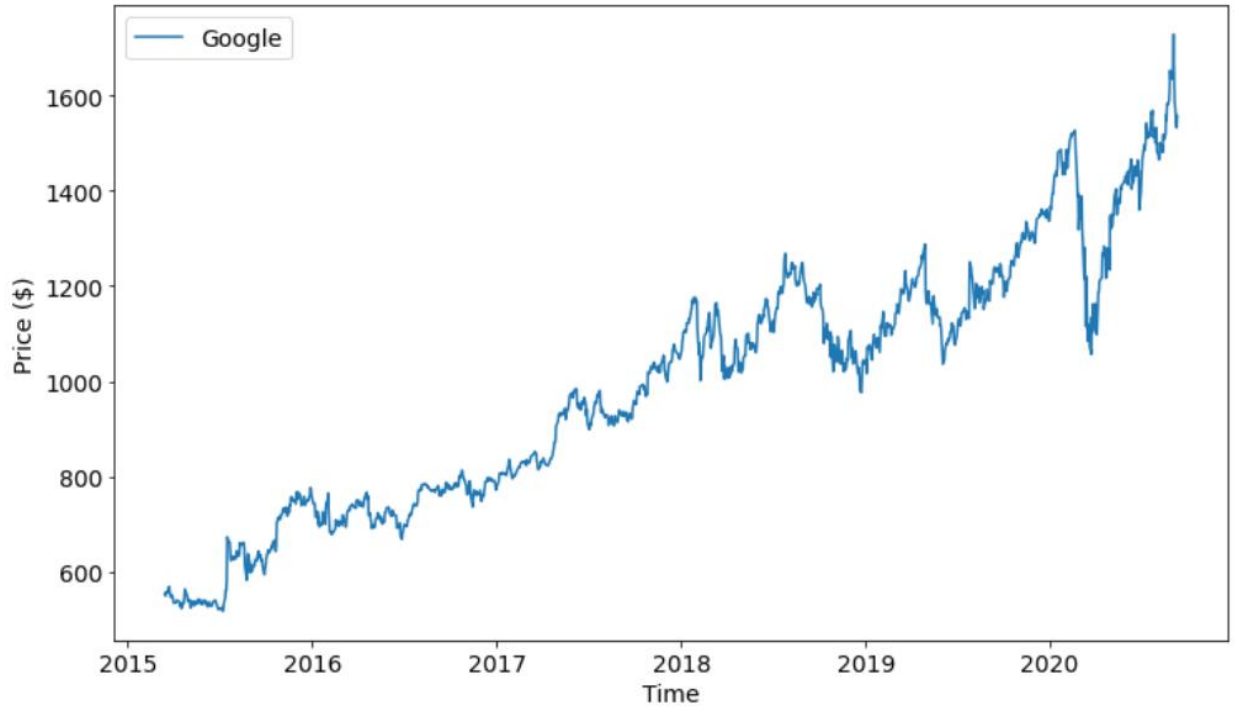
Tôn Quang Toại
Khoa Công nghệ thông tin
Trường đại học Ngoại ngữ - Tin học TP.HCM (HUFLIT)

Nội dung

- Dữ liệu trình tự (sequence data)
- RNN
  - Thuật toán lan truyền tiến
  - Thuật toán lan truyền ngược
- Các dạng RNN
- RNN trong keras
- Nhận dạng ảnh bằng RNN
- Phân loại MLP, CNN, RNN

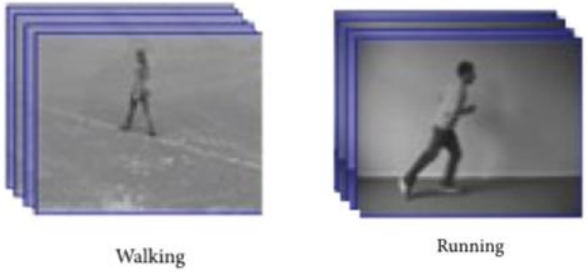
Dữ liệu trình tự

- Dữ liệu trình tự (sequence data): Dữ liệu trình tự là dữ liệu được sắp đặt có trật tự (theo thời gian, theo ràng buộc nào đó).



Dữ liệu trình tự

- Ví dụ
  - Mức tiêu thụ điện hàng giờ của một thành phố.
  - Doanh số hàng tuần của một cửa hàng.



“there is nothing new under the sun”

Dữ liệu trình tự và Bài toán

- Speech recognition

→ “After a long day of studying”

- Sentiment classification

“The pizza is pretty good” →

- Machine translation

“The pizza is pretty good” → “Pizza khá ngon”

- Name entity recognition

Sài Gòn ngày nay tập nập hơn Hà Nội → Sài Gòn ngày nay tập nập hơn Hà Nội

- Video activity recognition

→ “Walking”

Dữ liệu trình tự và Bài toán

- Nhận xét
  - Input: Sequence data
  - Output: Label, Sequence data
  - Độ dài input có thể không giống nhau
  - Độ dài input và output có thể bằng nhau hoặc không bằng nhau

Ký hiệu toán học cho dữ liệu trình tự

- Một mẫu dữ liệu trình tự

$$x = (x^{<1>}, x^{<2>}, \dots, x^{<T_x>})$$
$$y = (y^{<1>}, y^{<2>}, \dots, y^{<T_y>})$$

- Trong đó
  - $x^{<t>}$: vector đặc trưng của input
  - $y^{<t>}$: vector đặc trưng của output

- Ví dụ
  - Biểu diễn words trong NLP
  - Biểu diễn ảnh trong CV

Ký hiệu toán học cho dữ liệu trình tự

- Tập dữ liệu trình tự

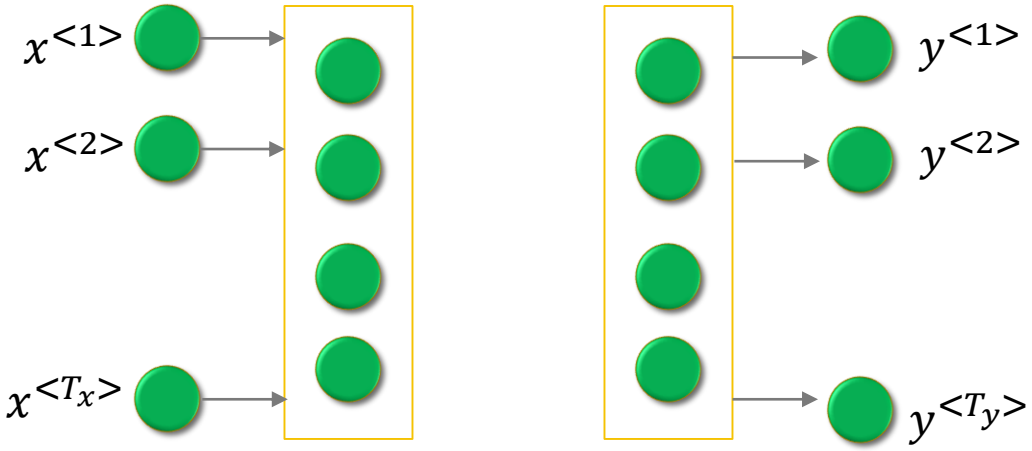
$$X = \left\{x^{(i)}\right\}_{i=1}^{m}, Y = \left\{y^{(i)}\right\}_{i=1}^{m}$$

$x^{(i)}$, có độ dài $T_x^{(i)}$

$x^{(i)}$, có độ dài $T_y^{(i)}$

$$x^{(i)} = \left(x^{(i)<1>}, x^{(i)<2>}, \ldots, x^{(i)<T_x^{(i)}>}\right)$$

MLP cho sequence data



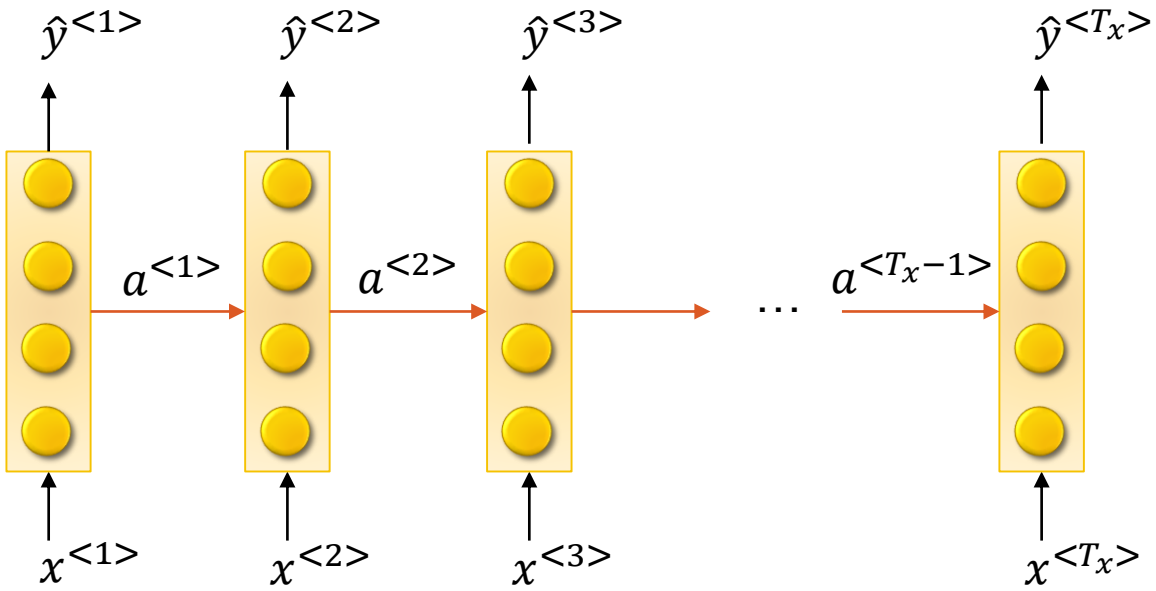
- Vấn đề
  - Input size, output size không có kích thước cố định
  - $x^{<t>}$ có kích thước lớn → số lượng tham số học lớn
  - MLP không có shared weights như trong CNN

MLP cho sequence data

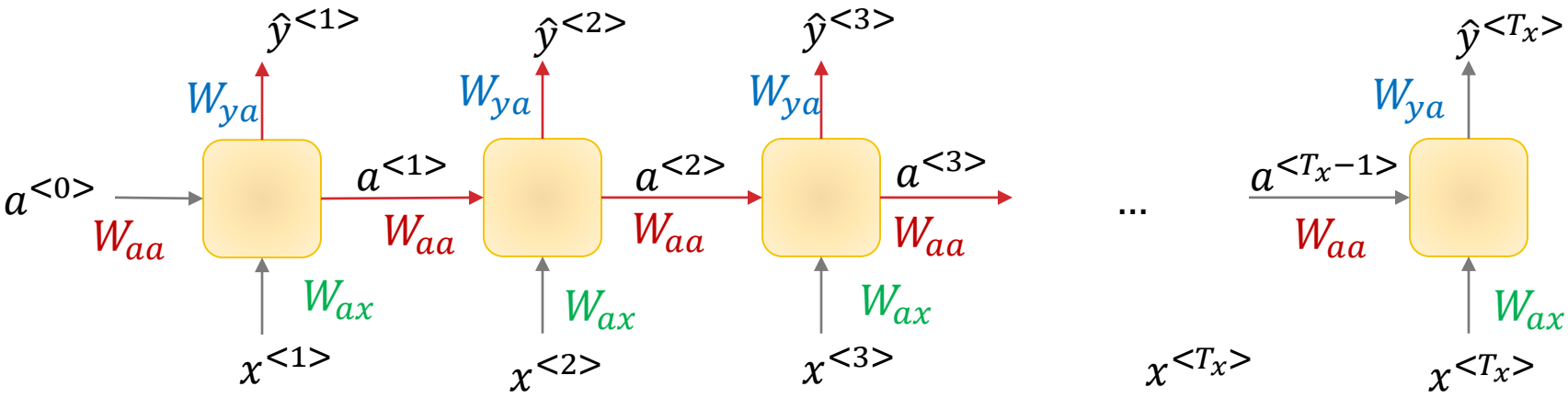
- Giải pháp
  - Input size, output size: Padding cho input và output
  - $x^{<t>}$ có kích thước lớn: Tìm mô hình ít tham số hơn
  - MLP không có shared weights: Tìm mô hình có shared weights

Mô hình Recurrent Neural Network

$$x = (x^{<1>}, x^{<2>}, \dots, x^{<T_x>})$$



Mô hình cho Dữ liệu trình tự

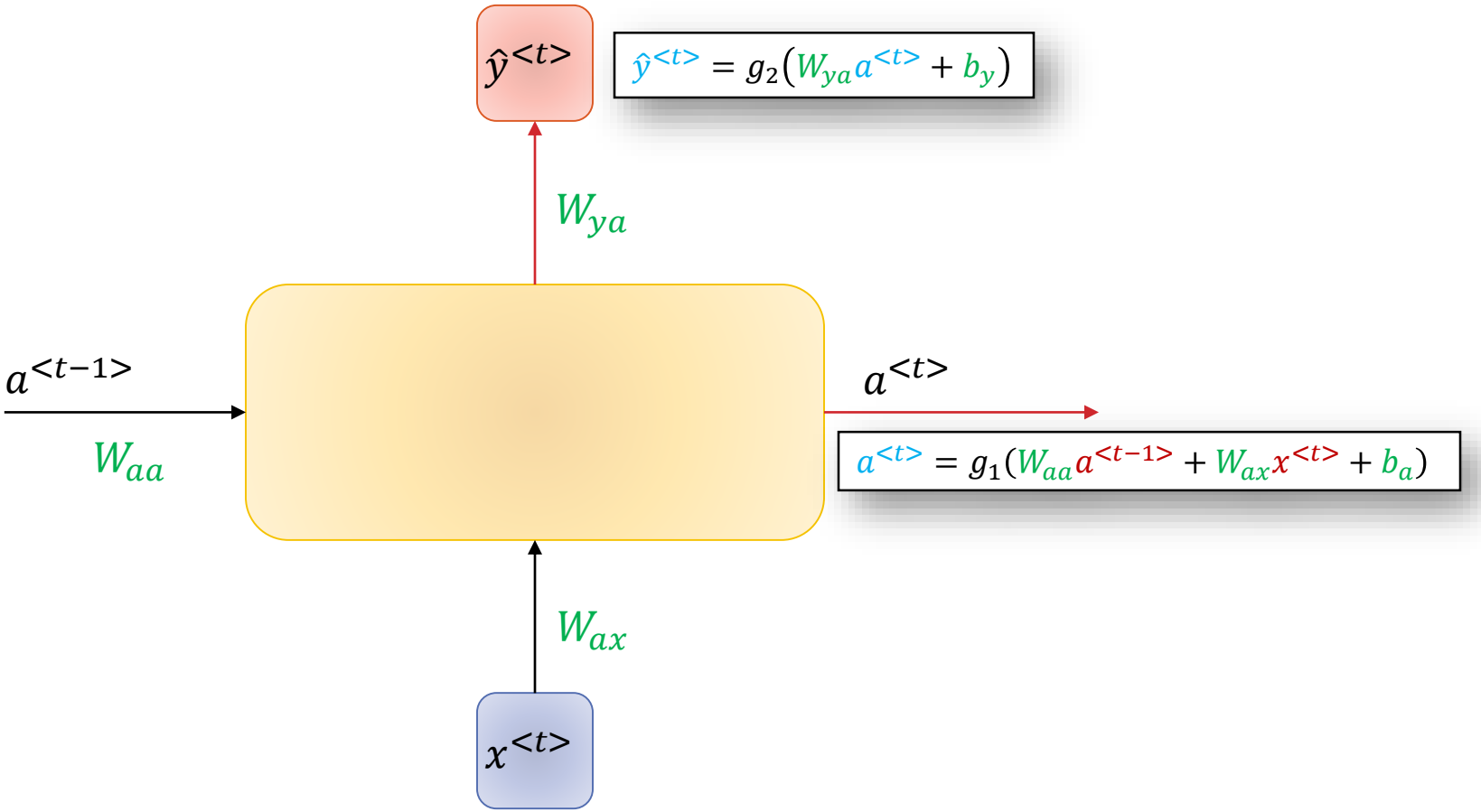


- Mỗi timestep t

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

Mô hình cho Dữ liệu trình tự



Mô hình cho Dữ liệu trình tự

- Nhận xét
  - RNN sử dụng cùng bộ trọng số (W_{aa}, W_{ax}, W_{ya}) tại mỗi timestep $t \rightarrow$ shared weights
  - RNN sử dụng thông tin của các data ở các bước trước ($t-1, t-2, \ldots$) cho data ở bước t

Thuật toán lan truyền tiến

- Thuật toán tính toán từ trái sang phải trên chuỗi

$$x = (x^{<1>}, x^{<2>}, \dots, x^{<T_x>})$$

- $a^{<0>} = \vec{0}$

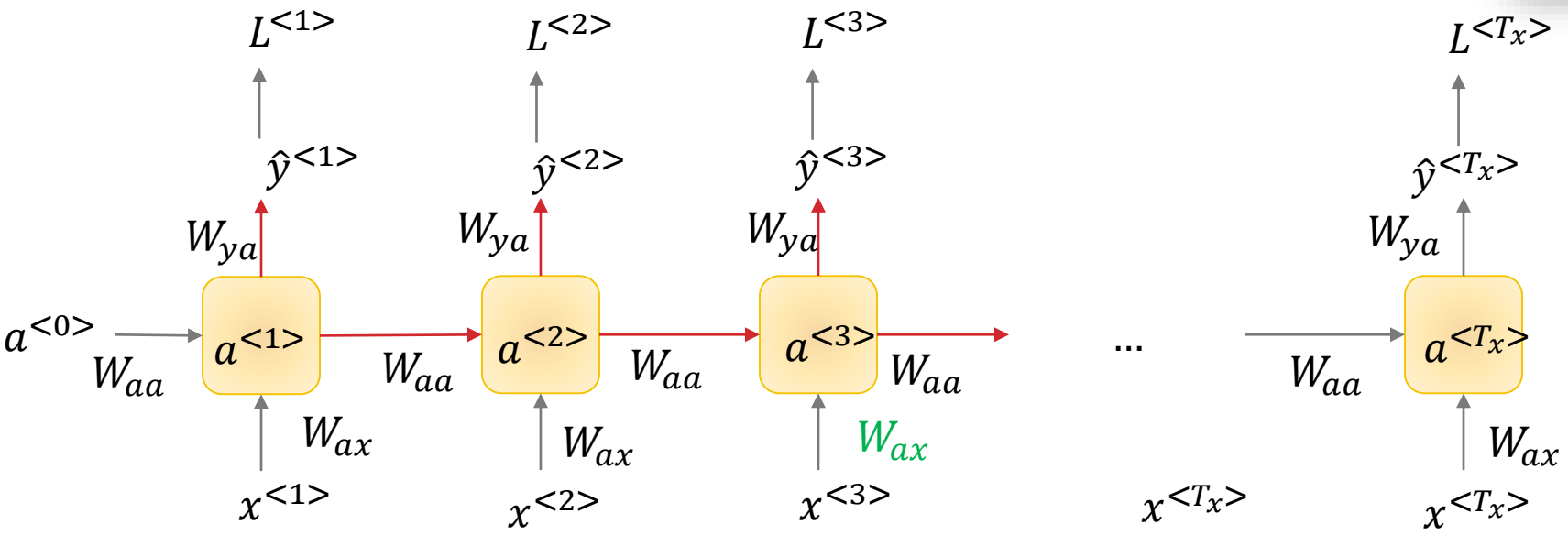
- $a^{<1>} = g_1(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a)$
- $\hat{y}^{<1>} = g_2(W_{ya}a^{<1>} + b_y)$

- $a^{<2>} = g_1(W_{aa}a^{<1>} + W_{ax}x^{<2>} + b_a)$
- $\hat{y}^{<2>} = g_2(W_{ya}a^{<2>} + b_y)$

- …

Thuật toán lan truyền ngược

$L(\hat{y}, y)$

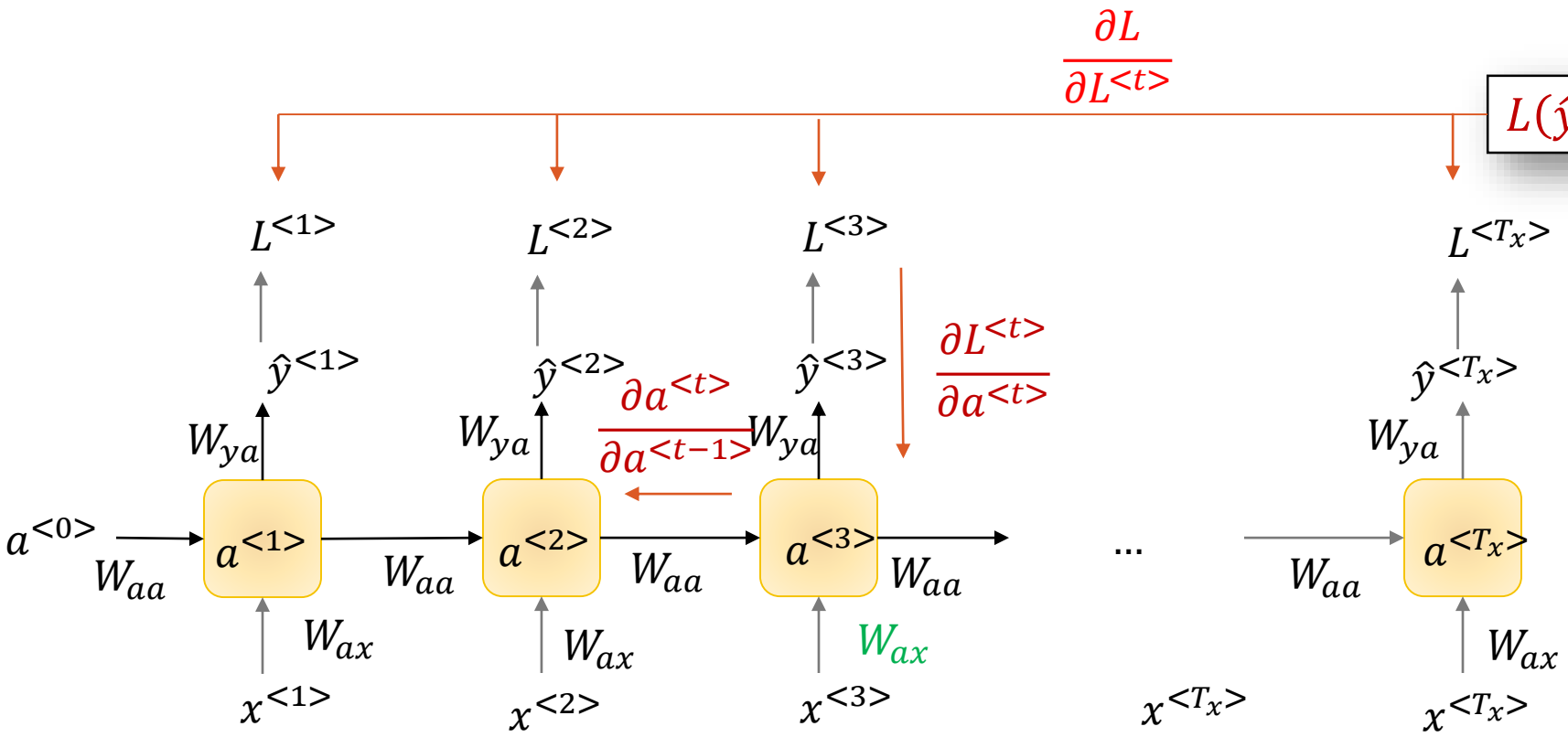


- Tính hàm loss

$$L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = y^{<t>} \log \hat{y}^{<t>} + (1 - y^{<t>})\, y^{<t>} \log(1 - \hat{y}^{<t>})$$

$$L(\hat{y}, y) = -\sum_{t=1}^{T_y} L^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Thuật toán lan truyền ngược

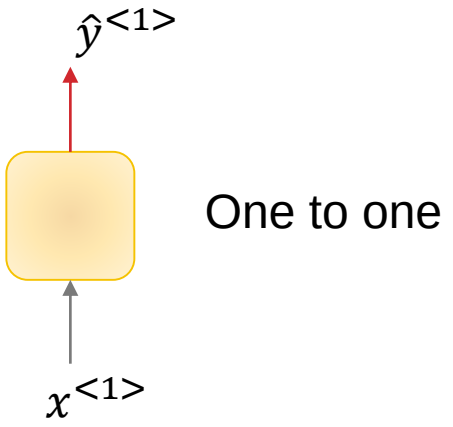


$$\frac{\partial L}{\partial L^{<t>}} \qquad \frac{\partial L^{<t>}}{\partial a^{<t>}} \qquad \frac{\partial a^{<t>}}{\partial a^{<t-1>}}$$

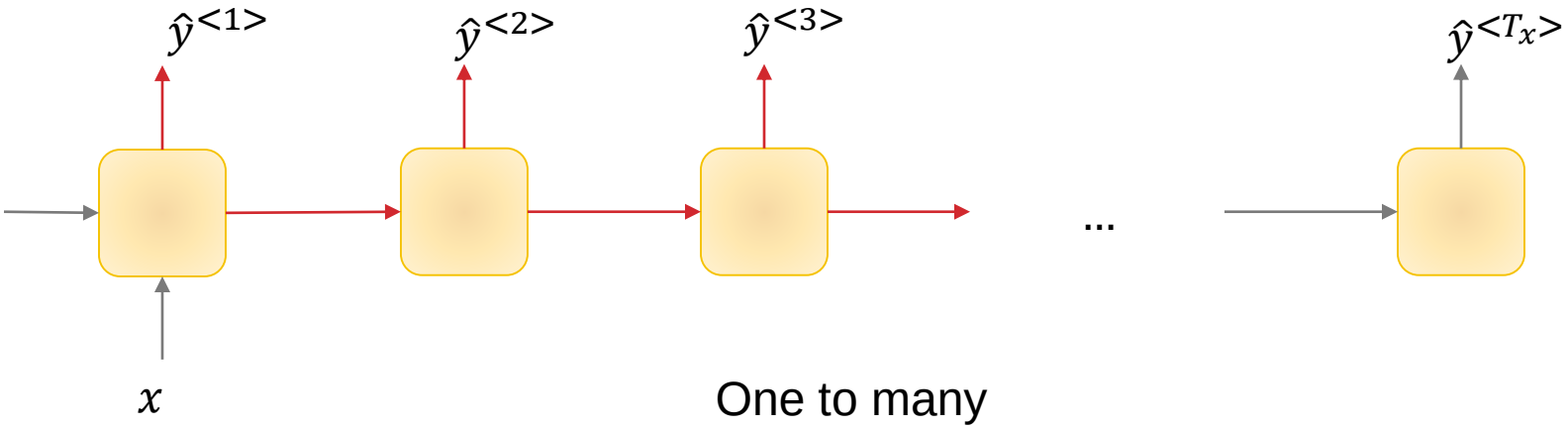
Một số kiến trúc RNN

- One to one
- One to many
- Many to one
- Many to many
  - Cùng chiều dài
  - Khác chiều dài

Một số kiến trúc RNN

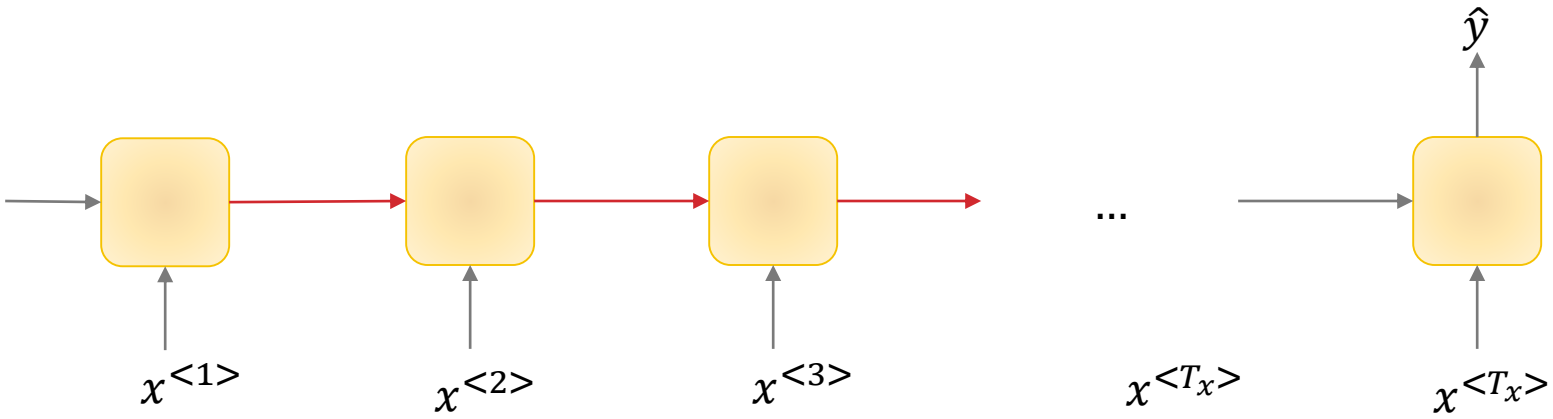


One to one



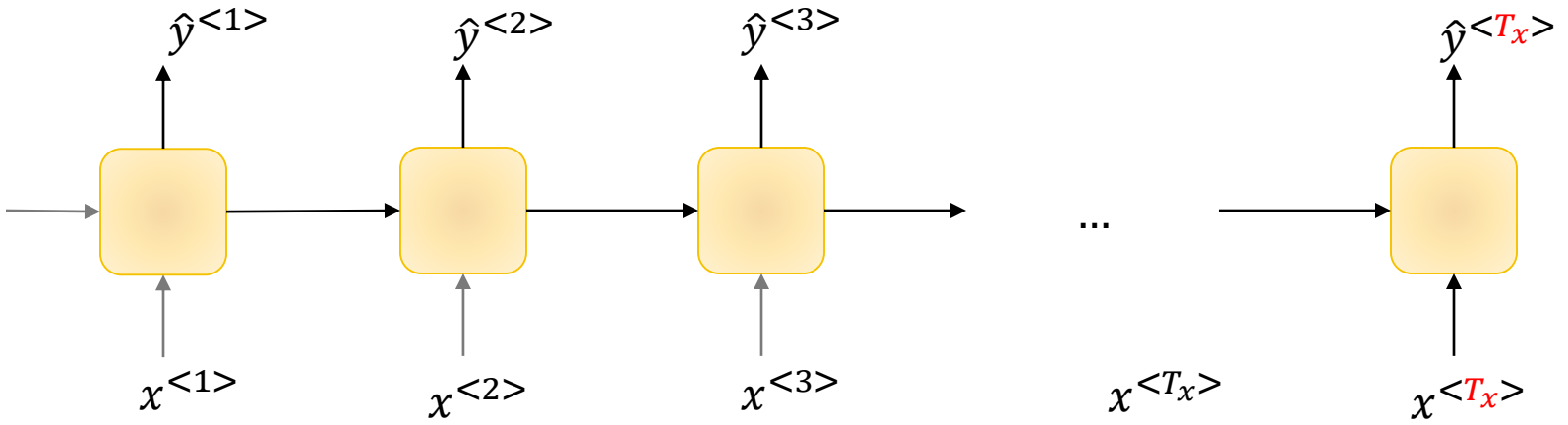
One to many

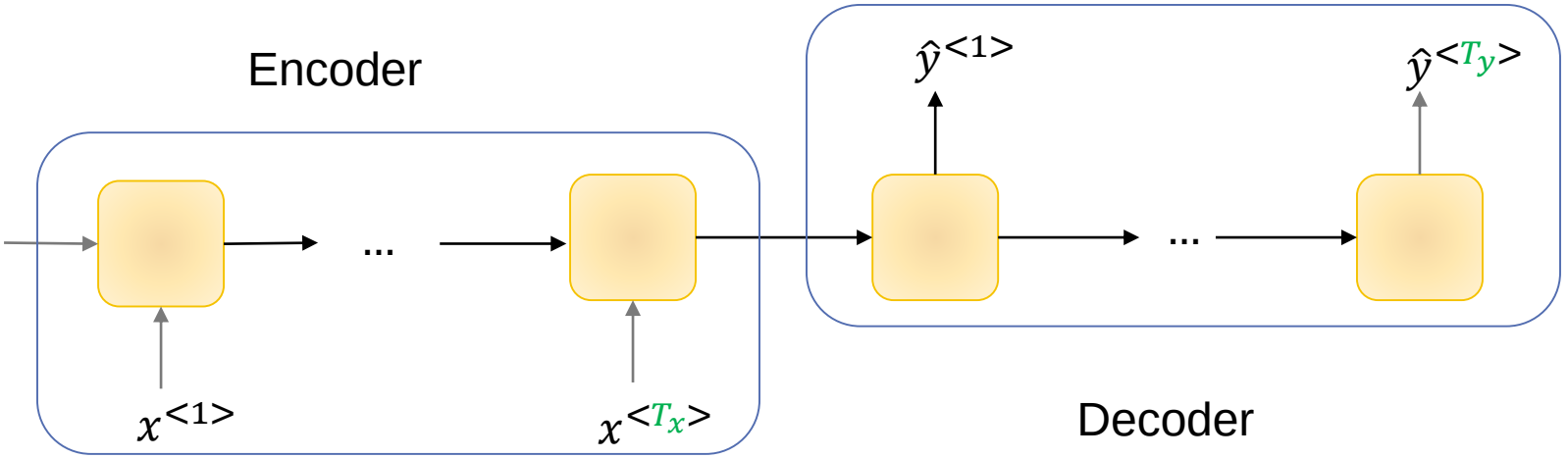
Một số kiến trúc RNN



Many to one

Một số kiến trúc RNN





Many to many

RNN trong Keras

```
tf.keras.layers.SimpleRNN(
    units,
    activation='tanh',
    use_bias=True,
    kernel_initializer='glorot_uniform',
    recurrent_initializer='orthogonal',
    bias_initializer='zeros',
    kernel_regularizer=None,
    recurrent_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    recurrent_constraint=None,
    bias_constraint=None,
    dropout=0.0,
    recurrent_dropout=0.0,
    return_sequences=False,
    return_state=False,
    go_backwards=False,
    stateful=False,
    unroll=False
)
```

RNN trong Keras

- Single output

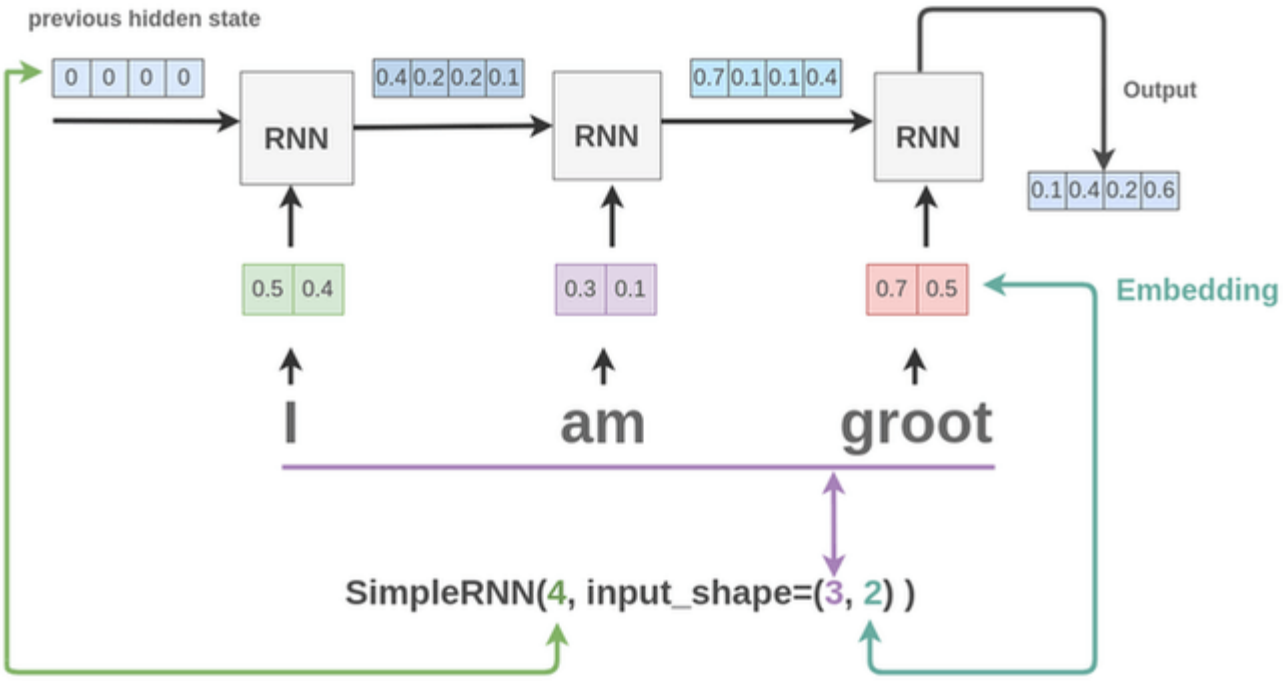
I am groot

- Chuyển cầu thành số
  - one-hot encoding
  - pretrained word vectors
  - Học word embeddings từ đầu

Word	E1	E2
I	0.5	0.4
am	0.3	0.1
groot	0.7	0.5

RNN trong Keras

```
model = Sequential()
model.add(SimpleRNN(4, input_shape=(3, 2)))
```



RNN trong Keras

- input_shape=(3, 2)
  - 3 từ → time-steps là 3 → RNN block sẽ unfold 3 lần
  - Mỗi từ có word embedding là 2

- SimpleRNN(4, …)
  - Có 4 units trong tầng hidden
  - Trạng thái ẩn có kích thước bằng 4 được truyền qua giữa các RNN block
  - Block đầu tiên, trạng thái ẩn là [0,0,0,0]

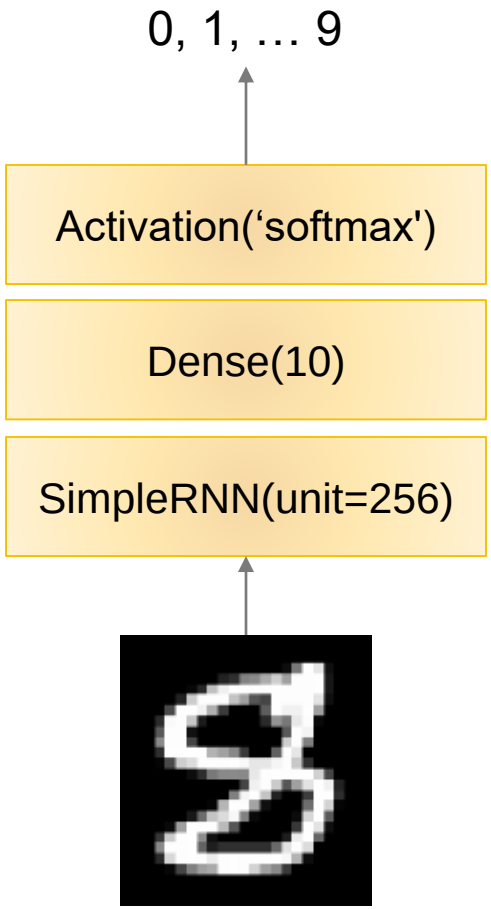
RNN trong Keras

```
import tensorflow as tf
from tensorflow.keras.layers import SimpleRNN

x = tf.random.normal((1, 3, 2))

layer = SimpleRNN(4, input_shape=(3, 2))
output = layer(x)

print(output.shape)
# (1, 4)
```

```
model = Sequential()
model.add(SimpleRNN(4, input_shape=(3, 2)))
model.add(Dense(1))
```

NHẬN DẠNG ẢNH VỚI RNN

Mô hình



Chương trình

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, SimpleRNN, Activation
import matplotlib.pyplot as plt

import numpy as np

# Load mnist dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
print(x_train.shape)

num_labels = len(np.unique(y_train))
image_size = x_train.shape[1]

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

x_train = x_train.astype("float")/255
x_test = x_test.astype("float")/255
```

Chương trình

```
# Network
input_shape = (image_size, image_size)
batch_size = 128;
units = 256
dropout = 0.2

model = Sequential()
model.add(SimpleRNN(units = units, dropout=dropout,
input_shape=input_shape))
model.add(Dense(num_labels))
model.add(Activation("softmax"))

print(model.summary())
plot_model(model, to_file='rnn_mnist.png', show_shapes=True)
```

Chương trình

```
# Training
model.compile(optimizer='sgd', loss='categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit(x_train, y_train, epochs=4, batch_size=batch_size,
validation_split=0.2,)
model.save('rnn-minst')


# Test
_, acc = model.evaluate(x_test, y_test, batch_size=batch_size)
print(acc*100)
```

Chương trình

```
# Plot history
print(history.history.keys())

# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

MLP, CNN VÀ RNN

3 lớp kiến trúc ANN

- Multilayer Perceptrons (MLPs)
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)

Khi nào dùng MLP

- Sử dụng MLP cho
  - Dataset dạng table
  - Bài toán Classification
  - Bài toán Regresstion
- Hãy thử MLP với các dữ liệu khác để tạo baseline model
  - Image
  - Text
  - Sequence data

Khi nào dùng CNN

- Sử dụng CNN cho
  - Dataset dạng Image (dữ liệu có quan hệ không gian)
  - Bài toán Classification
  - Bài toán Regresstion
  - Bài toán Generative

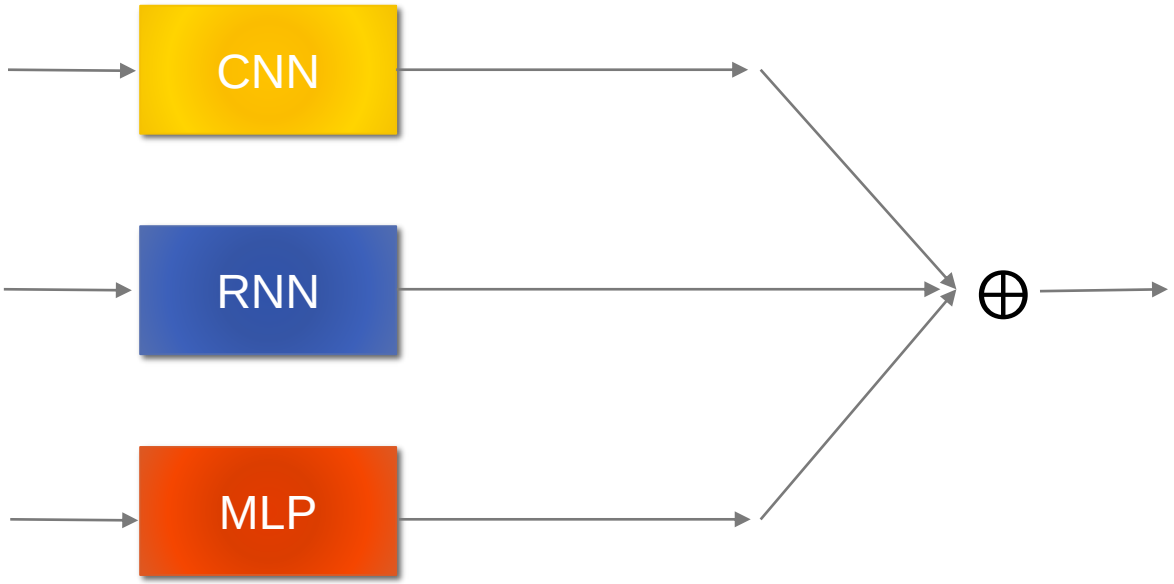
- Hãy thử CNN với các dữ liệu khác
  - Text
  - Sequence data

Khi nào dùng RNN

- Sử dụng RNN cho
  - Sequence data (dữ liệu có quan hệ tuần tự)
  - Bài toán Classification
  - Bài toán Regresstion
  - Bài toán Generative

- Hãy thử RNN với dữ liệu khác
  - Image

Mô hình kết hợp (Hybrid Network Models)



Mô hình kết hợp (Hybrid Network Models)

- Ví dụ

```
cnn = Sequential()
cnn.add(Conv2D(...)))
cnn.add(MaxPooling2D(...))
cnn.add(Flatten())
```

```
rnn = cnn = Sequential()
rnn.add(LSTM(...))
rnn.add(Dense(...))
```

```
cnn = Sequential()
cnn.add(Conv2D(...))
cnn.add(MaxPooling2D(...))
cnn.add(Flatten())

model = Sequential()
model.add(TimeDistributed(cnn, ...))
model.add(LSTM(..))

model.add(Dense(...))
```