Advanced Computer Science Course Lecture 2

Tishreen University

Computer and automatic control engineering dept.

Master Program- 2024

1st year

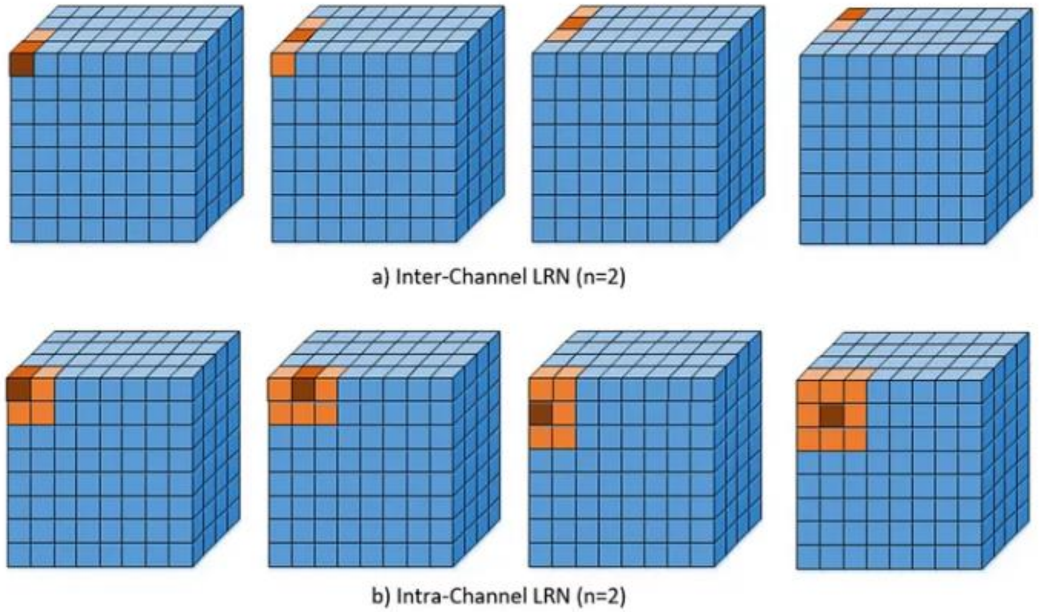
Dr. Ali Mahmoud Mayya

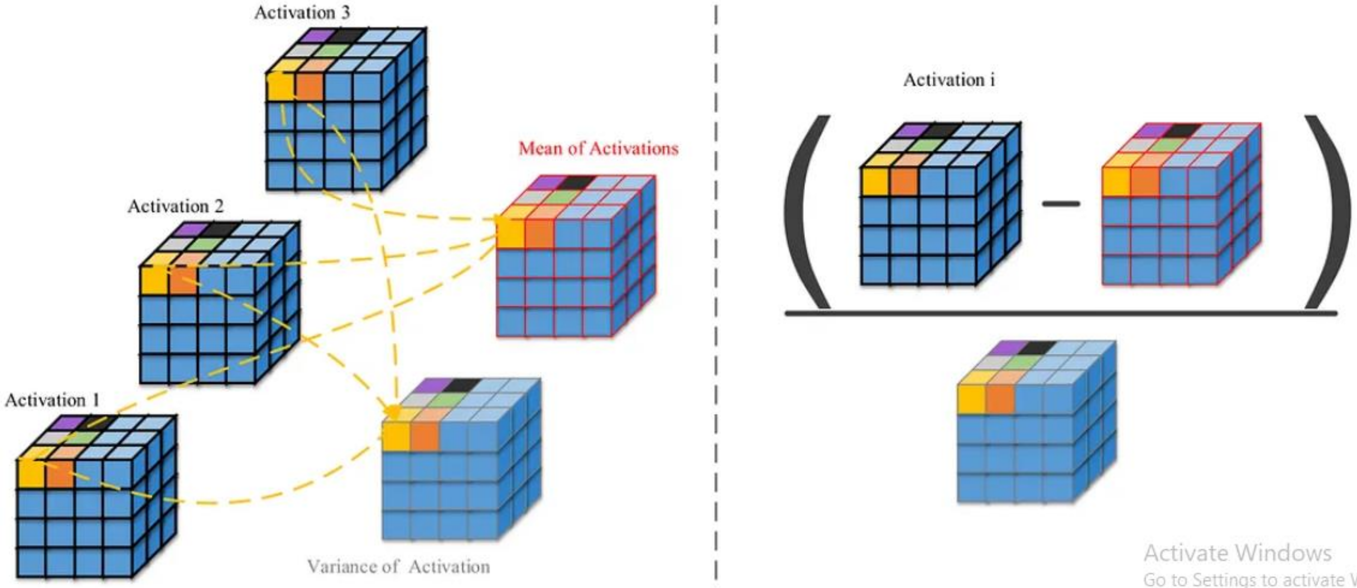
Batch normalization Vs. Local response normalization

improve the performance and stability of neural networks

Local Response Normalization (LRN):
Used after Relu function (the output layers are not constrained within a bounded range (such as [-1,1] for *tanh*), rather they can grow as high as the training allows it).
Use to improve the ability of neuron to reduce the activity of its neighbors ***(Lateral inhibition).***
Normalize neurons at the same batch (either in depth (maps) or in the same activation map).
Not trainable

Batch Normalization (BN):
Used after convolutional layers and fully connected layers.
Normalize neurons from different training batches.
BN normalizes gradients, making them less sensitive to vanishing/exploding gradient problems
Trainable (scale & Shift)





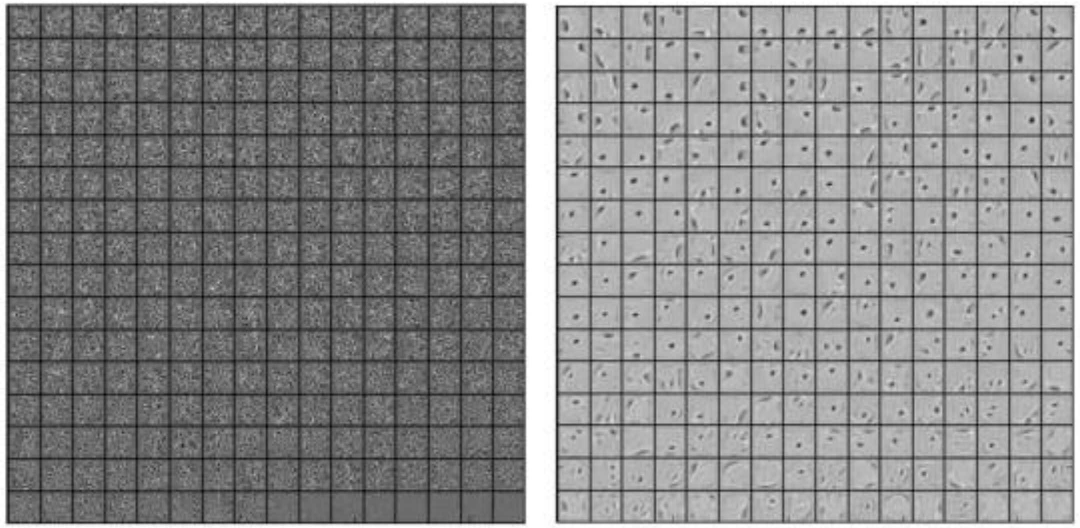
Dropout Layers

Reduce linearity/ reduce overfitting

“dropout” refers to dropping out the nodes (input and hidden layer) in a neural network.

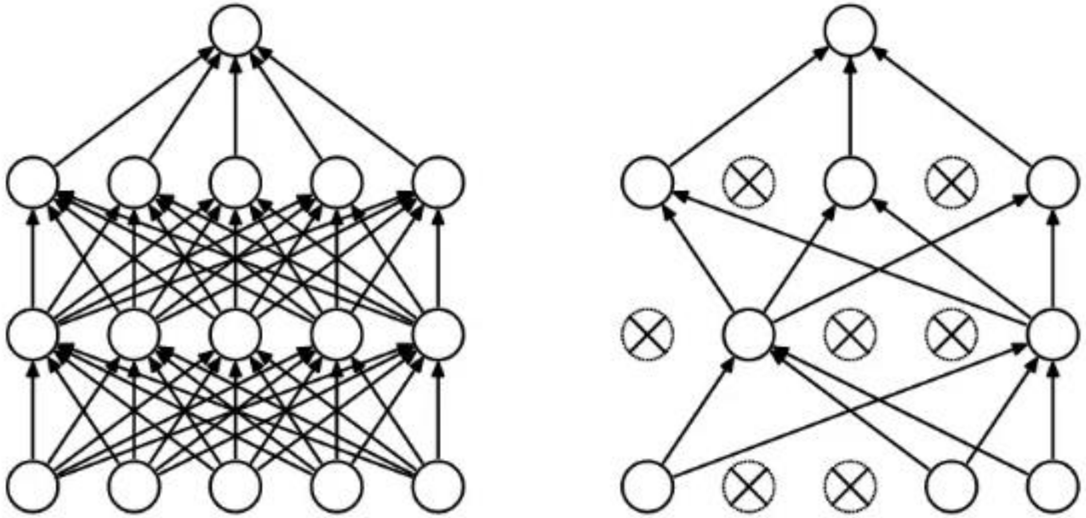
All the forward and backwards connections with a dropped node are temporarily removed, thus creating a new network architecture out of the parent network.

The nodes are dropped by a dropout probability of p.



(a) Without dropout (b) Dropout with $p = 0.5$.

Figure 2: (a) Hidden layer features without dropout; (b) Hidden layer features with dropout (Image by Nitish)



(a) Standard Neural Net (b) After applying dropout.

https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9

Global average pooling Layers

Fully-connected layers

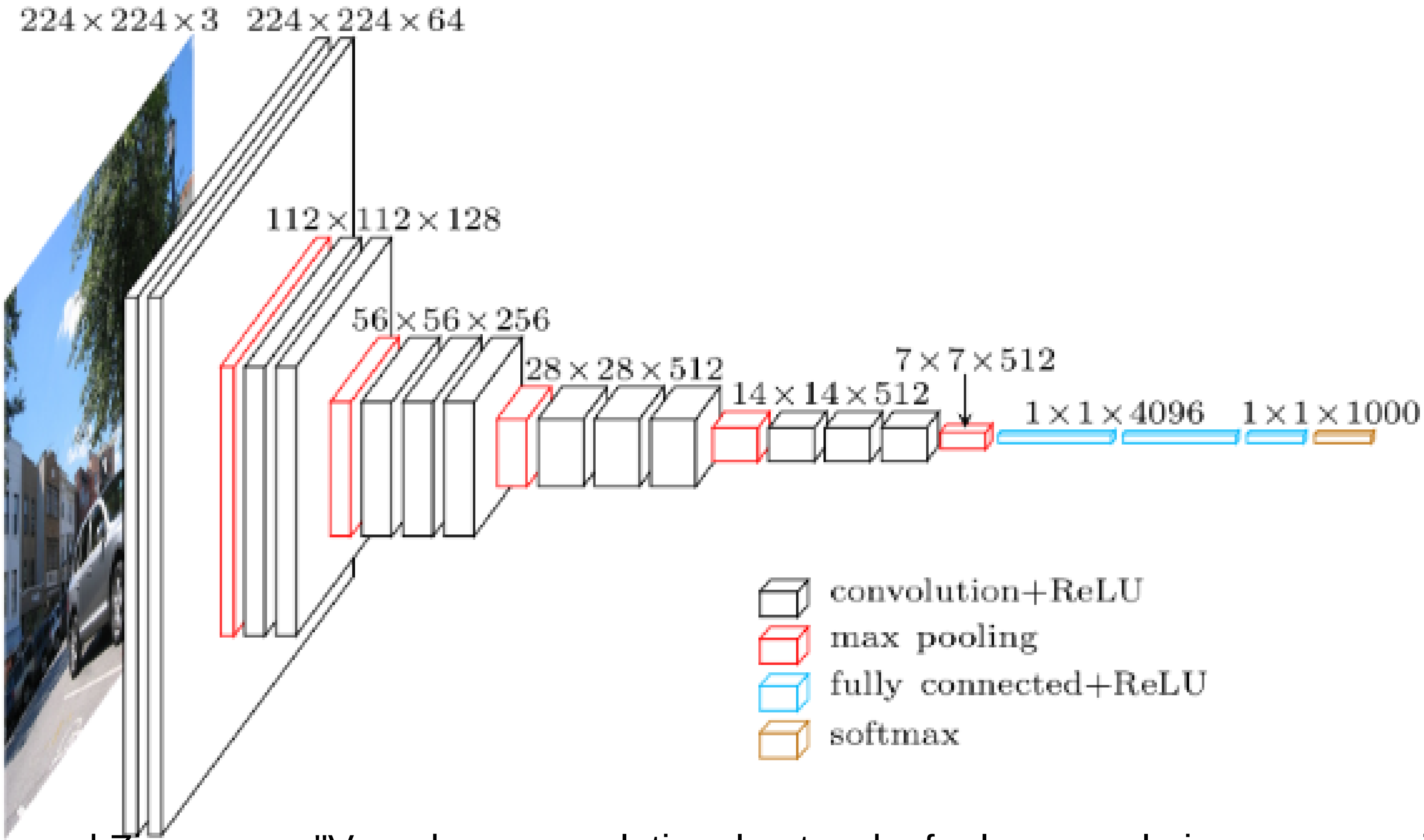
Global Average Pooling replaces fully connected layers in classical CNNs.

It is an operation that ***calculates the average output of each feature map in the previous layer***.

Most famous DL models

VGG16

-



Simonyan, Karen, and Zisserman. "Very deep convolutional networks for large-scale image recognition." (2014)

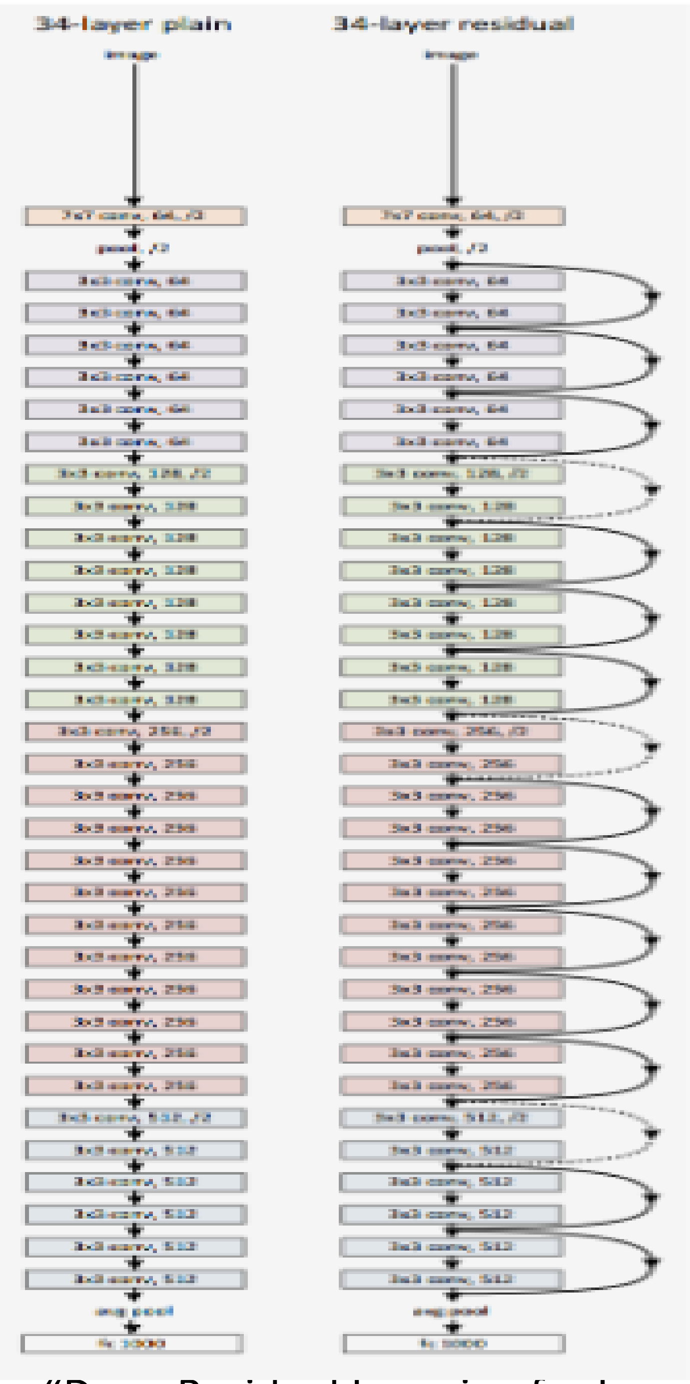
Most famous DL models

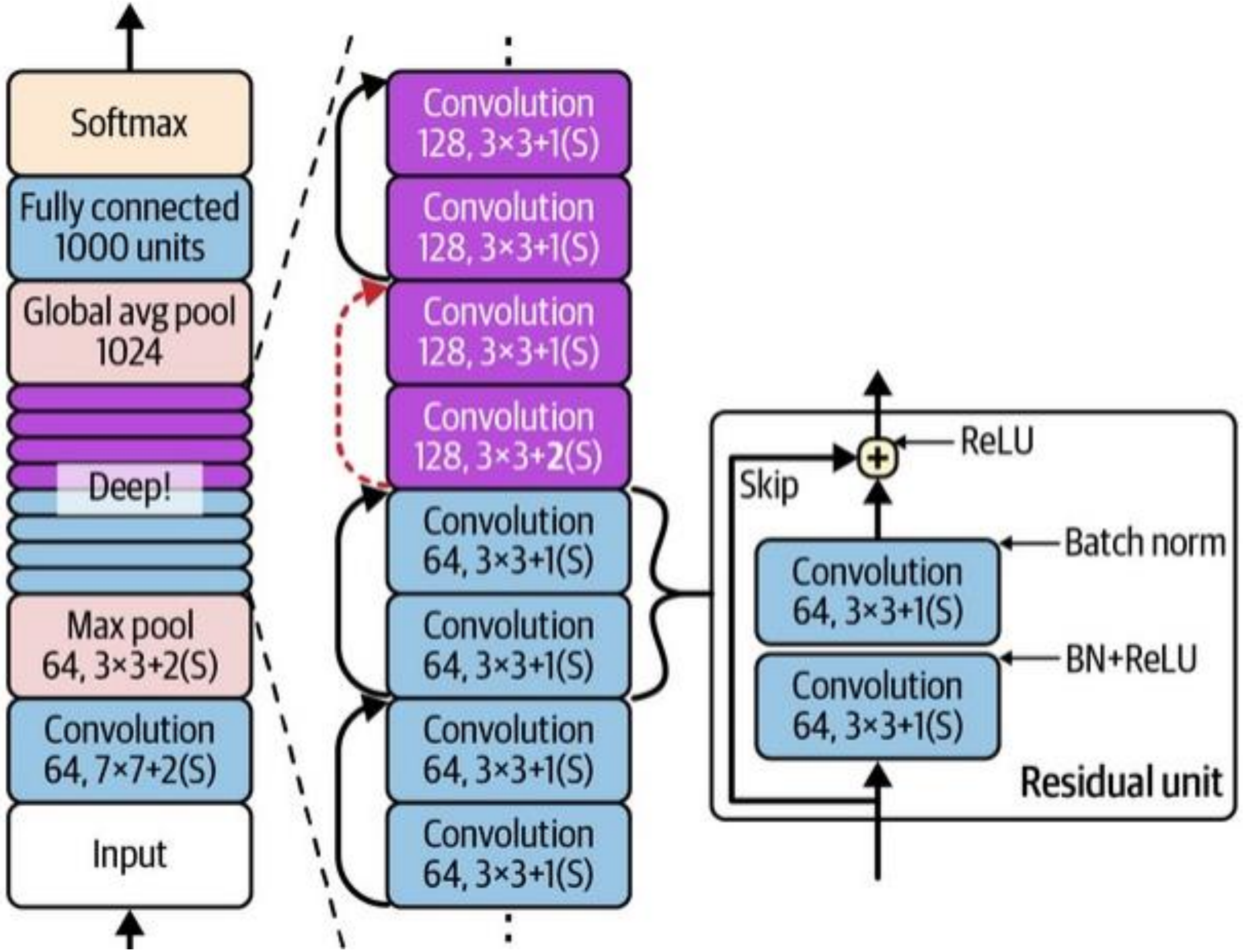
VGG16 in Keras

```
model.add(Convolution2D(64, 3, 3,
activation='relu',input_shape=(3,224,224)))
model.add(Convolution2D(64, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(Convolution2D(128, 3, 3, activation='relu'))
model.add(Convolution2D(128, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(Convolution2D(256, 3, 3, activation='relu'))
model.add(Convolution2D(256, 3, 3, activation='relu'))
model.add(Convolution2D(256, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1000, activation='softmax'))
```

Most famous DL models

ResNets

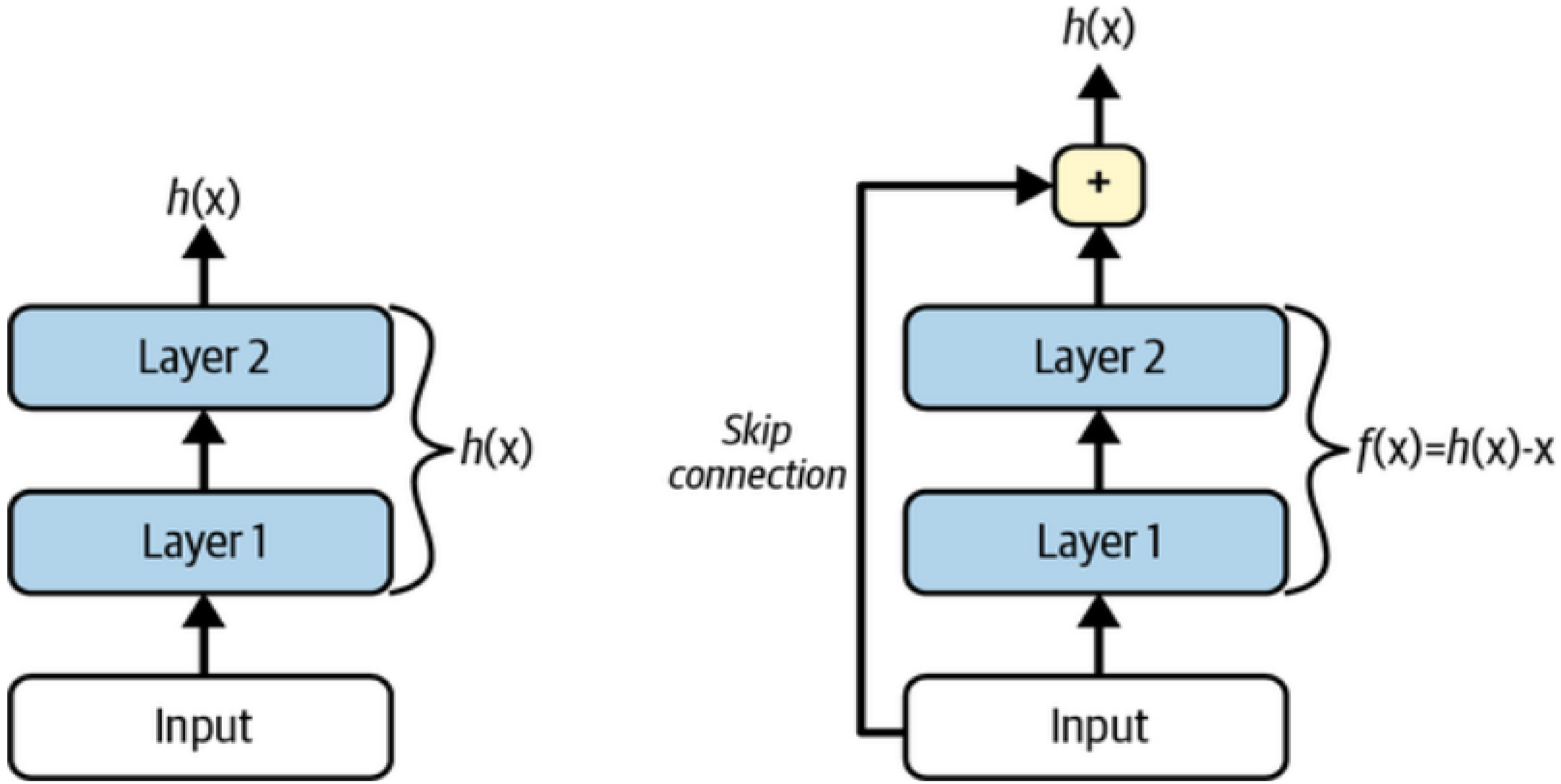




“Deep Residual Learning for Image Recognition” K. He

Most famous DL models

ResNets



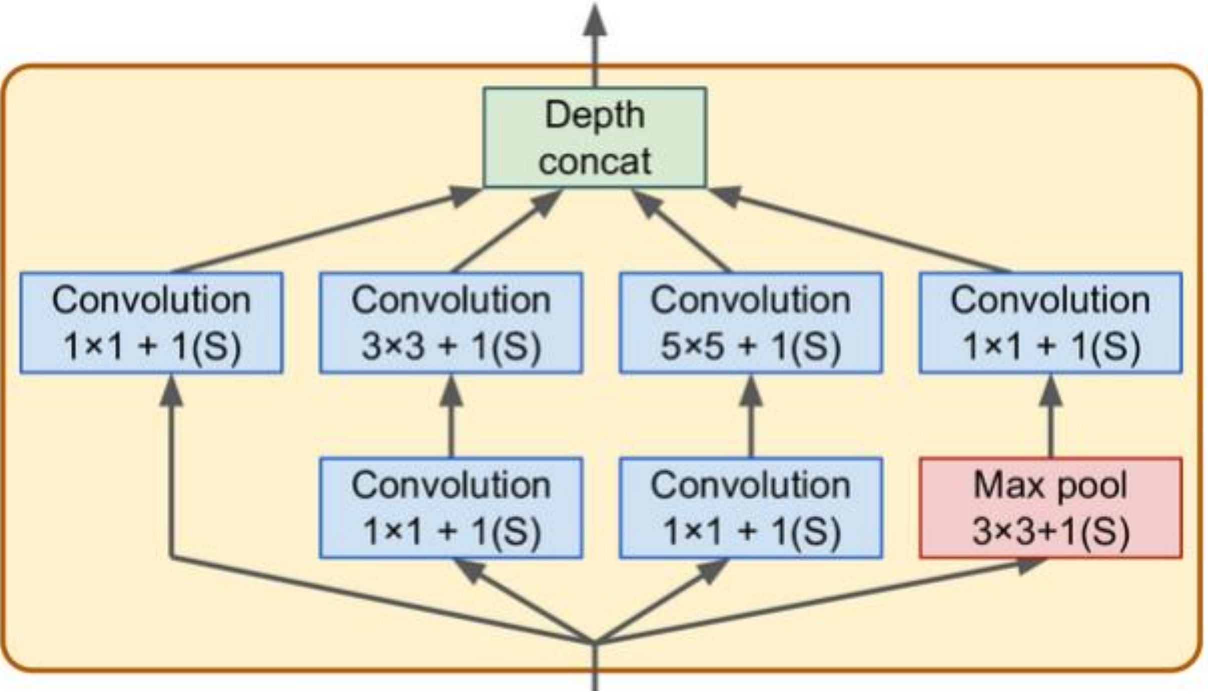
solve the vanishing gradient problem in very deep networks

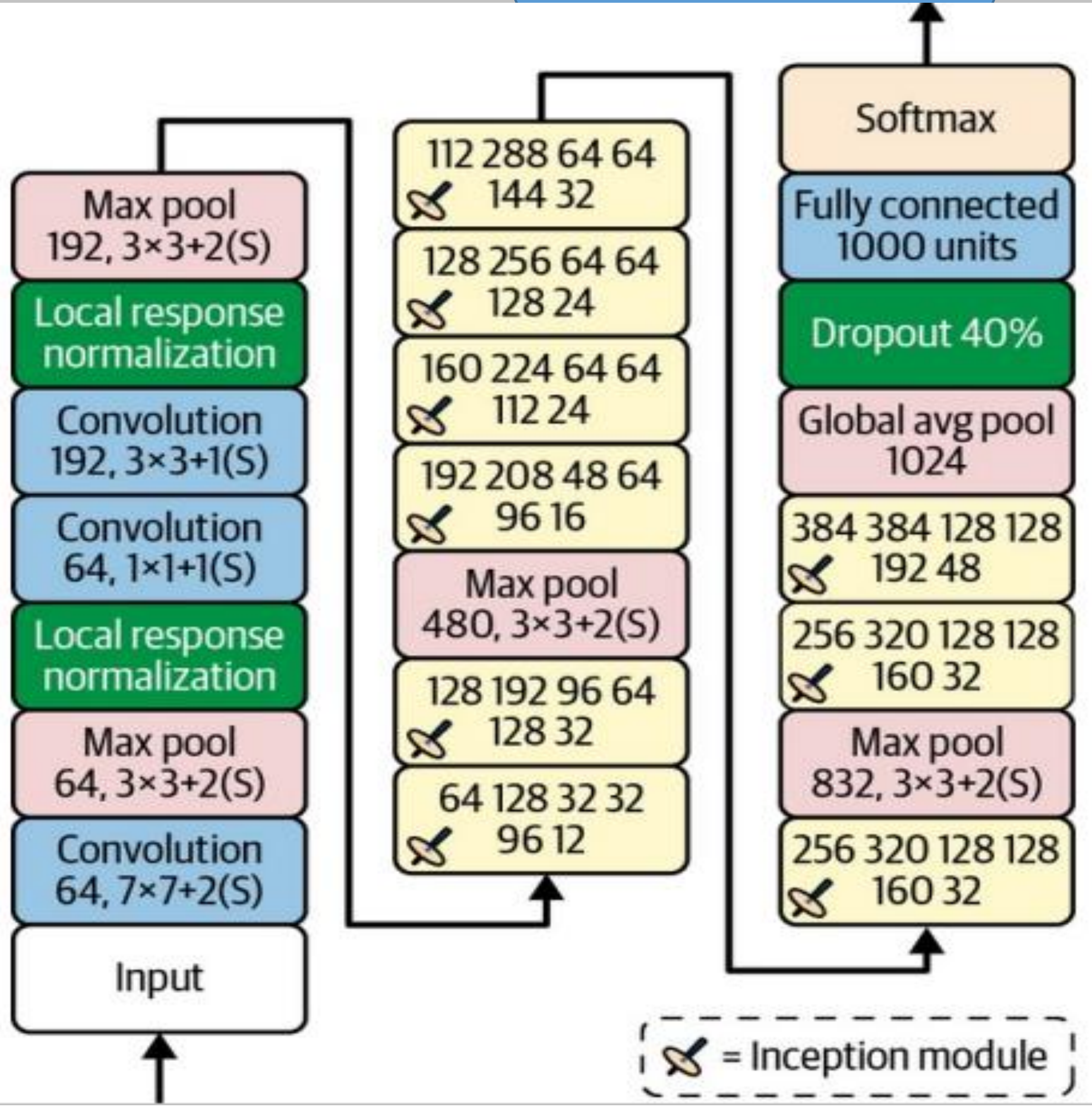
ResNet50 Compared to VGG:
Superior accuracy in all vision tasks
5.25% top-5 error vs 7.1%
Less parameters **25M** vs 138M
Computational complexity **3.8B Flops** vs 15.3B Flops

"Deep Residual Learning for Image Recognition" K. He

Most famous DL models

GoogleNet



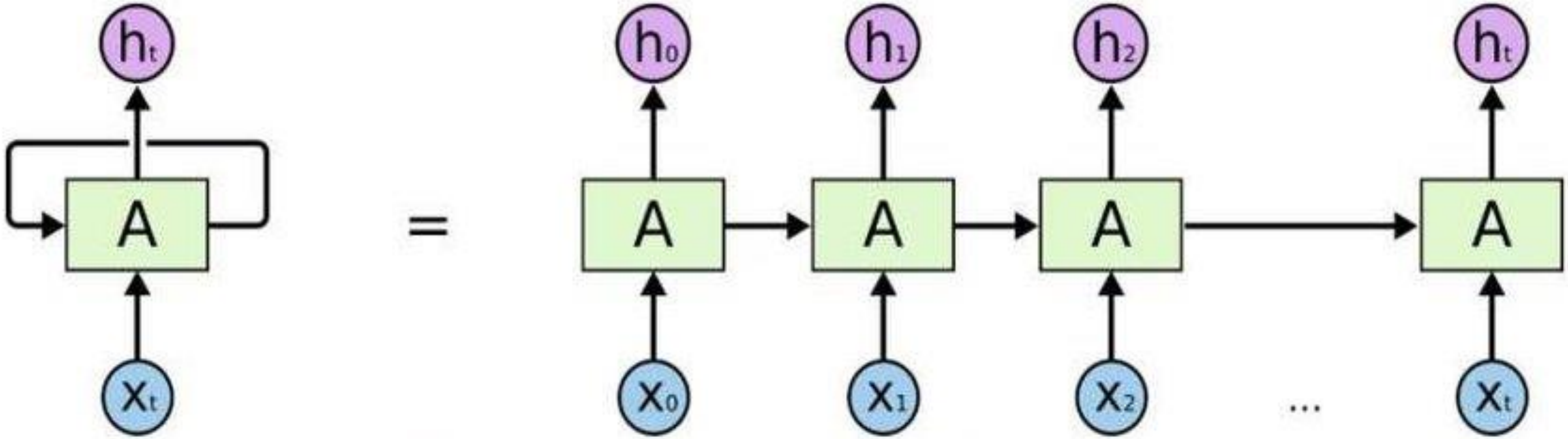


ResNet50 Compared to VGG:
Superior accuracy in all vision tasks
5.25% top-5 error vs 7.1%
Less parameters **25M** vs 138M
Computational complexity **3.8B Flops** vs 15.3B Flops

“Deep Residual Learning for Image Recognition” K. He

Language DL Models

RNN

- 

Time is essential in many natural language processing applications (sentence creation, Language translation, descriptions, etc.)

Language DL Models

RNN

-

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t; \Theta)$$

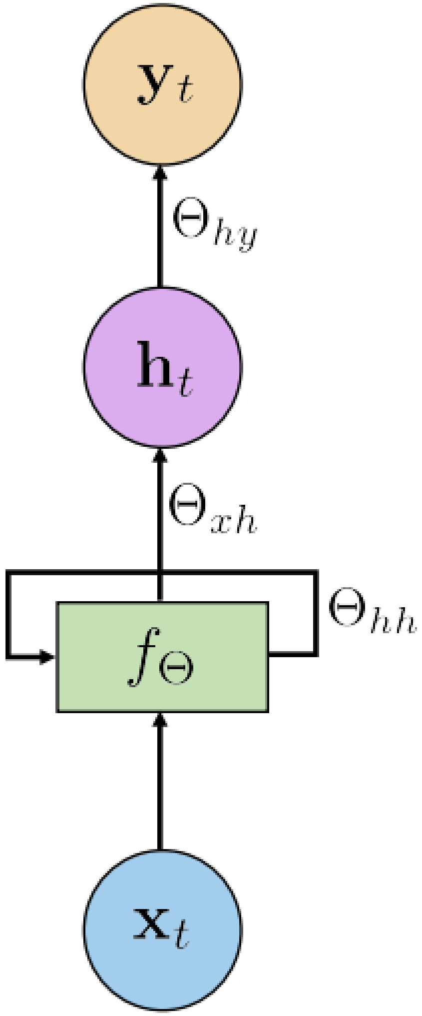
$$\downarrow$$

$$\mathbf{h}_t = \tanh(\Theta_{hh}\mathbf{h}_{t-1} + \Theta_{xh}\mathbf{x}_t)$$

$$\downarrow$$

$$\mathbf{y}_t = \Theta_{hy}\mathbf{h}_t$$

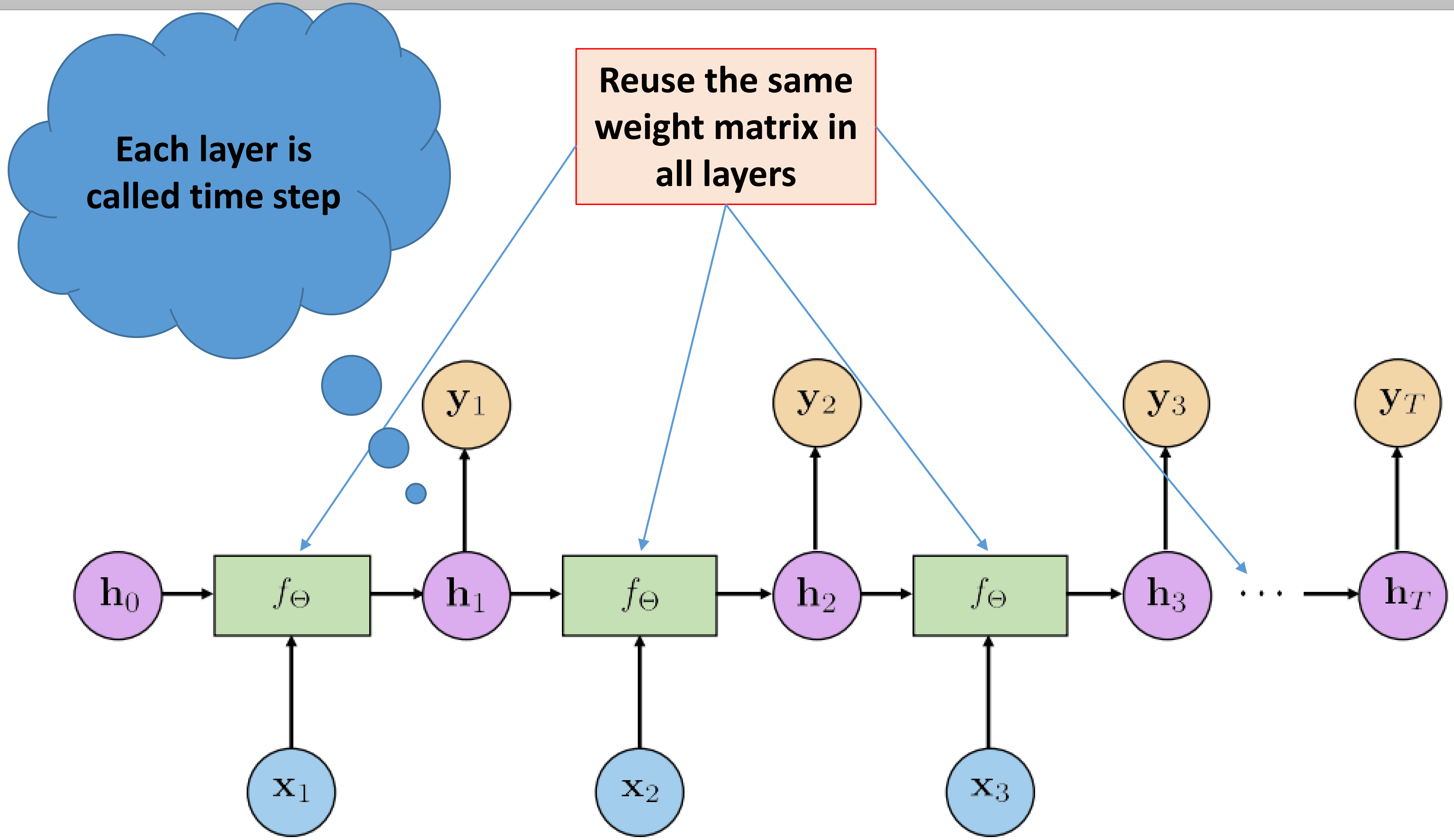
Use Tanh because its derivative doesn't vanishing through time



Language DL Models

RNN

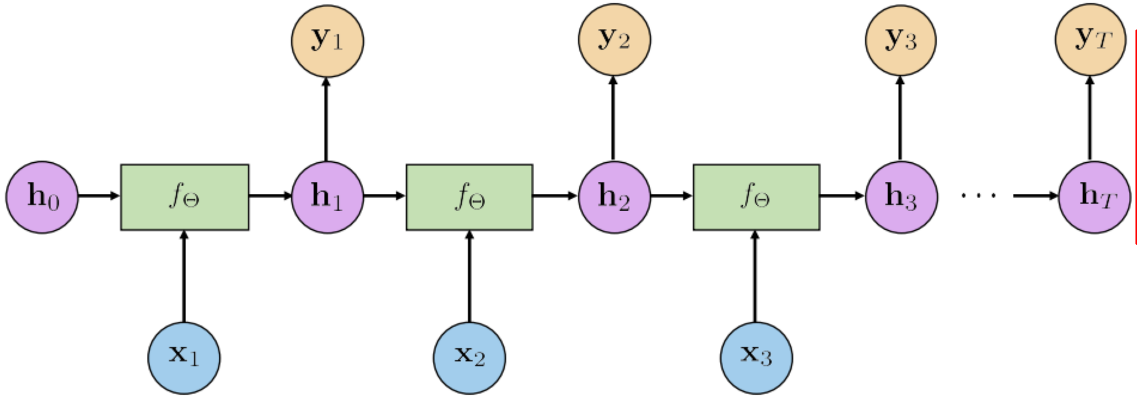
-



Language DL Models

RNN

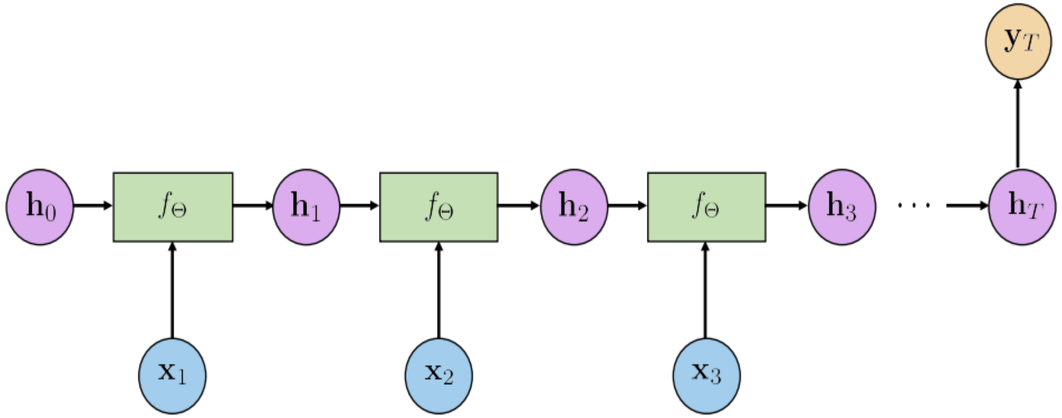
يستخدم في ترجمة الكلمات والجمل حيث لكل كلمة دخل كلمة مقابلة لها في الخرج وترجمة كل كلمة تعتمد على الكلمة السابقة



تسمى شبكات RNN من نمط many to many

كشف التشابه في المقالات العلمية (نعطي الشبكة تسلسل من الكلمات وتكشف لنا مدى أصلية المستند من عدمه)

تسمى شبكات RNN من نمط many to one

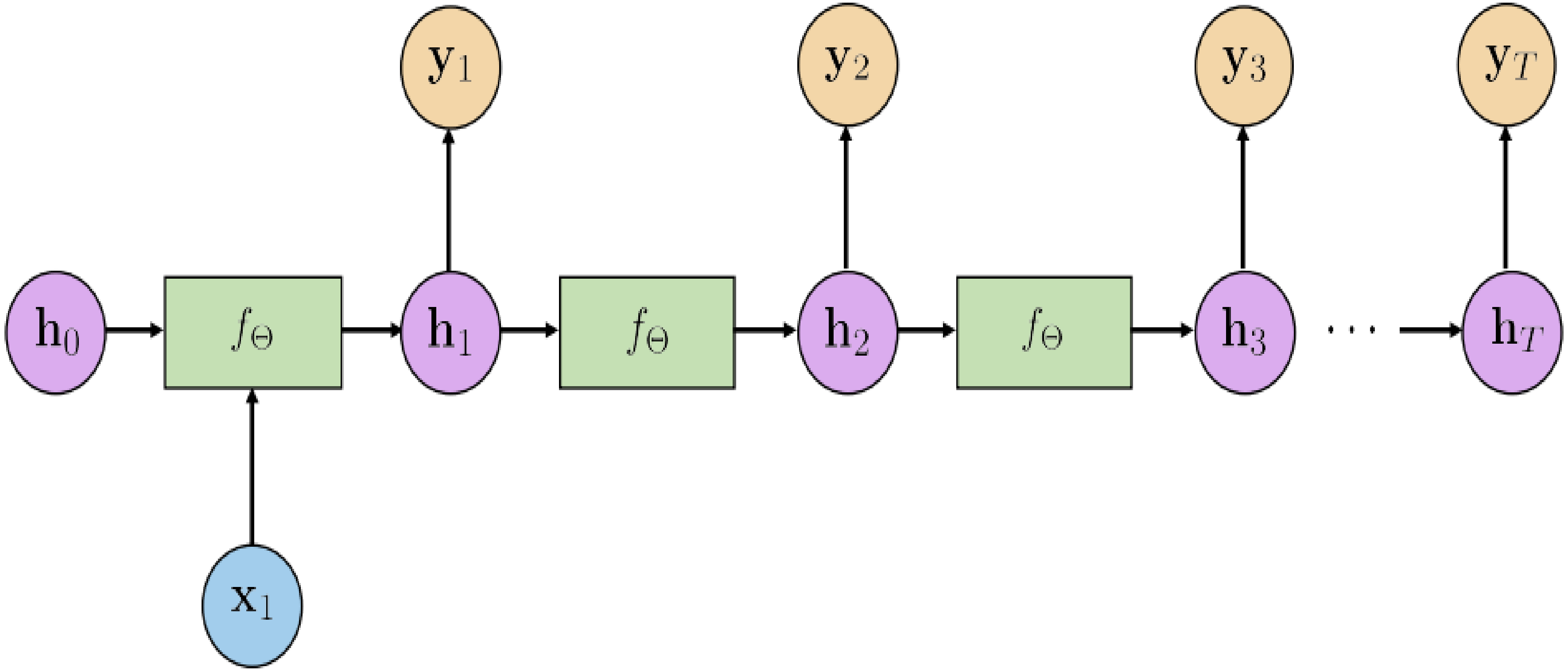


أيضاً تستخدم في تقييم النصوص

Language DL Models

RNN

يستخدم في عنونة الصور بمسميات توضيحية Image Captioning أو تحويل الصورة إلى جملة من الكلمات



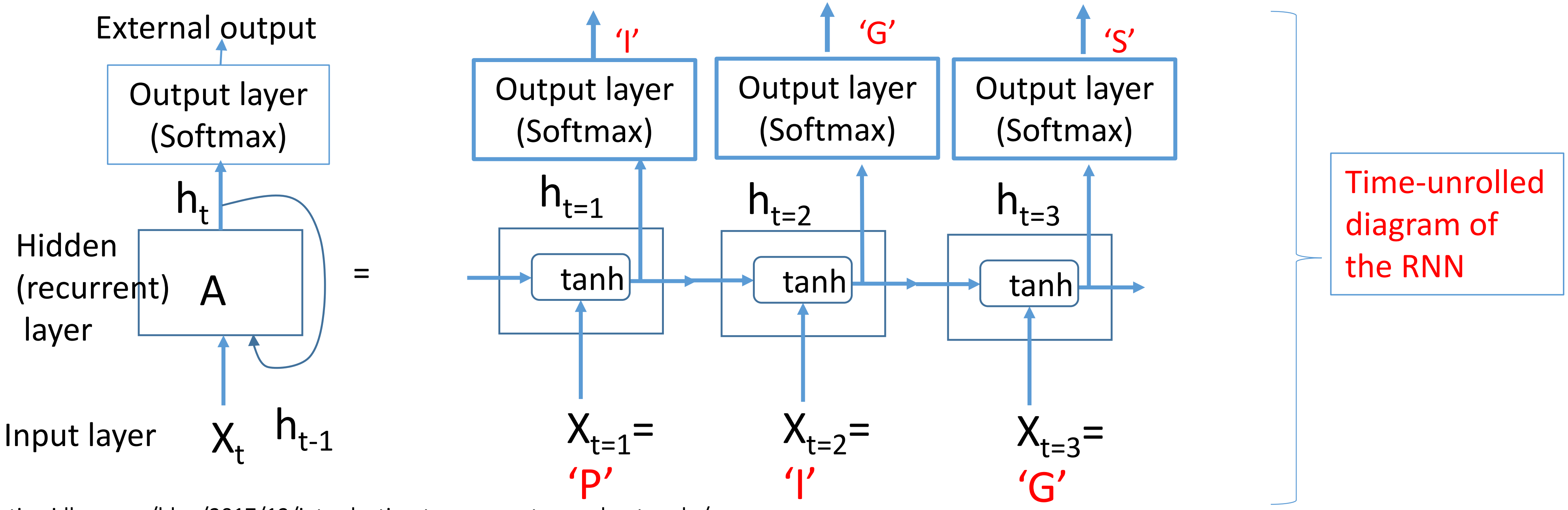
تسمى شبكات RNN من نمط one to many

Recurrent Neural Networks (RNN) | شبكات التعلم العميق

One hot	P	I	G	S
X1	1	0	0	0
X2	0	1	0	0
X3	0	0	1	0
X4	0	0	0	1

W_{hx}=[0.287027 0.84606 0.572392 0.486813
0.902874 0.871522 0.691079 0.18998
0.537524 0.09224 0.558159 0.491528]
W_{hy}=[0.37168 0.974829459 0.830034886
0.39141 0.282585823 0.659835709
0.64985 0.09821557 0.332487804
0.91266 0.32581642 0.144630018]
bias=0.567001*[1 1 1]' %random init. val
W_{hh}=0.427043*[1 1 1]'%random init. val

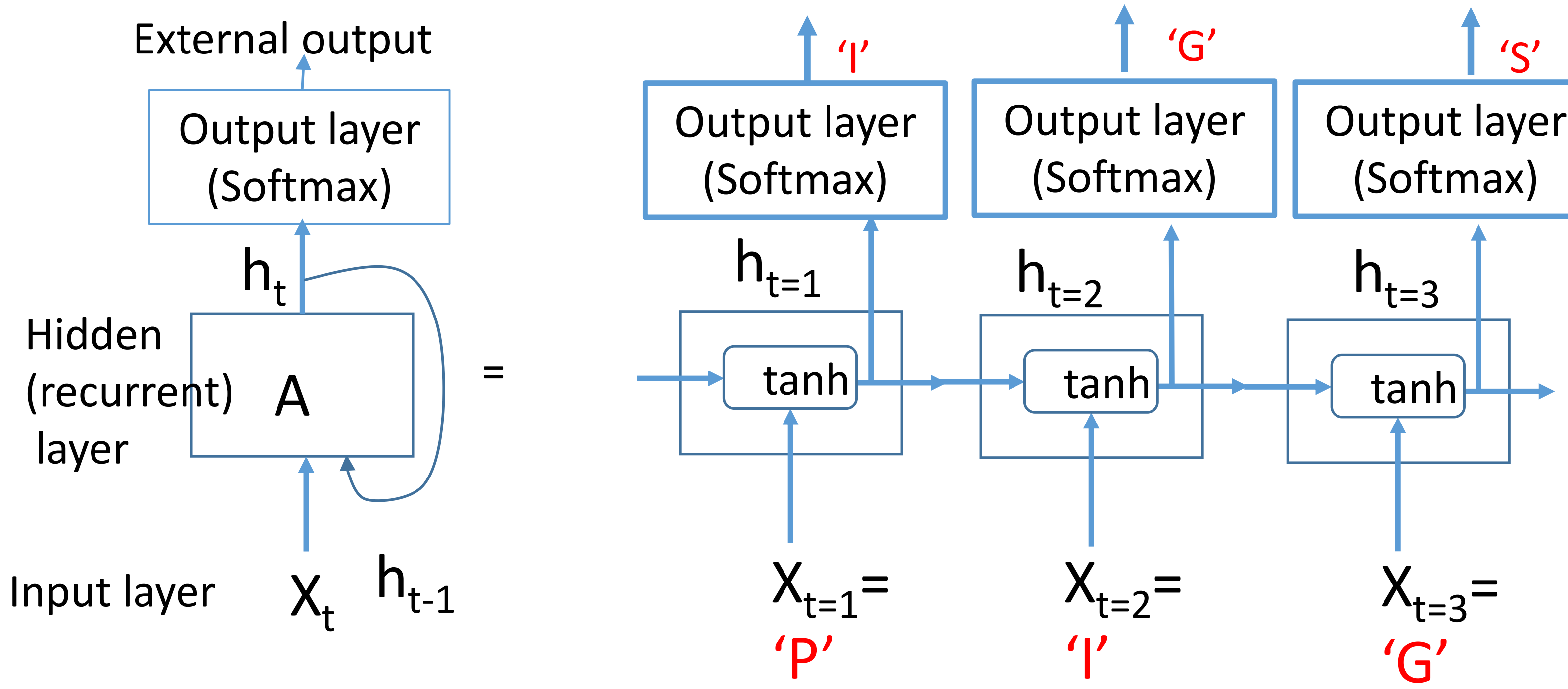
مثال تخمين الحرف التالي في لعبة
لدينا معجم المفردات التالي، ومصفوفة الدخل المجاورة مع مصفوفات الأوزان والانحيازات.
Dictionary={'P','I','G','S'}
If PIG is received then prediction is S
المطلوب وضح شكل شبكة RNN للمثال السابق.
احسبht(1,t+1) ،ht(2,t+1) ،ht(1,t+2) ،ht(1,t+3)
احسب المخارج النهائية
طبق تابع Softmax لحساب الخرج



https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

Recurrent Neural Networks (RNN) شبكات التعلم العميق

One hot	P	I	G	S
X1	1	0	0	0
X2	0	1	0	0
X3	0	0	1	0
X4	0	0	0	1



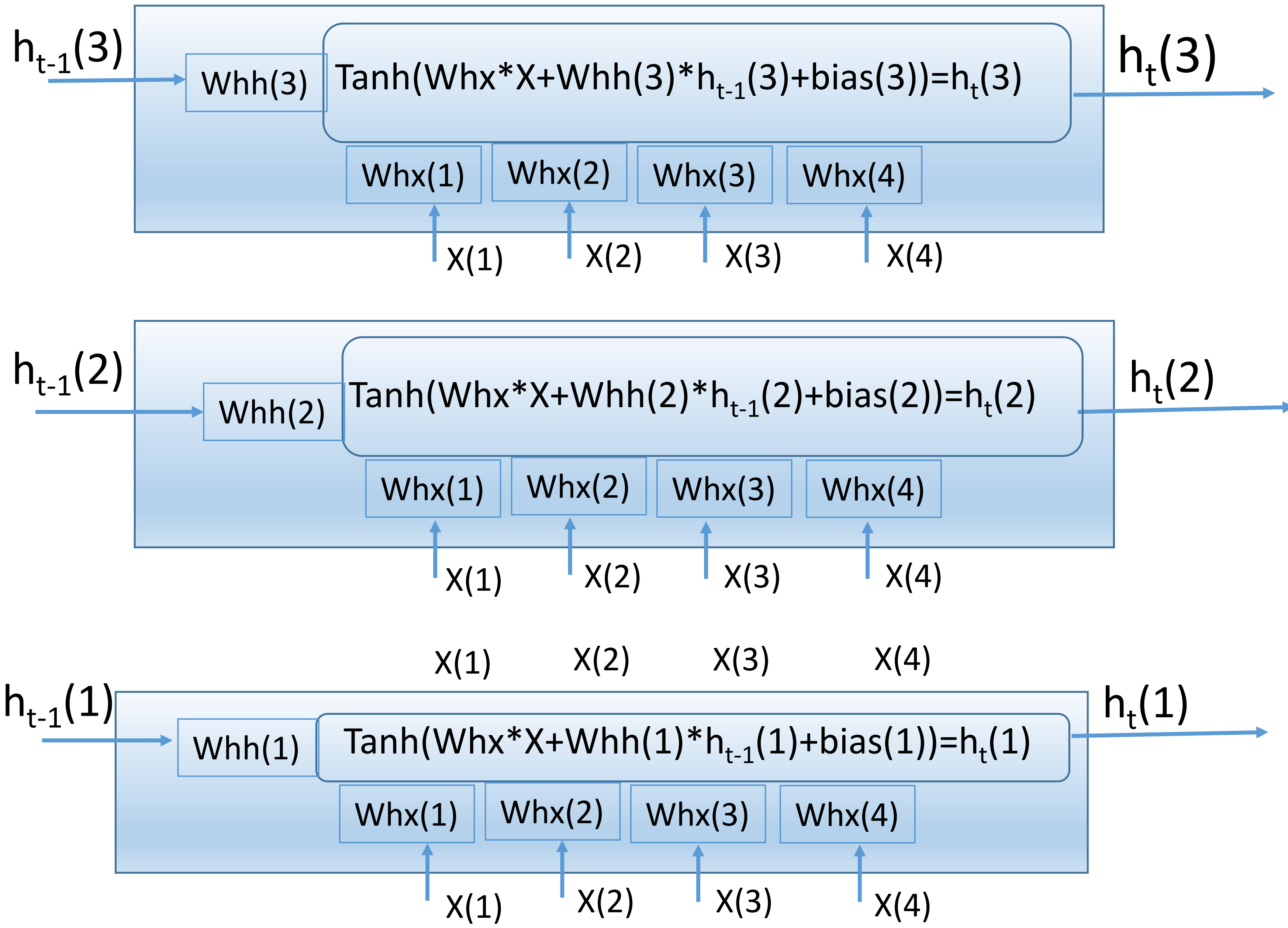
$$\mathrm{Tanh}(Whx*X + Whh(1)*h_{t-1}(1) + bias(1)) = h_t(1)$$
$$\mathrm{Tanh}(Whx*X + Whh(2)*h_{t-1}(2) + bias(2)) = h_t(2)$$
$$\mathrm{Tanh}(Whx*X + Whh(3)*h_{t-1}(3) + bias(3)) = h_t(3)$$

$$\tanh \chi = \frac{e^{\chi} - e^{-\chi}}{e^{\chi} + e^{-\chi}}$$

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

Recurrent Neural Networks (RNN) شبكات التعلم العميق

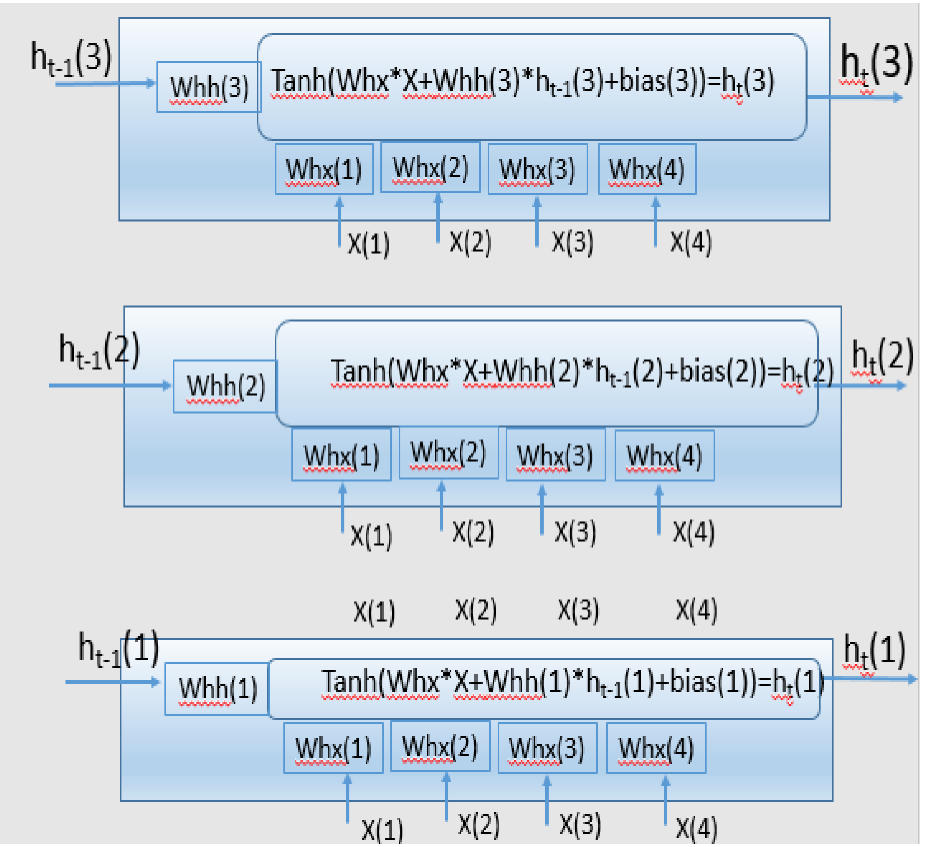
One hot	P	I	G	S
X1	1	0	0	0
X2	0	1	0	0
X3	0	0	1	0
X4	0	0	0	1



Recurrent Neural Networks (RNN) شبكات التعلم العميق

One hot	P	I	G	S
X1	1	0	0	0
X2	0	1	0	0
X3	0	0	1	0
X4	0	0	0	1

W_{hx}=[0.287027 0.84606 0.572392 0.486813
0.902874 0.871522 0.691079 0.18998
0.537524 0.09224 0.558159 0.491528]
W_{hy}=[0.37168 0.974829459 0.830034886
0.39141 0.282585823 0.659835709
0.64985 0.09821557 0.332487804
0.91266 0.32581642 0.144630018]
bias=0.567001*[1 1 1]' , **W_{hh}**=0.427043*[1 1 1]‘
ht(:,1)=[0 0 0]'



ht(:,t+1)=tanh(whx*in(:,t)+whh.*ht(:,t)+bias)
ht(1,t+1)=tanh([0.287027 0.84606 0.572392 0.486813]*[1 0 0 0]'+0*0.427043+0.567001)= **0.6932**

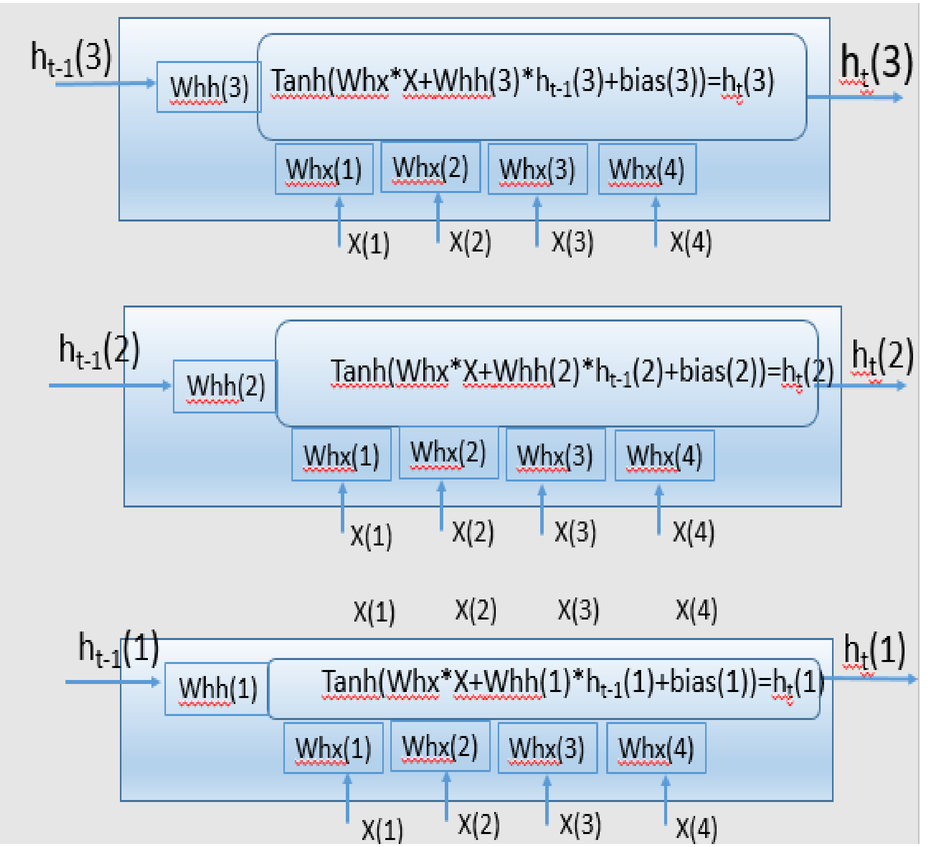
ht(2,t+1)=tanh([0.902874 0.871522 0.691079 0.18998]*[1 0 0 0]'+0*0.427043+0.567001)=**0.8996**

ht(3,t+1)=tanh([0.537524 0.09224 0.558159 0.491528]*[1 0 0 0]'+0*0.427043+0.567001)= **0.8021**

Recurrent Neural Networks (RNN) شبكات التعلم العميق

One hot	P	I	G	S
X1	1	0	0	0
X2	0	1	0	0
X3	0	0	1	0
X4	0	0	0	1

$\mathbf{W_{hx}}$=[0.287027 0.84606 0.572392 0.486813
0.902874 0.871522 0.691079 0.18998
0.537524 0.09224 0.558159 0.491528]
$\mathbf{W_{hy}}$=[0.37168 0.974829459 0.830034886
0.39141 0.282585823 0.659835709
0.64985 0.09821557 0.332487804
0.91266 0.32581642 0.144630018]
bias=0.567001*[1 1 1]' , $\mathbf{W_{hh}}$=0.427043*[1 1 1]'
ht(:,1)=[0 0 0]'



ht(:,t+1)=tanh(whx*in(:,t)+whh.*ht(:,t)+bias)

ht(1,t+2)=tanh([0.287027 0.84606 0.572392 0.486813]*[0 1 0 0]'+**0.6932***0.427043+0.567001)=**0.9365**

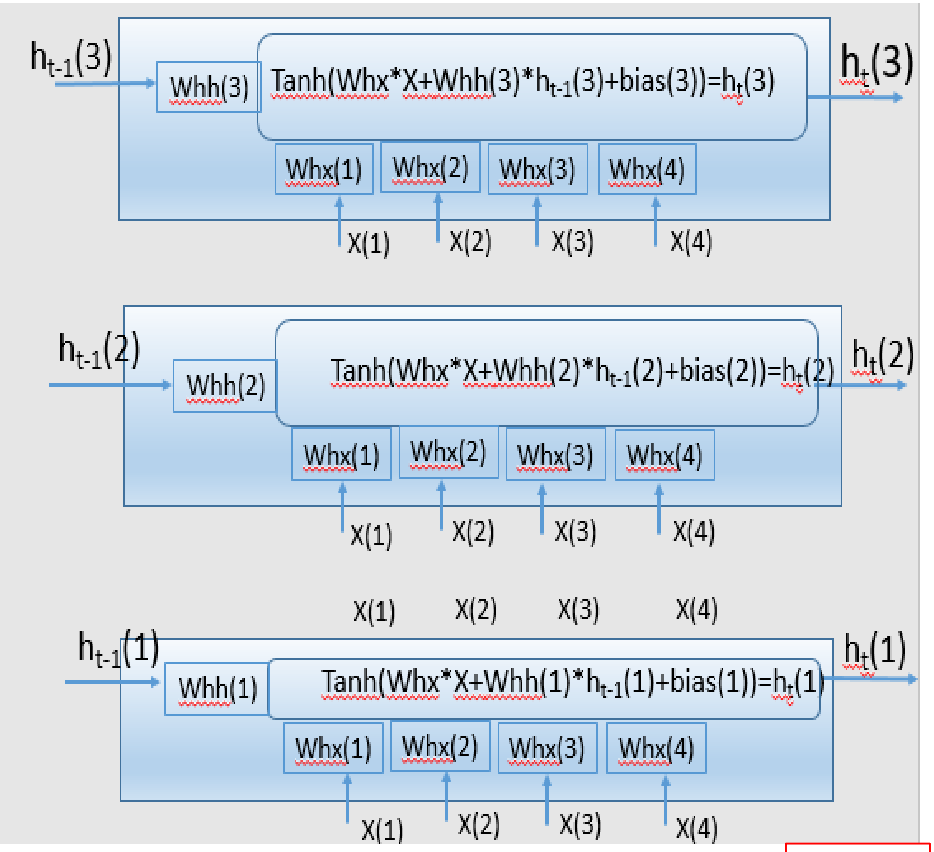
ht(2,t+2)=tanh([0.902874 0.871522 0.691079 0.18998]*[0 1 0 0]'+**0.8996***0.427043+0.567001)= **0.9491**

ht(3,t+2)=tanh([0.537524 0.09224 0.558159 0.491528]*[0 1 0 0]'+**0.8021***0.427043+0.567001)= **0.7623**

Recurrent Neural Networks (RNN) شبكات التعلم العميق

One hot	P	I	G	S
X1	1	0	0	0
X2	0	1	0	0
X3	0	0	1	0
X4	0	0	0	1

W_{hx}=[0.287027 0.84606 0.572392 0.486813
0.902874 0.871522 0.691079 0.18998
0.537524 0.09224 0.558159 0.491528]
W_{hy}=[0.37168 0.974829459 0.830034886
0.39141 0.282585823 0.659835709
0.64985 0.09821557 0.332487804
0.91266 0.32581642 0.144630018]
bias=0.567001*[1 1 1]' , **W_{hh}**=0.427043*[1 1 1]‘
ht(:,1)=[0 0 0]'



ht(:,t+1)=tanh(whx*in(:,t)+whh.*ht(:,t)+bias)
ht(1,t+3)=tanh([0.287027 0.84606 0.572392 0.486813]*[0 0 1 0]'+**0.9365***0.427043+0.567001)=**0.9120**

ht(2,t+3)=tanh([0.902874 0.871522 0.691079 0.18998]*[0 0 1 0]'+**0.9491***0.427043+0.567001)= **0.9307**

ht(3,t+3)=tanh([0.537524 0.09224 0.558159 0.491528]*[0 0 1 0]'+**0.7623***0.427043+0.567001)= **0.8958**

Time

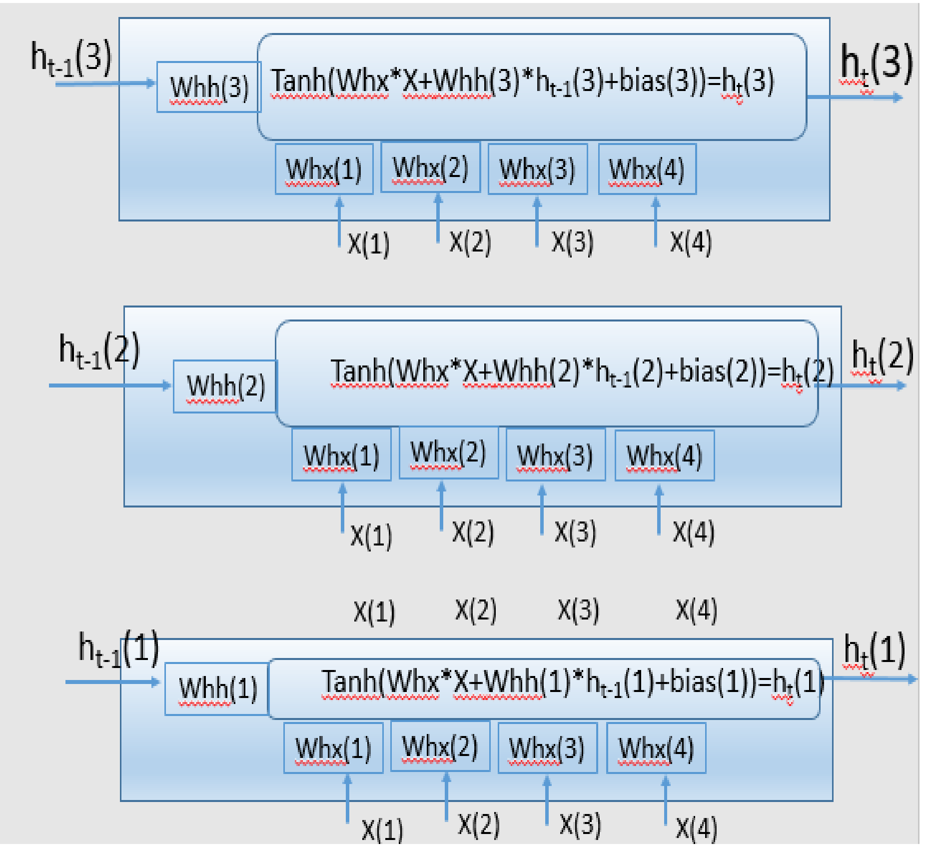
0	**0.6932**	**0.9365**	**0.9120**
0	**0.8996**	**0.9491**	**0.9307**
0	**0.8021**	**0.7623**	**0.8958**

H_t matrix

Recurrent Neural Networks (RNN) شبكات التعلم العميق

One hot	P	I	G	S
X1	1	0	0	0
X2	0	1	0	0
X3	0	0	1	0
X4	0	0	0	1

W_{hx}=[0.287027 0.84606 0.572392 0.486813
0.902874 0.871522 0.691079 0.18998
0.537524 0.09224 0.558159 0.491528]
W_{hy}=[0.37168 0.974829459 0.830034886
0.39141 0.282585823 0.659835709
0.64985 0.09821557 0.332487804
0.91266 0.32581642 0.144630018]
bias=0.567001*[1 1 1]' , **W_{hh}**=0.427043*[1 1 1]'
ht(:,1)=[0 0 0]'



y_out(:,t+1)=w_{hy}*ht(:,t+1)
y_out**(1,t+1)=**[0.37168 0.974829459 0.830034886]*[0.6932 0.8996 0.8021]'=**1.8004**

y_out**(2,t+1)=**[0.39141 0.282585823 0.659835709]*[0.6932 0.8996 0.8021]'= **1.0548**

y_out**(3,t+1)=**[0.64985 0.09821557 0.332487804]*[0.6932 0.8996 0.8021]'= **0.8055**

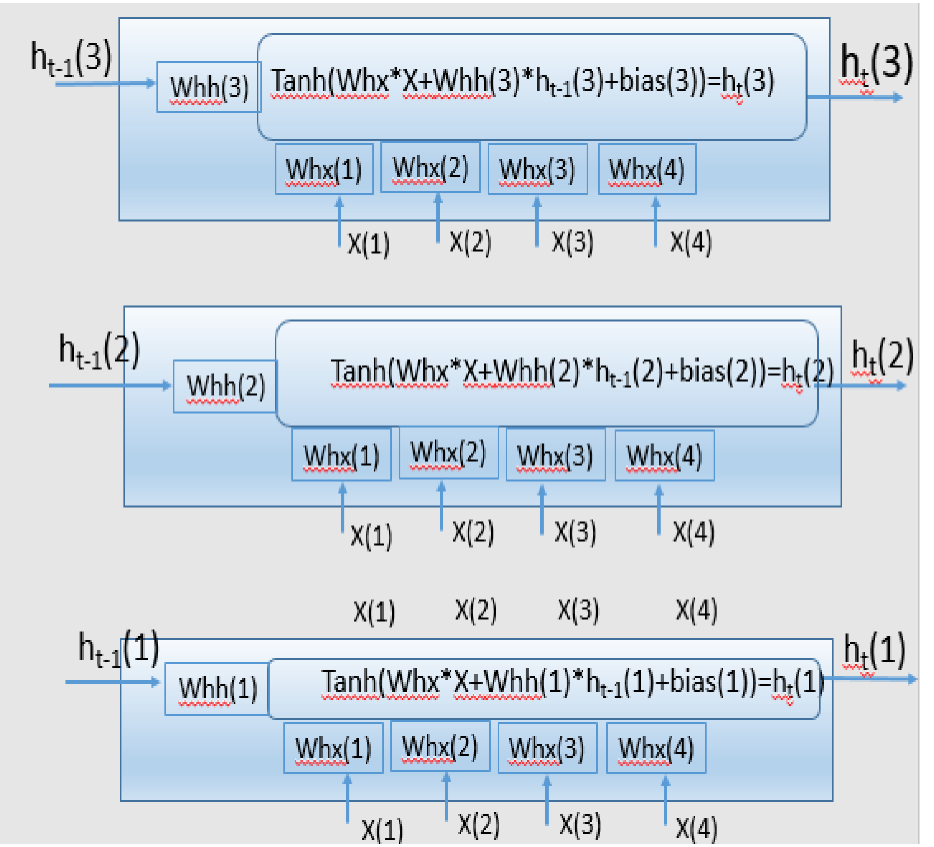
y_out**(4,t+1)=**[0.91266 0.32581642 0.144630018]*[0.6932 0.8996 0.8021]'= **1.0418**

0	**0.6932**	**0.9365**	**0.9120**
0	**0.8996**	**0.9491**	**0.9307**
0	**0.8021**	**0.7623**	**0.8958**

Recurrent Neural Networks (RNN) شبكات التعلم العميق

One hot	P	I	G	S
X1	1	0	0	0
X2	0	1	0	0
X3	0	0	1	0
X4	0	0	0	1

$\mathbf{W_{hx}}$=[0.287027 0.84606 0.572392 0.486813
0.902874 0.871522 0.691079 0.18998
0.537524 0.09224 0.558159 0.491528]
$\mathbf{W_{hy}}$=[0.37168 0.974829459 0.830034886
0.39141 0.282585823 0.659835709
0.64985 0.09821557 0.332487804
0.91266 0.32581642 0.144630018]
bias=0.567001*[1 1 1]' , $\mathbf{W_{hh}}$=0.427043*[1 1 1]'
ht(:,1)=[0 0 0]'



y_out(:,t+1)=w_{hy}*ht(:,t+1)
y_out**(1,t+2)=**[0.37168 0.974829459 0.830034886]*[0.9365 0.9491 0.7623]'=**1.9060**

y_out**(2,t+2)=**[0.39141 0.282585823 0.659835709]*[0.9365 0.9491 0.7623]'= **1.1378**

y_out**(3,t+2)=**[0.64985 0.09821557 0.332487804]*[0.9365 0.9491 0.7623]'= **0.9553**

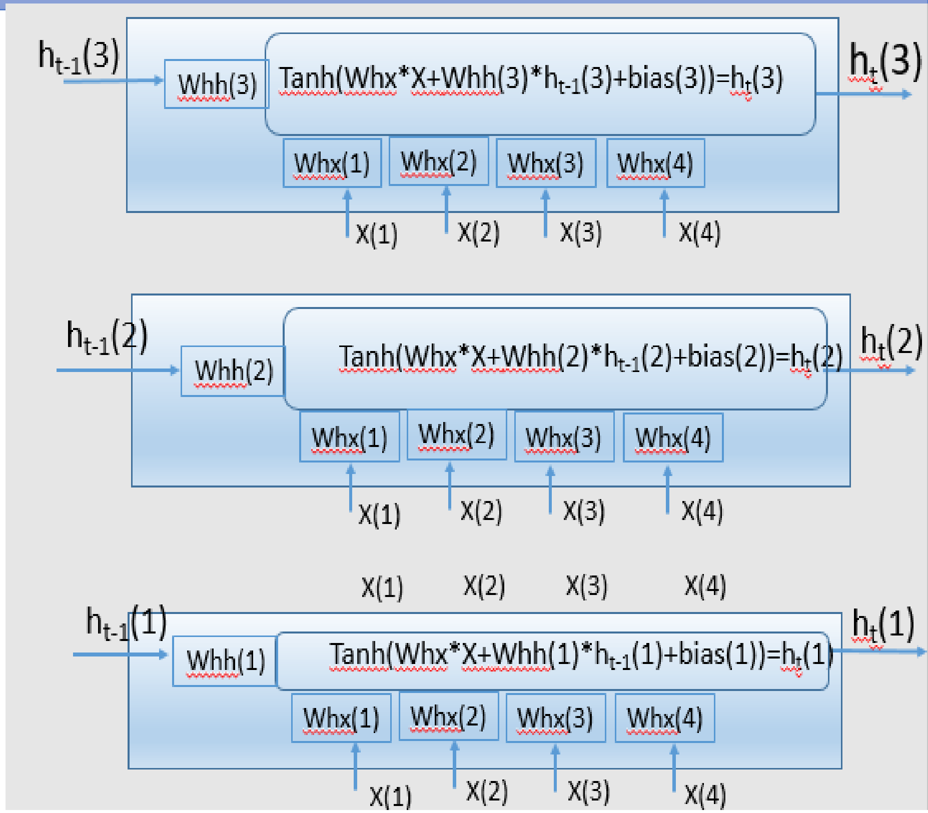
y_out**(4,t+2)=**[0.91266 0.32581642 0.144630018]*[0.9365 0.9491 0.7623]'= **1.2742**

0	**0.6932**	**0.9365**	**0.9120**
0	**0.8996**	**0.9491**	**0.9307**
0	**0.8021**	**0.7623**	**0.8958**

Recurrent Neural Networks (RNN) شبكات التعلم العميق

One hot	P	I	G	S
X1	1	0	0	0
X2	0	1	0	0
X3	0	0	1	0
X4	0	0	0	1

W_{hx}=[0.287027 0.84606 0.572392 0.486813
0.902874 0.871522 0.691079 0.18998
0.537524 0.09224 0.558159 0.491528]
W_{hy}=[0.37168 0.974829459 0.830034886
0.39141 0.282585823 0.659835709
0.64985 0.09821557 0.332487804
0.91266 0.32581642 0.144630018]
bias=0.567001*[1 1 1]' , **W_{hh}**=0.427043*[1 1 1]'
ht(:,1)=[0 0 0]'



y_out(:,t+1)=w_{hy}*ht(:,t+1)
y_out**(1,t+3)=**[0.37168 0.974829459 0.830034886]*[0.9120 0.9307 0.8958]'=**1.9898**

y_out**(2,t+3)=**[0.39141 0.282585823 0.659835709]*[0.9120 0.9307 0.8958]'= **1.2110**

y_out**(3,t+3)=**[0.64985 0.09821557 0.332487804]*[0.9120 0.9307 0.8958]'= **0.9819**

y_out**(4,t+3)=**[0.91266 0.32581642 0.144630018]*[0.8958]'= **1.2651**

Y_out matrix

0	**1.8003**	**1.9061**	**1.9898**
0	1.0548	1.1378	1.2111
0	0.8055	0.9553	0.9819
0	1.0417	1.2742	1.2651

Recurrent Neural Networks (RNN) شبكات التعلم العميق

One hot	P	I	G	S
X1	1	0	0	0
X2	0	1	0	0
X3	0	0	1	0
X4	0	0	0	1

Y_out matrix

0	**1.8003**	**1.9061**	**1.9898**
0	1.0548	1.1378	1.2111
0	0.8055	0.9553	0.9819
0	1.0417	1.2742	1.2651

Softmax(y_out)$_{i=1,2,3,4}$

$$\text{softmax}(y_i) = \frac{\exp(y_i)}{\sum_{i=1}^{n} \exp(y_i)},$$

for $i = 1,2,..,n$

Y_out$_{i=1,2,3,4}$

Weights: Why (4x3)

$h_t(1)$ $h_t(2)$ $h_t(3)$

The output layer

Softmax_y_out(:,t+1)=softmax(y_out(:,t+1))

Softmax_y_out(1,t+1)=
exp(1.8003)/(exp(1.8003)+exp(1.0548)+exp(0.8055)+exp(1.0417))=0.4324

Softmax_y_out(2,t+1)=
exp(1.0548)/(exp(1.8003)+exp(1.0548)+exp(0.8055)+exp(1.0417))=0.2052

Softmax_y_out(3,t+1)=
exp(0.8055)/(exp(1.8003)+exp(1.0548)+exp(0.8055)+exp(1.0417))=0.1599

Softmax_y_out(4,t+1)=
exp(1.0417)/(exp(1.8003)+exp(1.0548)+exp(0.8055)+exp(1.0417))=0.2025

softmax_y_out =

0	0.4324	0.4198	0.4332
0	0.2052	0.1947	0.1988
0	0.1599	0.1622	0.1581
0	0.2025	0.2232	0.2099

Recurrent Neural Networks (RNN) شبكات التعلم العميق

One hot	P	I	G	S
X1	1	0	0	0
X2	0	1	0	0
X3	0	0	1	0
X4	0	0	0	1

$\text{Softmax}(y\_out)_{i=1,2,3,4}$

$$\text{softmax}(y_i) = \frac{\exp(y_i)}{\sum_{i=1}^{n} \exp(y_i)},$$

$\text{for } i = 1,2,..,n$

$Y\_out_{i=1,2,3,4}$

Weights: Why (4x3)

$h_t(1)$ $h_t(2)$ $h_t(3)$

The output layer

Time=	0	1	2	3
P	0	**0.4324**	**0.4198**	**0.4332**
I	0	0.2052	0.1947	0.1988
G	0	0.1599	0.1622	0.1581
S	0	0.2025	0.2232	0.2099

y

نلاحظ أن الشبكة خمنت عند أول تكرار أن الخرج هو P وليس S وهذا خطأ مما يعني أننا نحتاج لنشر الخطأ ومتابعة تدريب الشبكة

Recurrent Neural Networks (RNN) | تطبيقات RNN

استخدام شبكات CNN و
RNN معاً لوصف الصور

Image Captioning

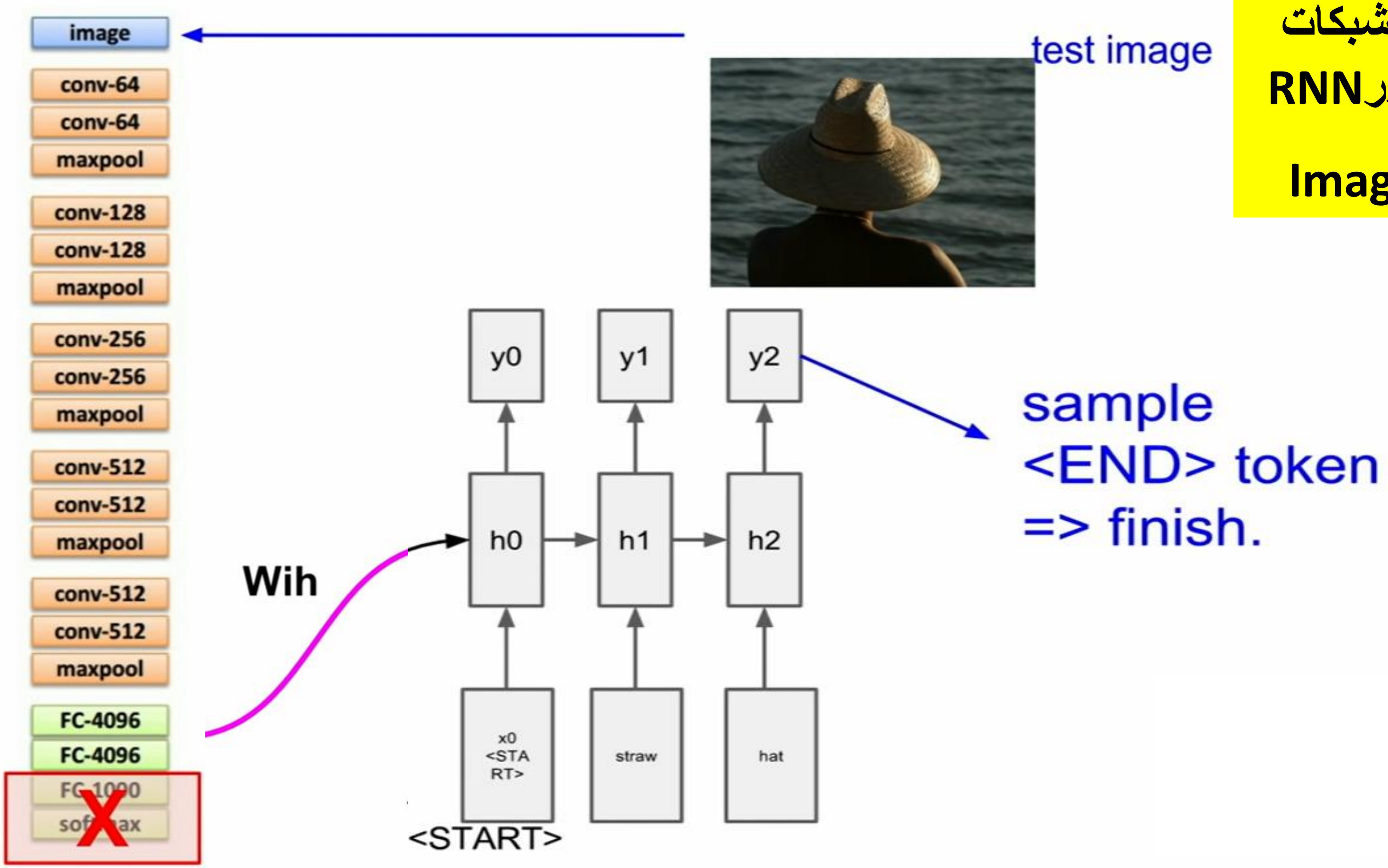


Image Captioning: Example Results

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

Language DL Models

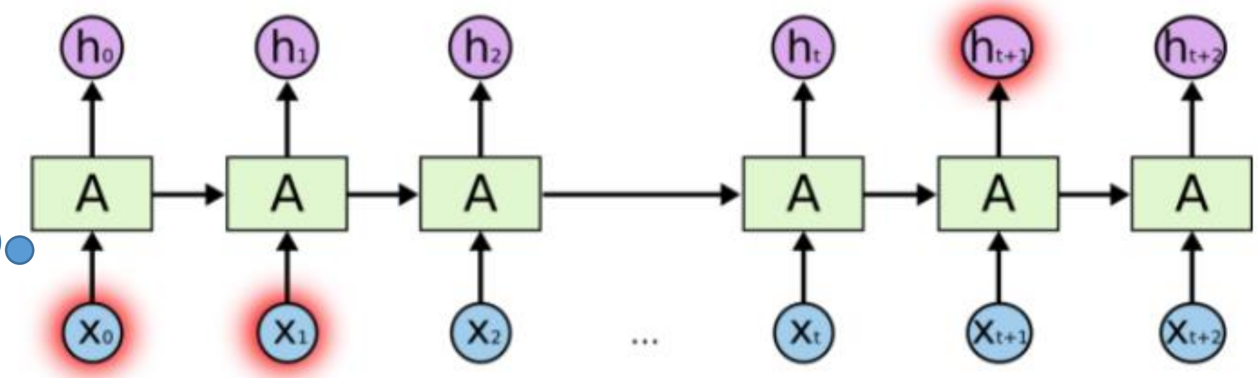
RNN Problems

-

Solve: using non-linear functions like Relu

Vanishing Gradient because of using Tanh or sigmoid activation functions

Solve: Upgrade to LSTM model

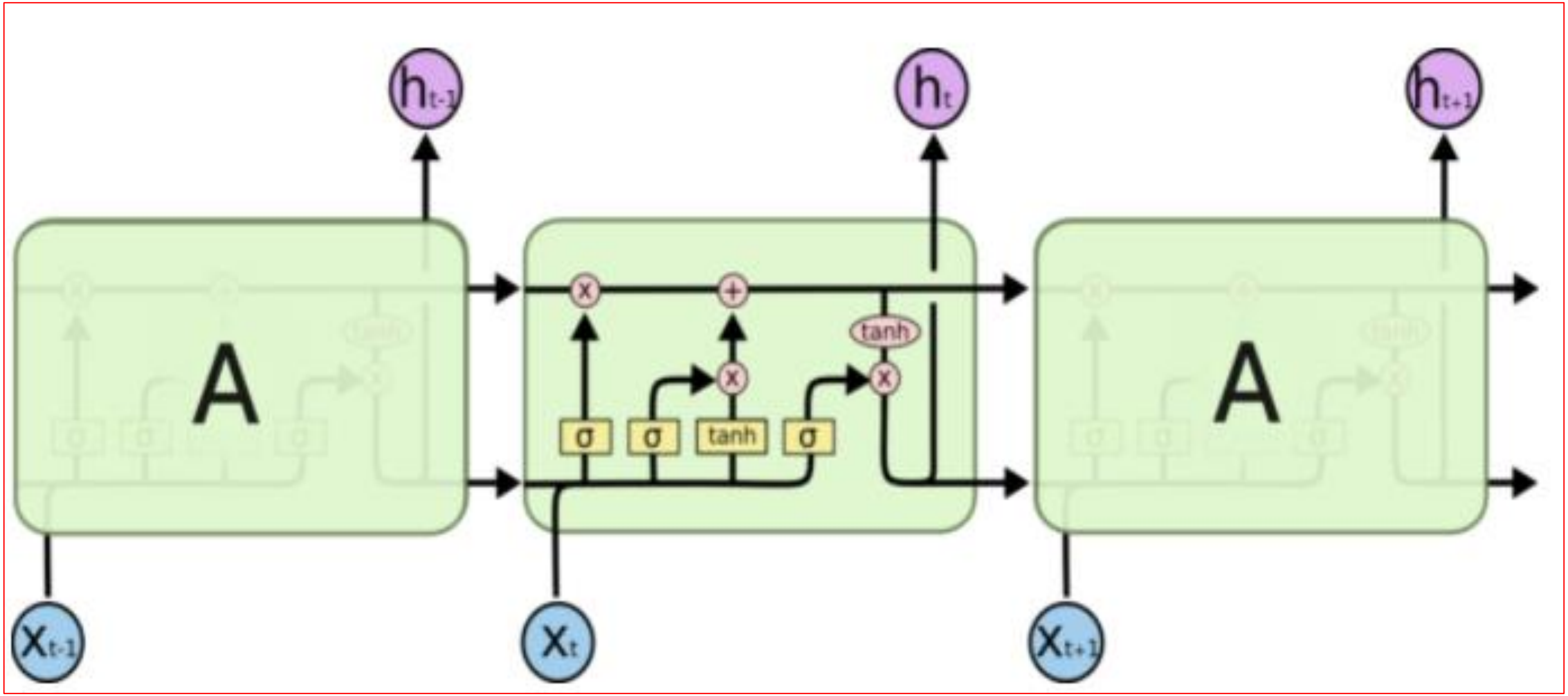


الشكل 3: ظهور حالة الاعتمادية طويلة المدى

Long term dependency الاعتمادية طويلة المدى (توليد جمل طويلة)

Language DL Models

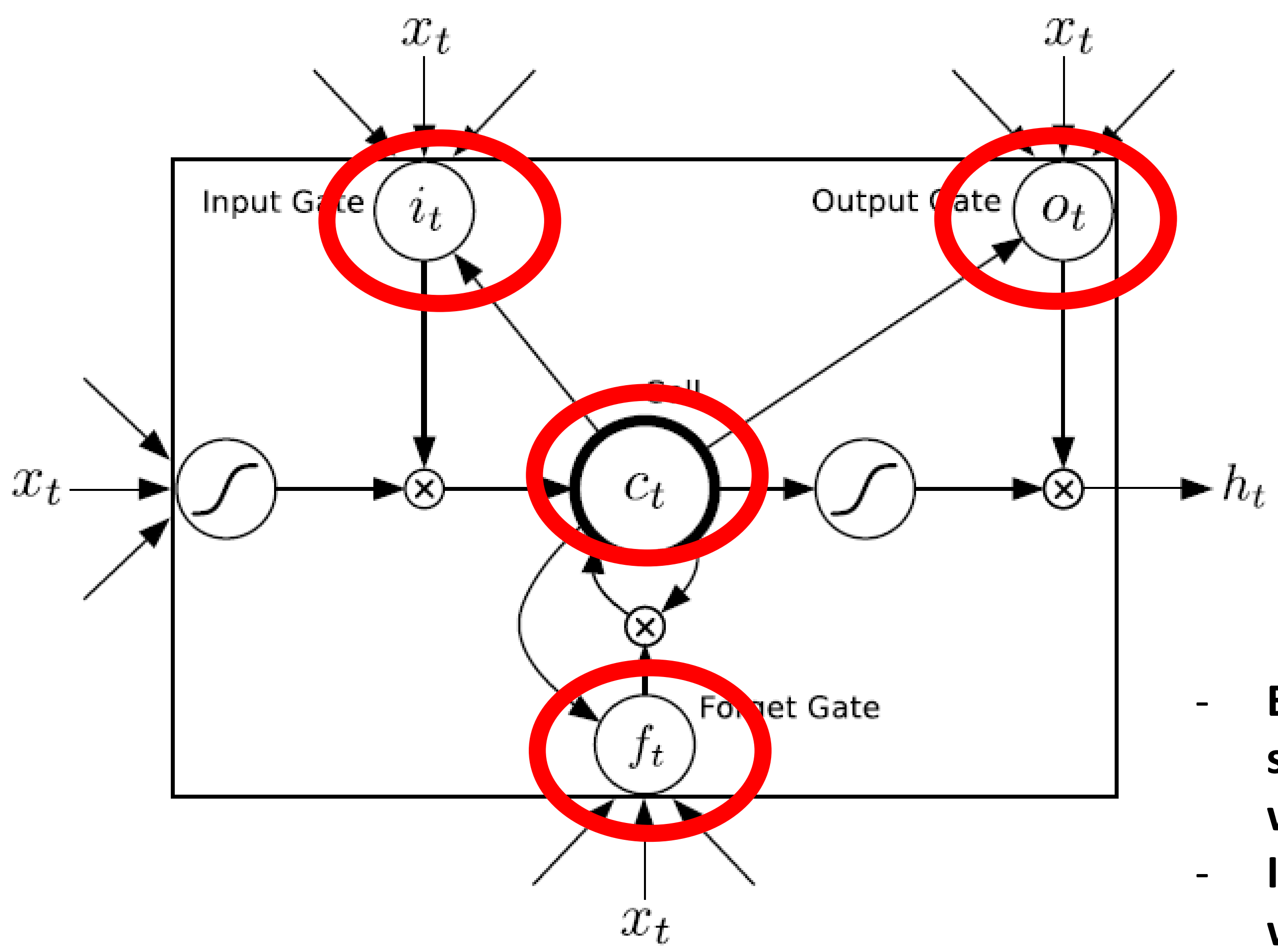
LSTM long-short term memory

- 

Language DL Models

LSTM long-short term memory

•



بوابة الدخل Input gate:

وزن الدخل وتقديمه للخلية وهي بمثابة بوابة الكتابة **Write**.

بوابة الخرج Output Gate:

أخذ الخرج من البوابة وتسمى بوابة القراءة **Read**.

بوابة النسيان Forget Gate:

إهمال الحالات المخفية السابقة وتسمى بوابة التصفير **Reset**.

خلية الذاكرة Cell:

مجموع قيمتين هما الحالة السابقة لخلية الذاكرة C_{t-1} والحالة المخفية السابقة h_{t-1}.

- **Example of generating descriptions for images: Here, similar words are linked together by generating the next word based on the previous one.**
- **If we want to forget the repetition of an unnecessary word, we make use of the "forget" cell.**

Long-Short Term Memory (LSTM) | شبكات التعلم العميق

LSTM for image description (captioning)

Ref: A man and a woman ride a motorcycle

A man and a woman are talking on the road

Ref: A woman is frying food

Someone is frying a fish in a pot

Classify Text Data Using Deep Learning (LSTM)

This example trains an LSTM model to predict whether a message represents Spam or a normal Ham (safe) message.

Step1: Import Dependencies

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import
pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.regularizers import l2
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
# Layers
from tensorflow.keras.layers import Bidirectional, Embedding,
LSTM, Dense, Dropout
```

Classify Text Data Using Deep Learning (LSTM)

Step2: View data

```
df = pd.read_csv('spam.csv', encoding = 'latin-1')
df
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will Ì_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

5572 rows × 5 columns

Classify Text Data Using Deep Learning (LSTM)

Step3: Removing empty columns

```
df = df.drop(columns=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'])
```

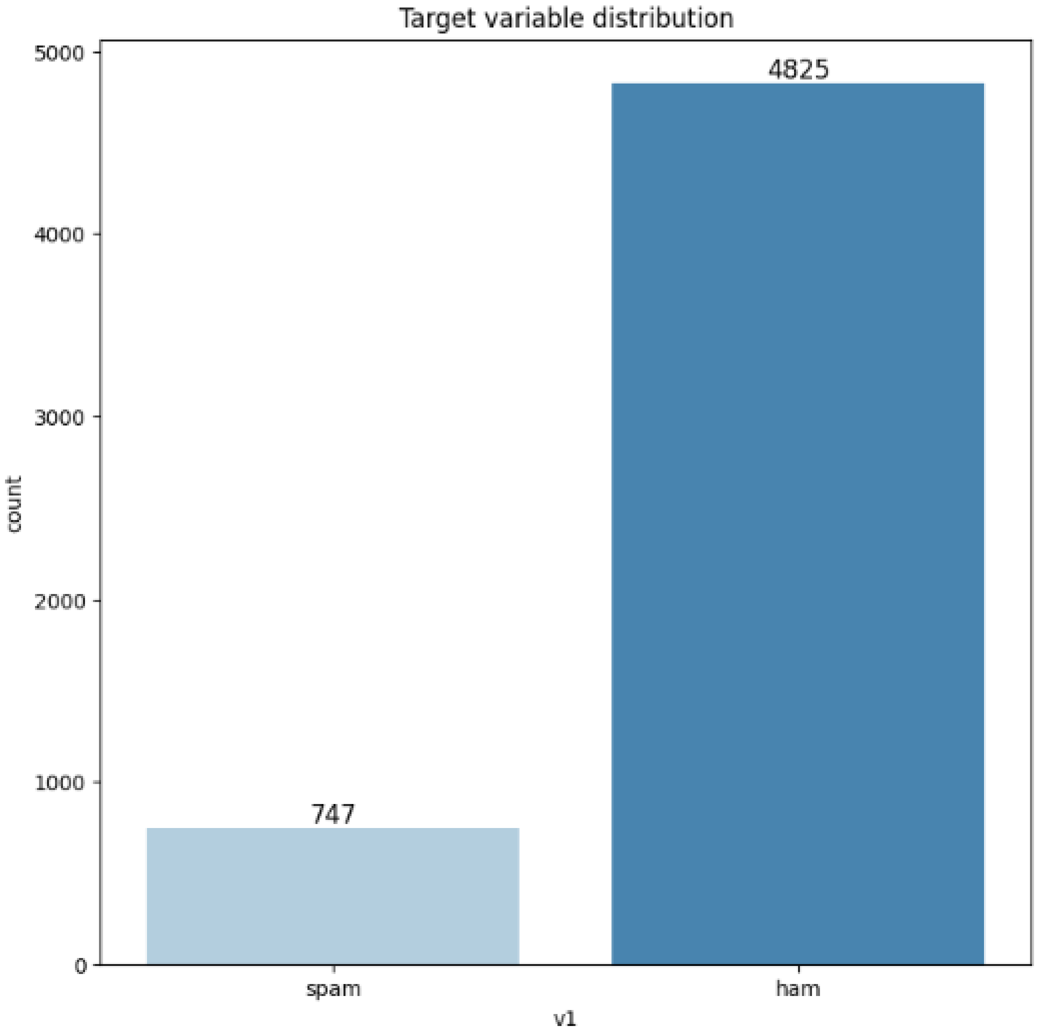
	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

Classify Text Data Using Deep Learning (LSTM)

Show class distribution

```
# Plotting a class distribution barplot
class_distribution =
df['v1'].value_counts().sort_index(ascen
ding = False)
plt.figure(figsize=(8,8))
ax = sns.countplot(x='v1', data=df,
order = class_distribution.index,
palette="Blues")
for i in ax.containers:
    ax.bar_label(i, label_type = 'edge',
fontsize = 12)
plt.title('Target variable
distribution')
plt.show()
```



Classify Text Data Using Deep Learning (LSTM)

Display some samples

```
# Printing examples of ham messages
print("Ham texts:")
print(df[df['v1'] == 'ham']['v2'].head())
```

```
Ham texts:
0    Go until jurong point, crazy.. Available only ...
1                        Ok lar... Joking wif u oni...
3    U dun say so early hor... U c already then say...
4    Nah I don't think he goes to usf, he lives aro...
6    Even my brother is not like to speak with me. ...
```

Display some samples

```
# Printing examples of Spam messages
print("Spams:")
print(df[df['v1'] == 'spam']['v2'].head())
```

```
Spams:
2     Free entry in 2 a wkly comp to win FA Cup fina...
5     FreeMsg Hey there darling it's been 3 week's n...
8     WINNER!! As a valued network customer you have...
9     Had your mobile 11 months or more? U R entitle...
11    SIX chances to win CASH! From 100 to 20,000 po...
```

Long-Short Term Memory (LSTM)

شبكات التعلم العميق

Classify Text Data Using Deep Learning (LSTM)

Step 4: Split dataset into train and test

```
import nltk
nltk.download('stopwords')
# Splitting training and testing
sets
X_train, X_test, y_train,y_test =
train_test_split(df['v2'],
df['v1'], test_size = 0.4,
random_state = 123)
```

تنظيف البيانات: إزالة علامات الترقيم، التحويل إلى أحرف صغيرة، إزالة الأرقام، إزالة كلمات التوقف في اللغة الإنكليزية، إعادة الكلمات لأصلها المعجمي، إزالة الفراغات

Step 5: Clean train and test datasets

```
def text_cleaning(text):
    # Removing punctuation
    text = re.sub(r'[^\w\s]', '', text)
    # Converting text to lowercase
    text = text.lower()
    # Removing digits
    text = re.sub(r'\d+','',text)
    # Removing stopwords that are common in English
    stop = stopwords.words('english')
    text = " ".join([word for word in text.split() if
word not in stop])
    # Lemmatizing text
    lemmatizer = WordNetLemmatizer()
    text = " ".join([lemmatizer.lemmatize(word) for word
in text.split()])
    # Removing white spaces
    text = text.strip()
    return text
```

```
# Applying text_cleaning function
X_train = X_train.apply(text_cleaning)
X_test = X_test.apply(text_cleaning)
```

Long-Short Term Memory (LSTM)

شبكات التعلم العميق

Classify Text Data Using Deep Learning (LSTM)

Step 6: Data tokenization and padding

```
max_lenght = max([len(i) for i in X_train])
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)

X_train =
tokenizer.texts_to_sequences(X_train)
X_train = pad_sequences(X_train, maxlen =
max_lenght)
X_test =
tokenizer.texts_to_sequences(X_test)
X_test = pad_sequences(X_test, maxlen =
max_lenght)
```

Step 7: Solve data imbalance problem

```
smote = SMOTE(random_state = 42)
X_train, y_train =
smote.fit_resample(X_train, y_train)
```

```
# Counting values after SMOTE
y_train.value_counts()
```

```
ham     2884
spam    2884
```

Step 8: Encode the targets

```
encoder = LabelEncoder()
y_train = encoder.fit_transform(y_train)
y_test = encoder.transform(y_test)
```

Classify Text Data Using Deep Learning (LSTM)

Step 9: Create LSTM Model

```
model = Sequential()
# Adding embedding layer to convert input data
into a dense vector representation
model.add(Embedding(input_dim=len(tokenizer.word
_index)+1, output_dim = 100, input_length =
max_lenght))
# Adding LSTM layers
model.add(LSTM(units=32, return_sequences =
True))
model.add(LSTM(units=32))
# Adding a Dense Layer
model.add(Dense(units=32, activation = 'relu'))
# Adding a Dropout layer, in order to prevent
overfitting
model.add(Dropout(rate=0.2))
# Adding an output Dense layer
model.add(Dense(units=1, activation =
'sigmoid'))
model.summary()
```

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 419, 100)	596500
lstm_2 (LSTM)	(None, 419, 32)	17024
lstm_3 (LSTM)	(None, 32)	8320
dense_2 (Dense)	(None, 32)	1056
dropout_1 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 1)	33

```
Total params: 622933 (2.38 MB)
Trainable params: 622933 (2.38 MB)
Non-trainable params: 0 (0.00 Byte)
```

Classify Text Data Using Deep Learning (LSTM)

Step 10: Define Training Parameters

```
# Defining an early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience = 5)
# Compiling model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics =
['accuracy'])
# Fitting model
history = model.fit(X_train, y_train, epochs = 10, batch_size = 32,
validation_split = 0.3, callbacks =[early_stopping])
```

```
Epoch 1/10
127/127 [==============================] - 28s 153ms/step - loss: 0.3835 - accuracy: 0.8236 - val_loss: 0.8241 - val_accuracy: 0.611
Epoch 2/10
127/127 [==============================] - 8s 61ms/step - loss: 0.1808 - accuracy: 0.9346 - val_loss: 0.6606 - val_accuracy: 0.7204
Epoch 3/10
127/127 [==============================] - 7s 52ms/step - loss: 0.0797 - accuracy: 0.9802 - val_loss: 0.9662 - val_accuracy: 0.6979
Epoch 4/10
127/127 [==============================] - 5s 39ms/step - loss: 0.0487 - accuracy: 0.9896 - val_loss: 1.2259 - val_accuracy: 0.6869
Epoch 5/10
127/127 [==============================] - 8s 66ms/step - loss: 0.0364 - accuracy: 0.9916 - val_loss: 1.6679 - val_accuracy: 0.5505
Epoch 6/10
127/127 [==============================] - 9s 74ms/step - loss: 0.0290 - accuracy: 0.9938 - val_loss: 1.7980 - val_accuracy: 0.6153
Epoch 7/10
127/127 [==============================] - 7s 55ms/step - loss: 0.0257 - accuracy: 0.9943 - val_loss: 1.1631 - val_accuracy: 0.7002
```

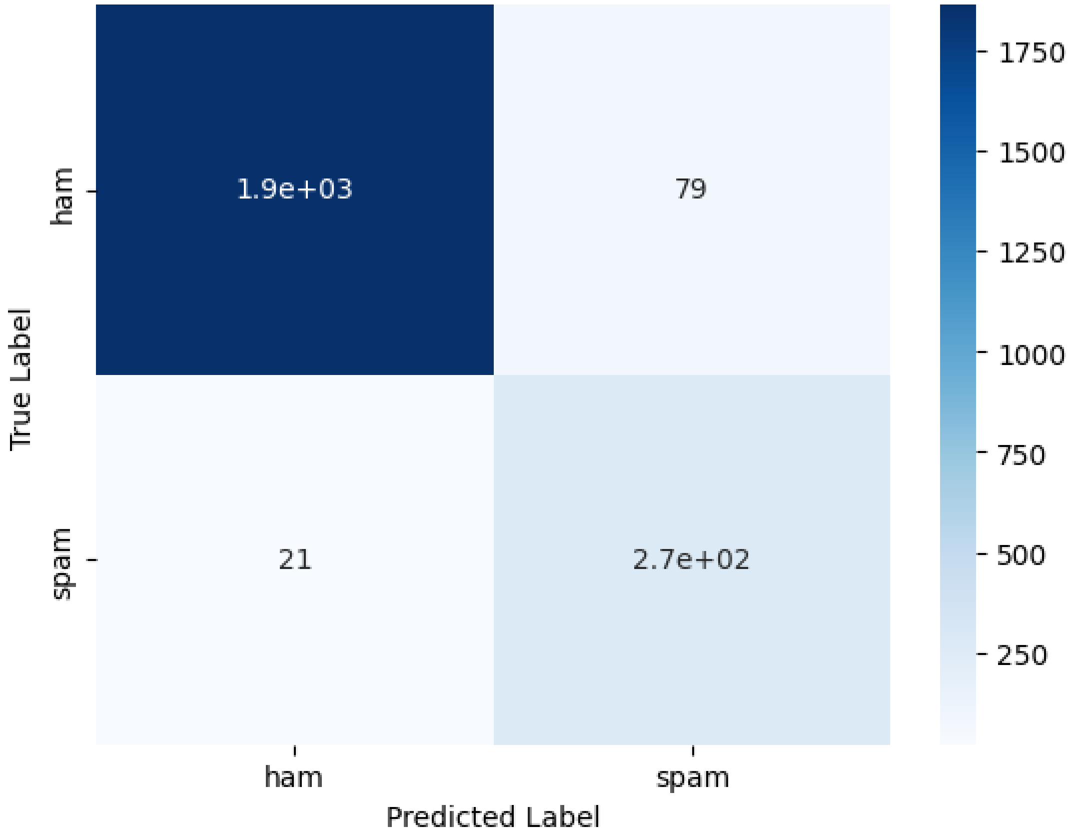
Classify Text Data Using Deep Learning (LSTM)

Step 10: Evaluate the model using X_test

```
# Running model on testing set
y_pred = model.predict(X_test)
y_pred = np.round(y_pred)
# Printing metric scores
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred))

# Plotting a confusion matrix
cm = confusion_matrix(y_test, y_pred).astype(int)
sns.heatmap(cm, annot=True, cmap='Blues',
xticklabels=['ham', 'spam'], yticklabels=['ham',
'spam'])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

```
Accuracy: 0.9551368326603858
Precision: 0.7716763005780347
Recall: 0.9270833333333334
F1 Score: 0.8422712933753943
```



Classify Text Data Using Deep Learning (LSTM)

Step 11: Test real example

```
def predict_spam_or_ham(sentence):
    # Preprocess the sentence (tokenization, removing stopwords, etc.)
    clean_sentence = text_cleaning(sentence)  # Assuming text_cleaning is your custom function
for preprocessing
    clean_tokenized_sentence = tokenizer.texts_to_sequences([clean_sentence])
    sentence_padded = pad_sequences(clean_tokenized_sentence, maxlen=max_lenght)
    # Predict
    prediction = model.predict(sentence_padded)

    # Interpret the prediction
    if prediction < 0.5:
        return "ham"
    else:
        return "spam"
```

```
# Example usage
sentence = "Follow this link to win 100$"
prediction = predict_spam_or_ham(sentence)
print(f"The sentence is predicted as: {prediction}")
```

```
1/1 [==============================] - 0s 41ms/step
The sentence is predicted as: spam
```

```
# Example usage
sentence = "Hi! How are you"
prediction = predict_spam_or_ham(sentence)
print(f"The sentence is predicted as: {prediction}")
```

```
1/1 [==============================] - 0s 114ms/step
The sentence is predicted as: ham
```