GLIACLOUD

GDG DevFest TAIPEI 2016

Activity Recognition with TensorFlow

TensorFlow, Deep Learning, and Activity Recognition

Presented by: Aaron Lai

Who Am I

Data Scientist

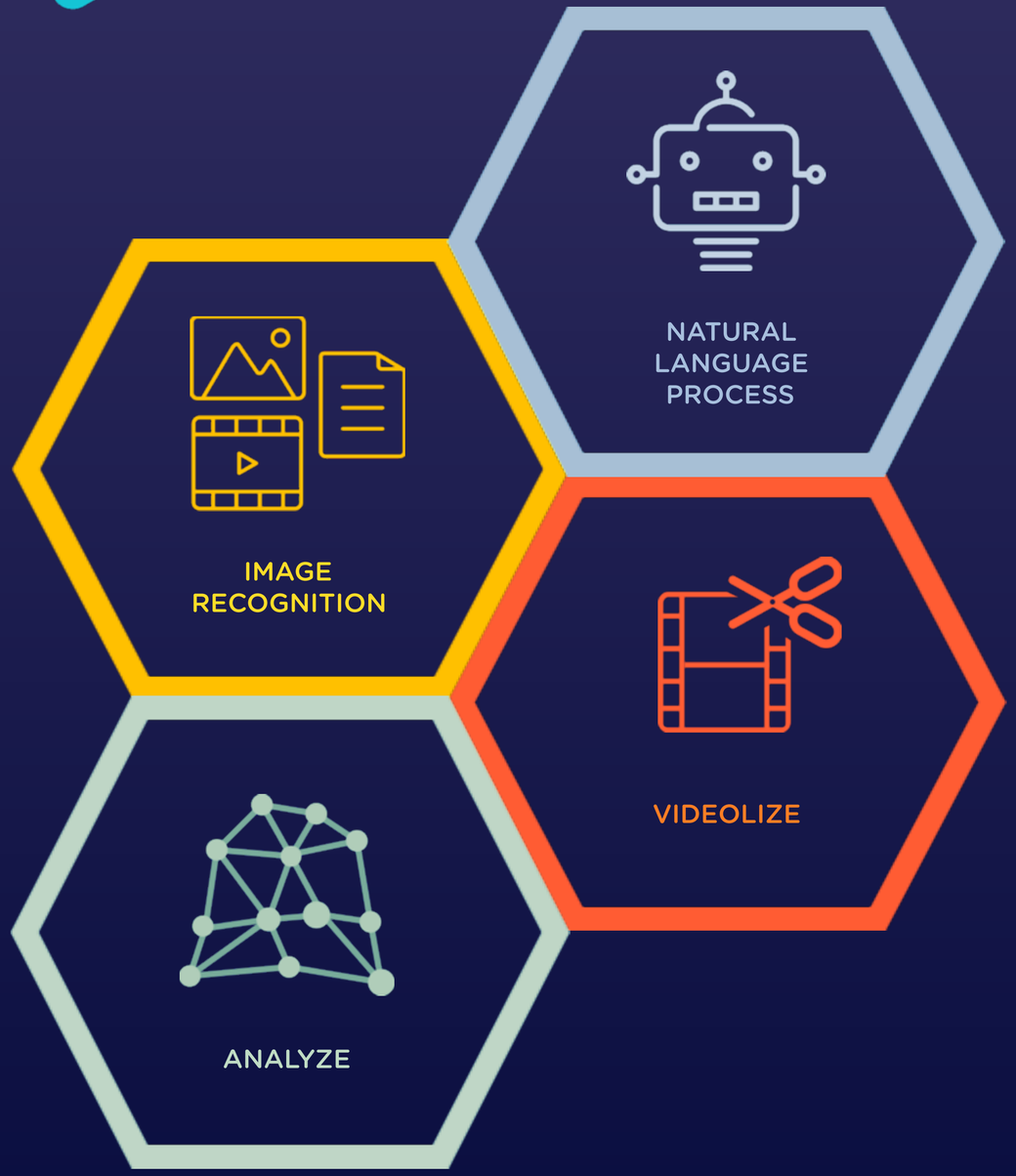
Building System

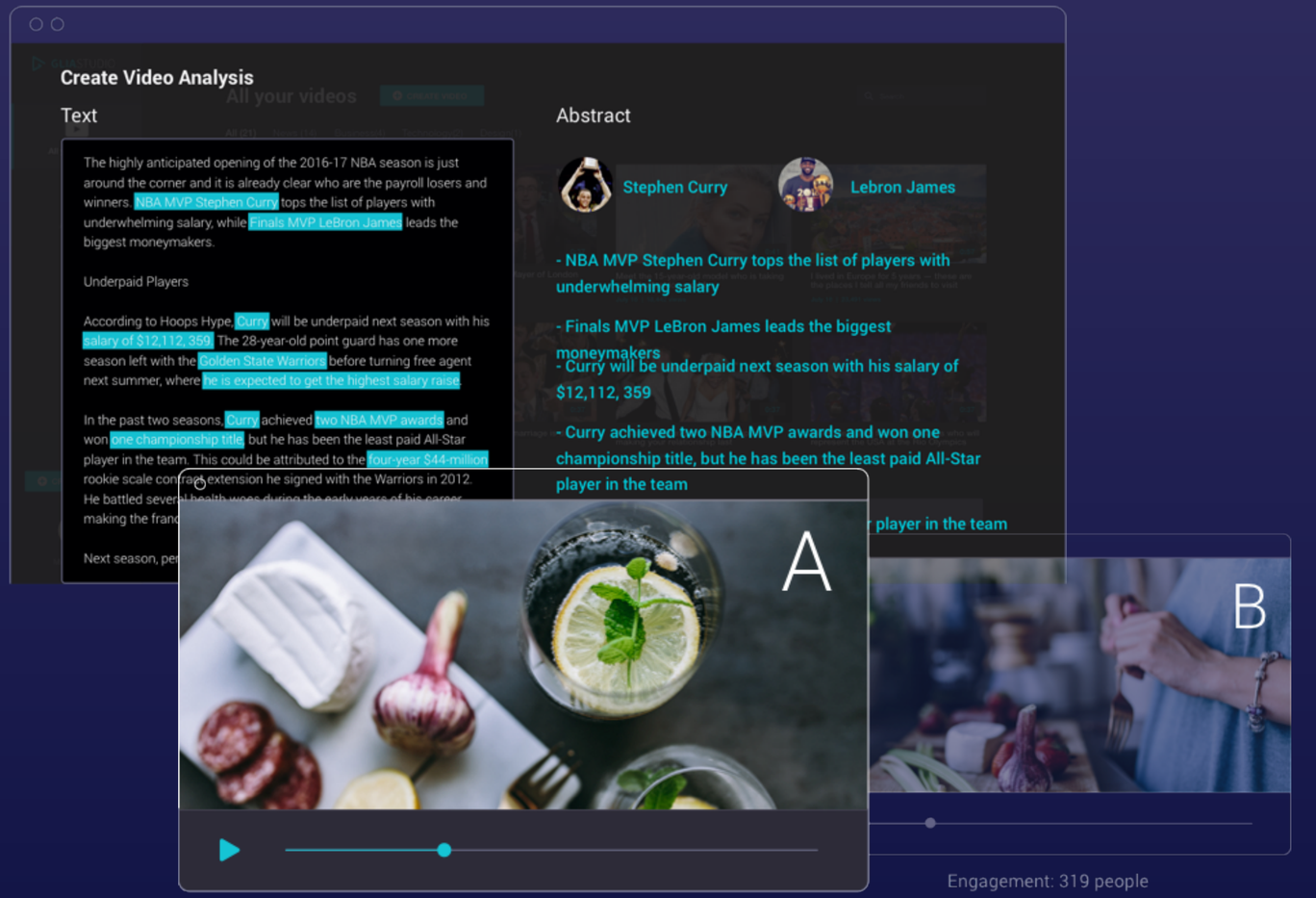
Playing New Tools

Data Analysis

Machine Learning

GLIASTUDIO - AI Video Creation Platform

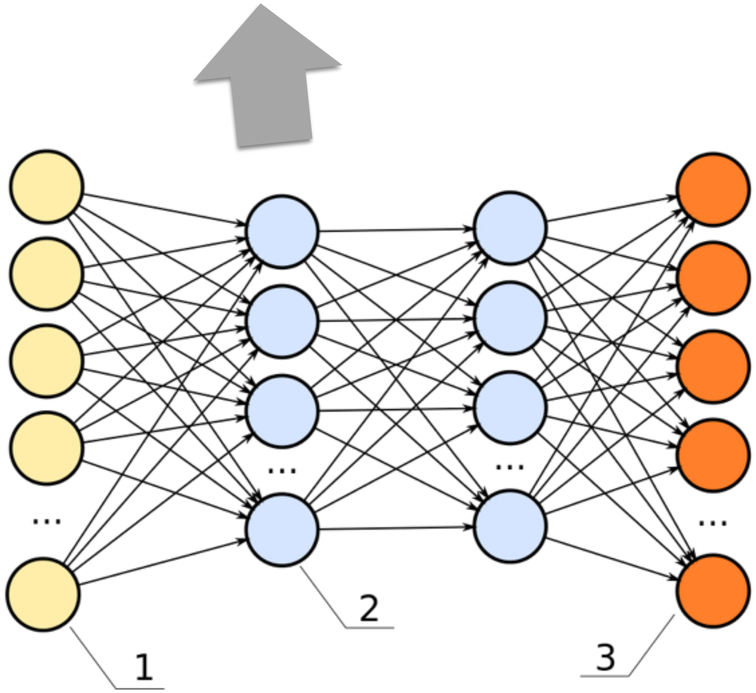




GDG DevFest TAIPEI 2016

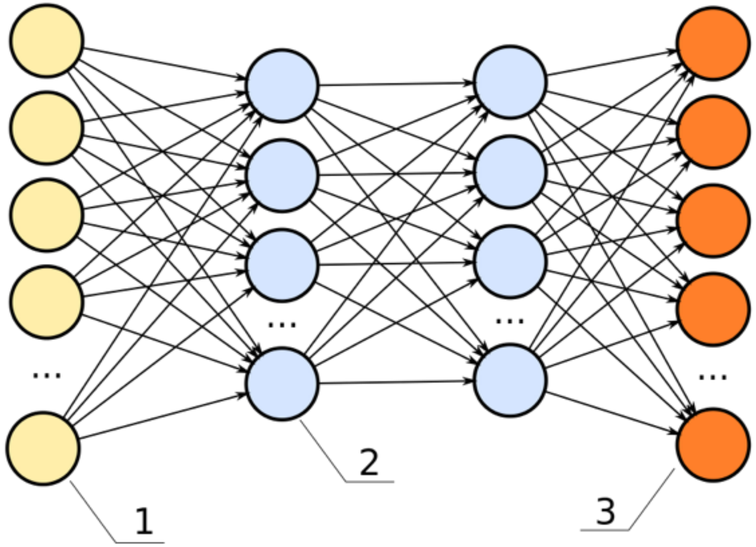
GLIACLOUD

Outline



Outline

ACTIVITYNET



TensorFlow

Activity Recognition

Dataset

- more than 648 hours of untrimmed videos
- ~20K videos
- 200 activity categories
- Youtube

Untrimmed Classification Challenge

Given a long video, predict the labels of the activities present in the video

Detection Challenge

Given a long video, predict the labels and temporal extents of the activities present in the video

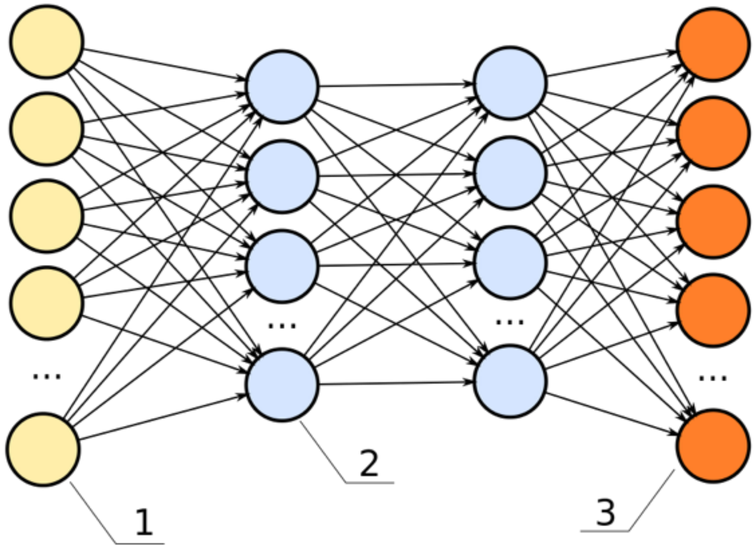
Reference : http://activity-net.org/challenges/2016/index.html

Activity Recognition

Reference : http://activity-net.org/challenges/2016/index.html

Outline

ACTIVITYNET

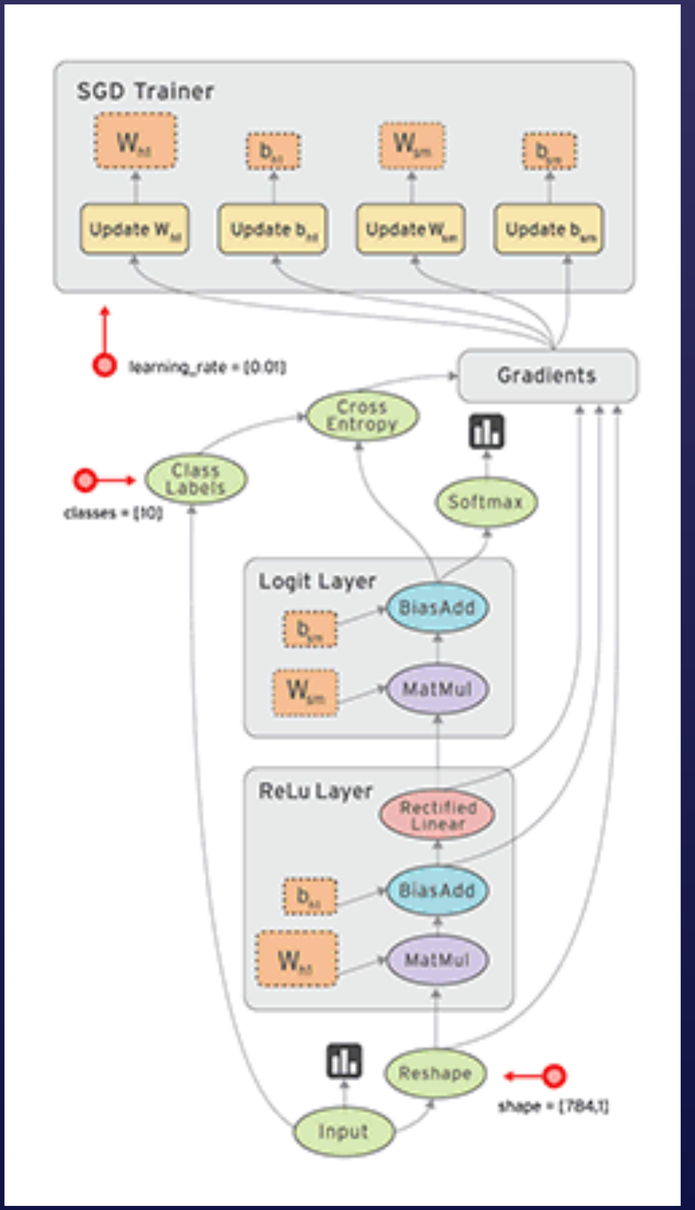


TensorFlow

TensorFlow

Library for Machine Intelligence

- Open source at 2015.11
- Deep Learning
- Numerical computation : “data flow graphs” (node: ops, edge: tensors)
- One or more CPUs or GPUs
- **Auto-Differentiation!!** (Numerical Methods)



Reference : https://www.tensorflow.org/

Inputs, Variables and Operations

Inputs - Placeholders

- Nodes for data inputs
- Static or dynamic shape

```
# tf Graph Input
X = tf.placeholder("float", shape=[None, 784])
Y = tf.placeholder("float", shape=[None, 10])
```

Variables

- Pass initial values
- Need to be initialized (later)

```
# Set model weights
W = tf.Variable(np.random.randn(784,10))
b = tf.Variable(np.random.randn(10))
```

Basic Ops

- Matrix multiply, concatenate
- add (broadcast), multiply

```
Wx = tf.matmul(X, W)
pred = tf.add(Wx, b)

# higher order
W0 = tf.random_normal([784*2, 10])
Ws = tf.Variable(W0)
X2 = tf.mul(X, X)
Xs = tf.concat(1, [X, X2])

WXs = tf.matmul(Xs, Ws)
pred = tf.add(WXs, b)
```

Reference : https://www.tensorflow.org/

How to Run

Session

- Connect the backend
- Run the graph
- tf.initialize_all_variables()
- tf.Session()
- tf.run()
- Feed data : feed_dict={}

```
init = tf.initialize_all_variables()
sess = tf.Session()
# initialize variables
with sess.as_default():
    sess.run(init)

with sess.as_default():
    print(sess.run(b))
    sess.run(init)
    print(b.eval())
```

```
[ 1.6032598  -1.3250742   0.77060711
 -0.12344709  2.22597957 -0.52275437
[-0.09863706 -2.65096951 -1.14960706
  1.18818402 -1.63062012 -0.935911
```

Reference : https://www.tensorflow.org/

What is Scope - Shared Variables

Variable Scope

- Variables within the same scope could be “reused”
- tf.get_variable(name, shape)
- tf.variable_scope()
- Manually set “reuse”: scope.reuse_variables()

```
import tensorflow as tf

with tf.variable_scope("foo") as foo_scope:
    v = tf.get_variable("v", [1])
with tf.variable_scope(foo_scope):
    w = tf.get_variable("w", [1])

# reuse variables : get them back!
with tf.variable_scope(foo_scope, reuse=True):
    v1 = tf.get_variable("v", [1])
    w1 = tf.get_variable("w", [1])

assert v1 == v
assert w1 == w
```

Reference : www.jikexueyuan.com

Example 1 : Linear Regression

Update Parameters

- tf.train.(optimizer)
- minimize(cost)
- Auto-gradient & Auto-update!!

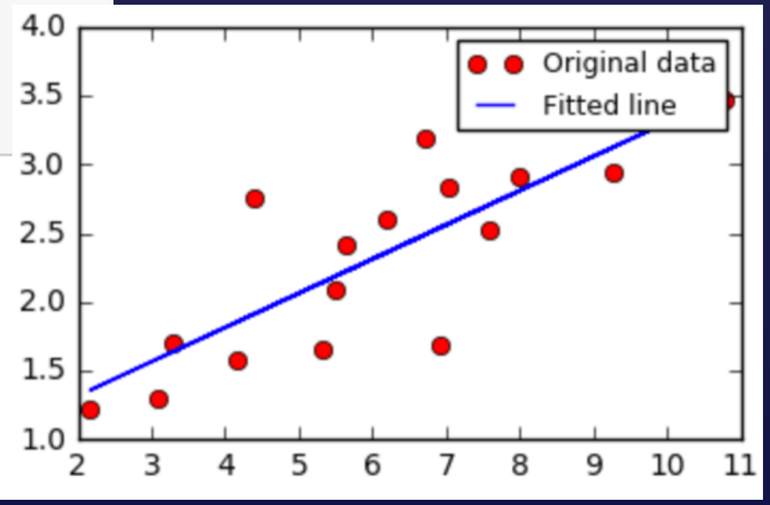
```
train_X = np.asarray([3.3,4.4,5.5,6.71,6.93,4.168,9.779,6.182,7.59,2.167,
                      7.042,10.791,5.313,7.997,5.654,9.27,3.1])
train_Y = np.asarray([1.7,2.76,2.09,3.19,1.694,1.573,3.366,2.596,2.53,1.221,
                      2.827,3.465,1.65,2.904,2.42,2.94,1.3])
n_samples = train_X.shape[0]

X = tf.placeholder("float")
Y = tf.placeholder("float")

W = tf.Variable(rng.randn(), name="weight")
b = tf.Variable(rng.randn(), name="bias")

pred = tf.add(tf.mul(X, W), b)
cost = tf.reduce_sum(tf.pow(pred-Y, 2)) / (2*n_samples)
# Gradient descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
```

Reference : https://github.com/aymericdamien/TensorFlow-Examples

Example 1 : Linear Regression

```
init = tf.initialize_all_variables()
with tf.Session() as sess:
    sess.run(init)

    for epoch in range(training_epochs):
        sess.run(optimizer, feed_dict={X: train_X, Y: train_Y})
        log(epoch, display_step, sess, cost, train_X, train_Y, W, b)

    training_cost = sess.run(cost, feed_dict={X: train_X, Y: train_Y})
    print("Training cost=", training_cost, "W=",
          sess.run(W), "b=", sess.run(b), '\n')

    plot_graph(train_X, train_Y, sess, W, b)
```

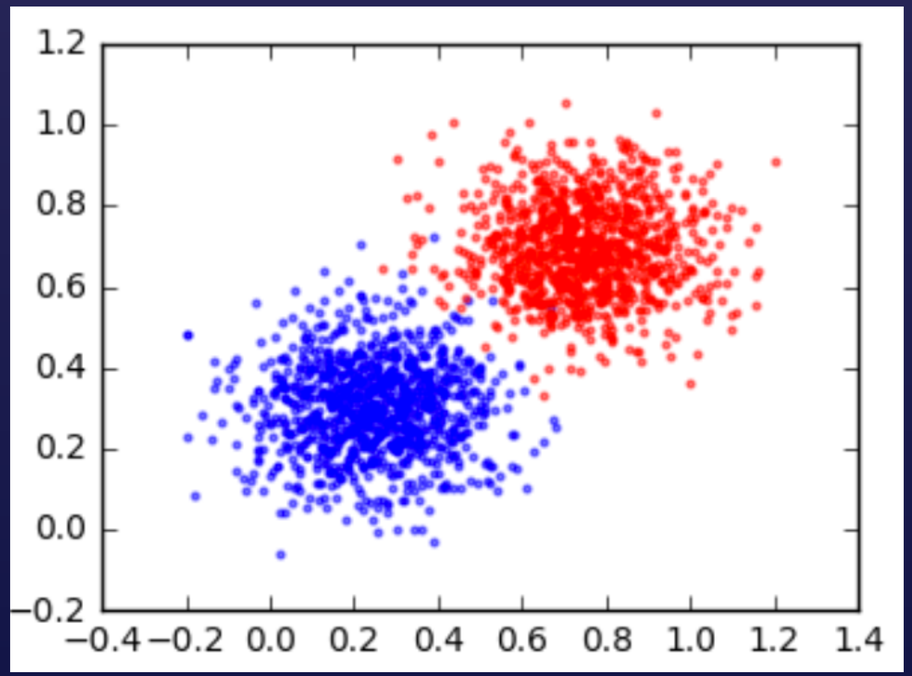
```
Epoch: 2000 cost= 0.088393934 W= 0.190929 b= 1.22918
Epoch: 4000 cost= 0.079599813 W= 0.222334 b= 1.00653
Epoch: 6000 cost= 0.077551104 W= 0.237493 b= 0.899064
Epoch: 8000 cost= 0.077073790 W= 0.244809 b= 0.847195
Epoch: 10000 cost= 0.076962613 W= 0.24834 b= 0.82216
Optimization Finished!
Training cost= 0.0769626 W= 0.24834 b= 0.82216
```



Reference : https://github.com/aymericdamien/TensorFlow-Examples

Example II : Binary Classification

Data



Logistic Regression

- Loss function :

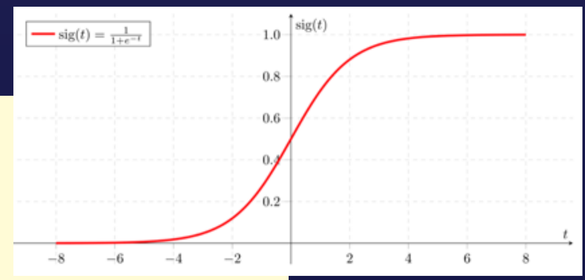
$$\min_{\mathbf{w}} \quad E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)\right)$$

- Decision :

logistic regression: use

$$h(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

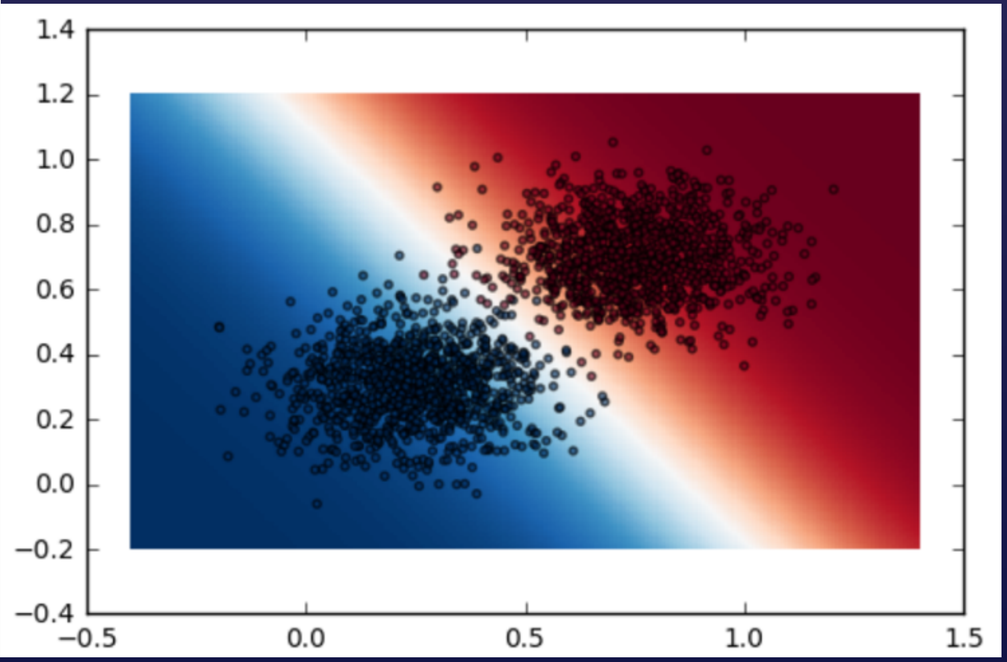
to approximate target function $f(\mathbf{x}) = P(+1|\mathbf{x})$



Reference : https://www.csie.ntu.edu.tw/~htlin/course/mlfound16fall/doc/10_handout.pdf

Example II : Binary Classification

```
# tf Graph Input - dynamic shape
x = tf.placeholder(tf.float32, [None, 2])
y = tf.placeholder(tf.float32, [None, 1])

# Set model weights
W = tf.Variable(tf.zeros([2, 1]))
b = tf.Variable(tf.zeros([1]))

# Construct model
exp_a1 = tf.exp(tf.mul((tf.matmul(x, W) + b), -y)) + 1
pred = tf.nn.sigmoid(tf.add(tf.matmul(x, W), b))

cost = tf.reduce_mean(tf.reduce_sum(tf.log(exp_a1),
                                    reduction_indices=1))
# Gradient Descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate)
optimizer = optimizer.minimize(cost)
```



Reference : https://github.com/aymericdamien/TensorFlow-Examples

Using Multicores

Assign device

- tf.device()
- Matrix multiplication
- (500, 1000) * (1000, 500)

```
Using cpu: 0:00:00.239567
```

```
Using gpu: 0:00:00.030540
```

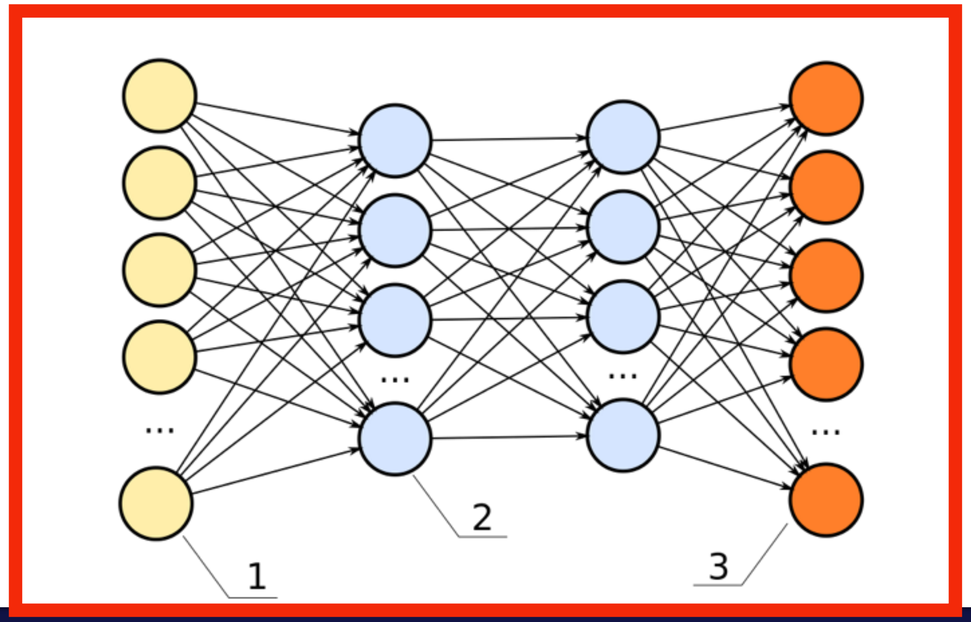
```
# CPU
with tf.device('/cpu:0'):
    a = tf.constant(A)
    b = tf.constant(B)
    c = tf.matmul(a, b)

start = datetime.now()
with tf.Session() as sess:
    sess.run(c)
    end = datetime.now()
    print('Using cpu: %s' % str(end - start))
```

```
# GPU
with tf.device('/gpu:0'):
    a = tf.constant(A)
    b = tf.constant(B)
    c = tf.matmul(a, b)

start = datetime.now()
with tf.Session() as sess:
    sess.run(c)
    end = datetime.now()
    print('Using gpu: %s' % str(end - start))
```

Reference : https://github.com/aymericdamien/TensorFlow-Examples

Outline

ACTIVITYNET



TensorFlow

Machine Learning Framework

Hypothesis	Inference	Training
F(x) -> y	x -> F(x)	Data -> “Best” F
Function Approximation	Given input, predict y	Define and Minimize “Loss”
Define Hypothesis Function Set	Testing	“Best” F: over “new” data

Deep Learning

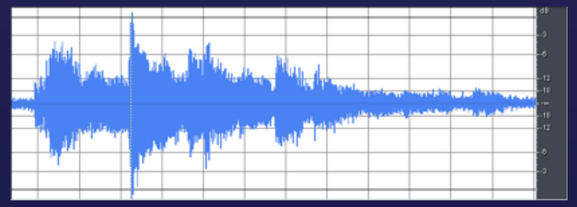
A set of very complex functions

- Based on “neuron” and being “deep”
- **Architecture** & parameters

Course by
Hung-Yi Lee

http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD15_2.html

f() = “5”

f() = “Hi good day”

f() = “dog run in the water”

Neural Network

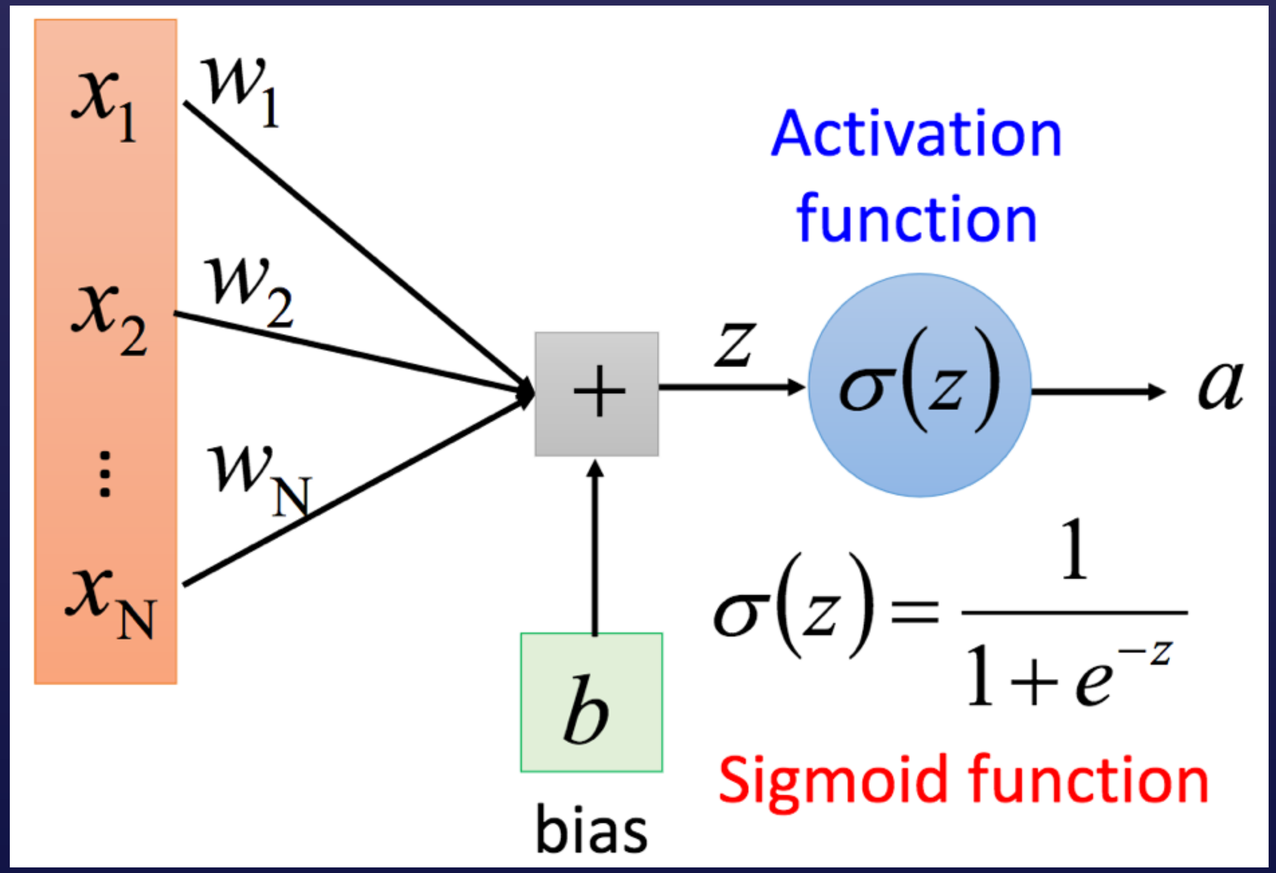
Logistic Regression

- Just a single neuron with "sigmoid" activation function

Many neurons ?

Different activation functions ?

More layers (Cascade) ?



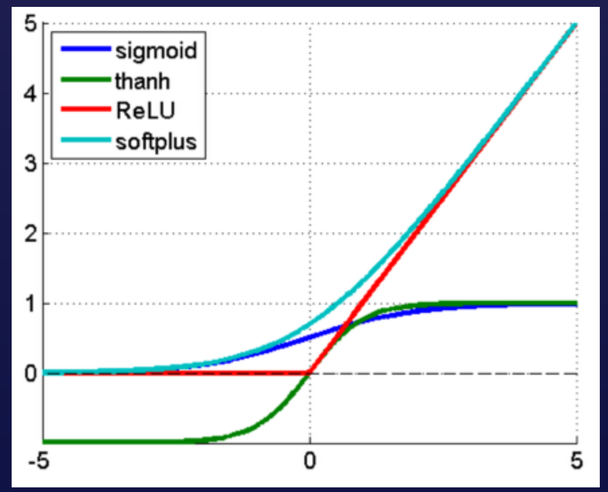
Reference : http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD15_2.html

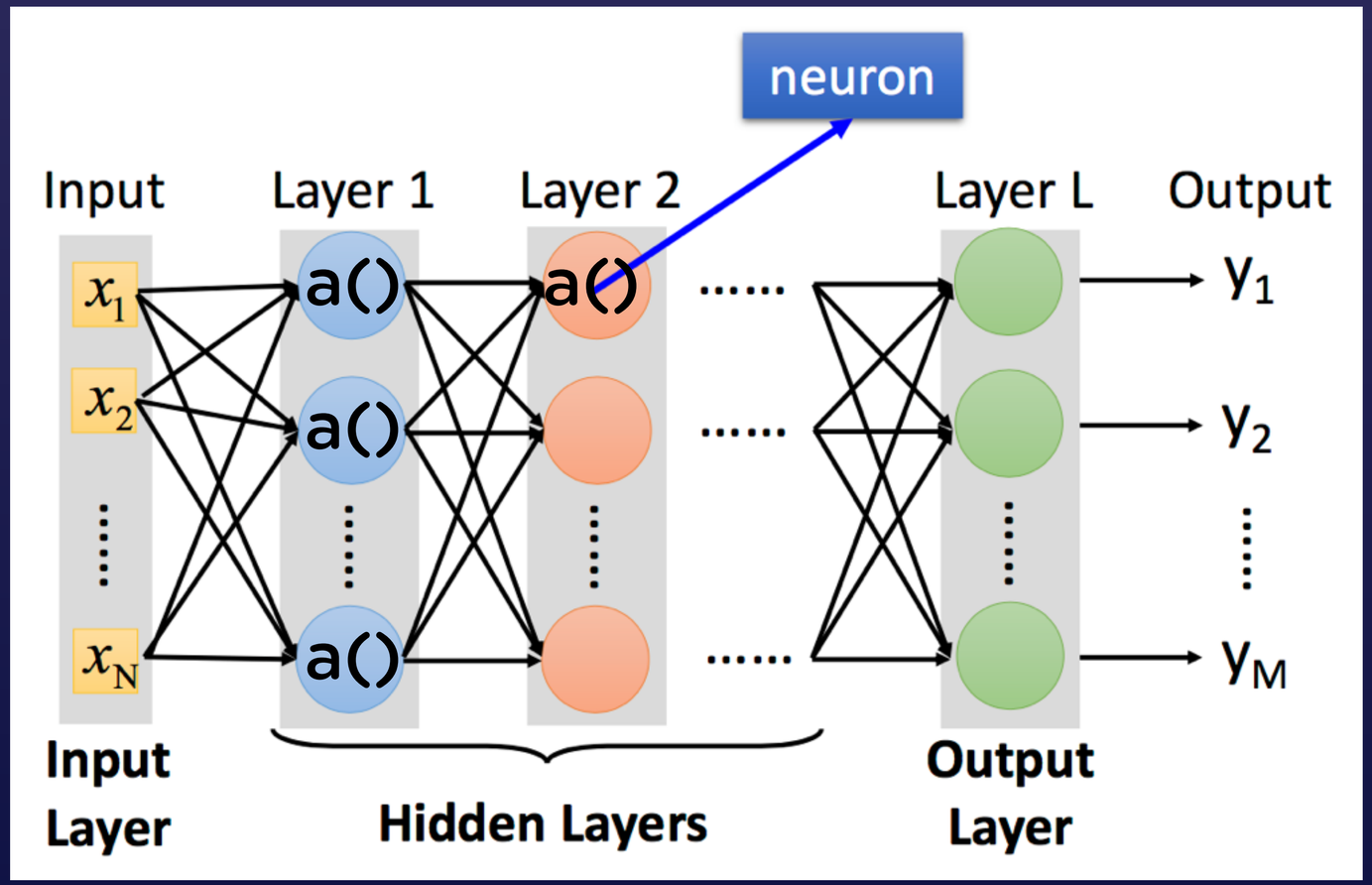
Deep Neural Network

Very Rich Set of Hypothesis !

Fully Connected

- Deep: many hidden layers
- Hidden layer: many neurons
- Neuron: non-linear transform of weighted sum of inputs
- Transform:





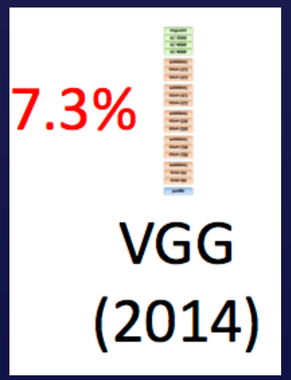
Reference : http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML16.html
https://imiloainf.wordpress.com/2013/11/06/rectifier-nonlinearities/

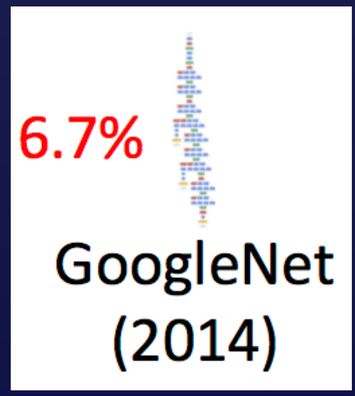
Deep Neural Network

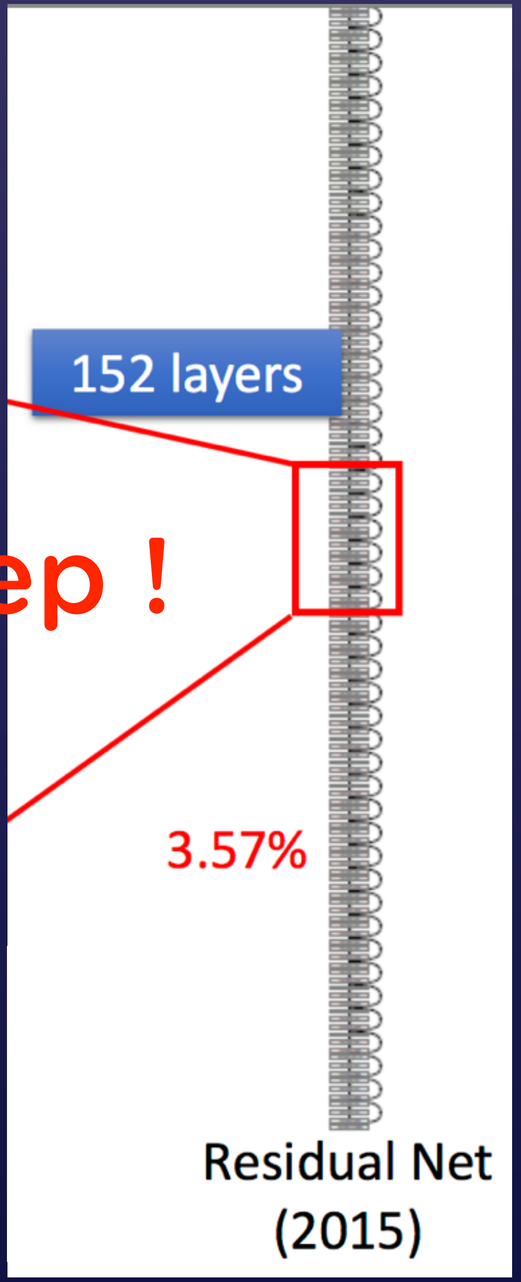
ImageNet

Large Scale Visual Recognition Challenge (ILSVRC)

- Classification: 1000 categories
- Human error rate: 5.1%







Reference : http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML16.html

TensorFlow Playground

Reference : http://playground.tensorflow.org

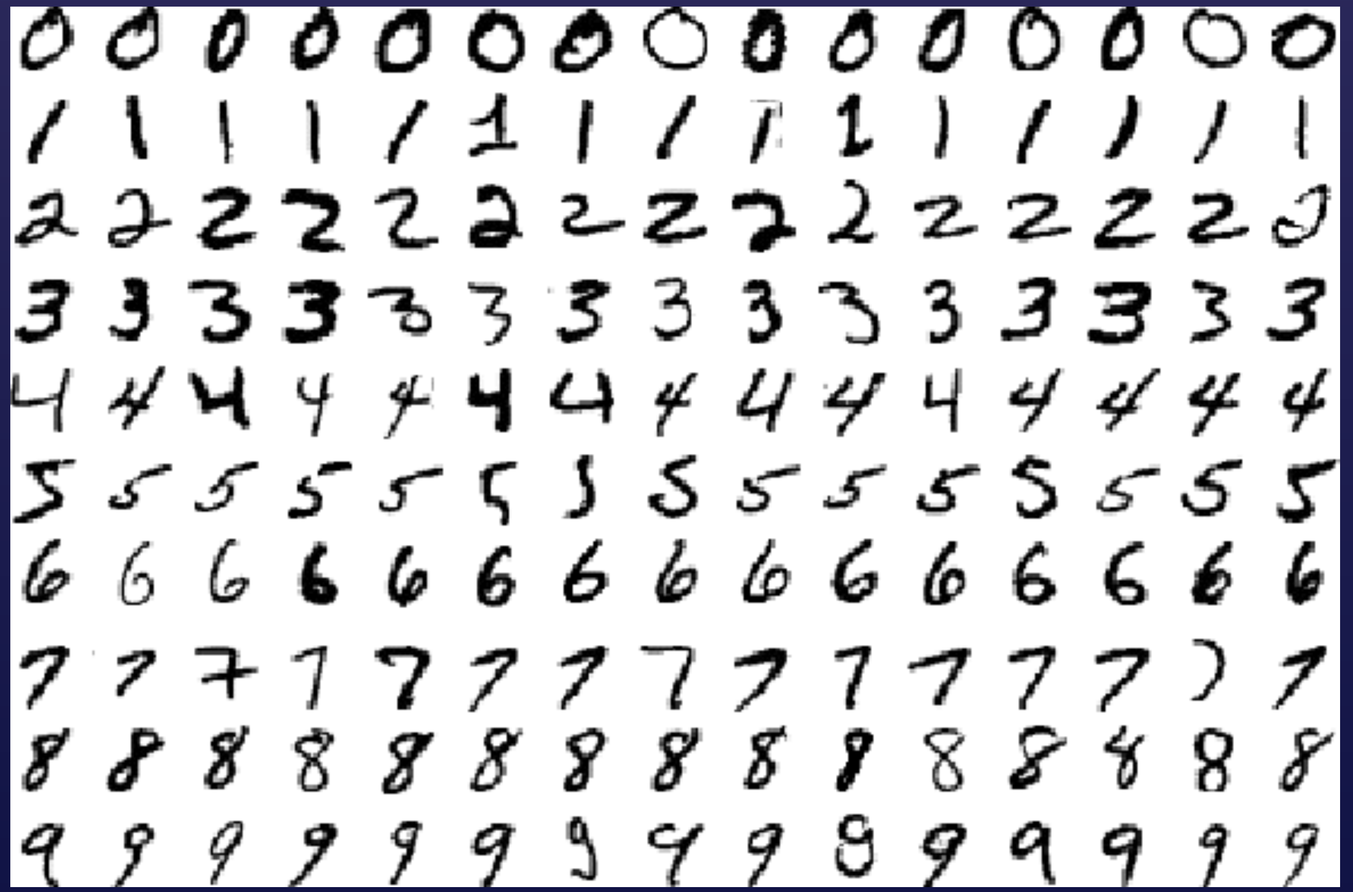
GDG DevFest TAIPEI 2016

GLIACLOUD

DNN Example : Autoencoder

MNIST

- For hand-written digits recognition
- Each digit is a 28 * 28 BW images (0.0 - 1.0 array)
- 10 categories: 0 - 9



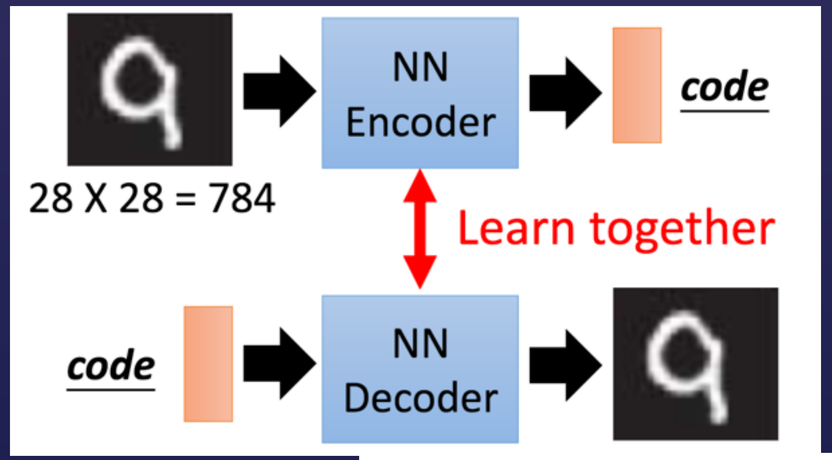
- Demo: https://www.youtube.com/watch?v=FwFduRA_L6Q

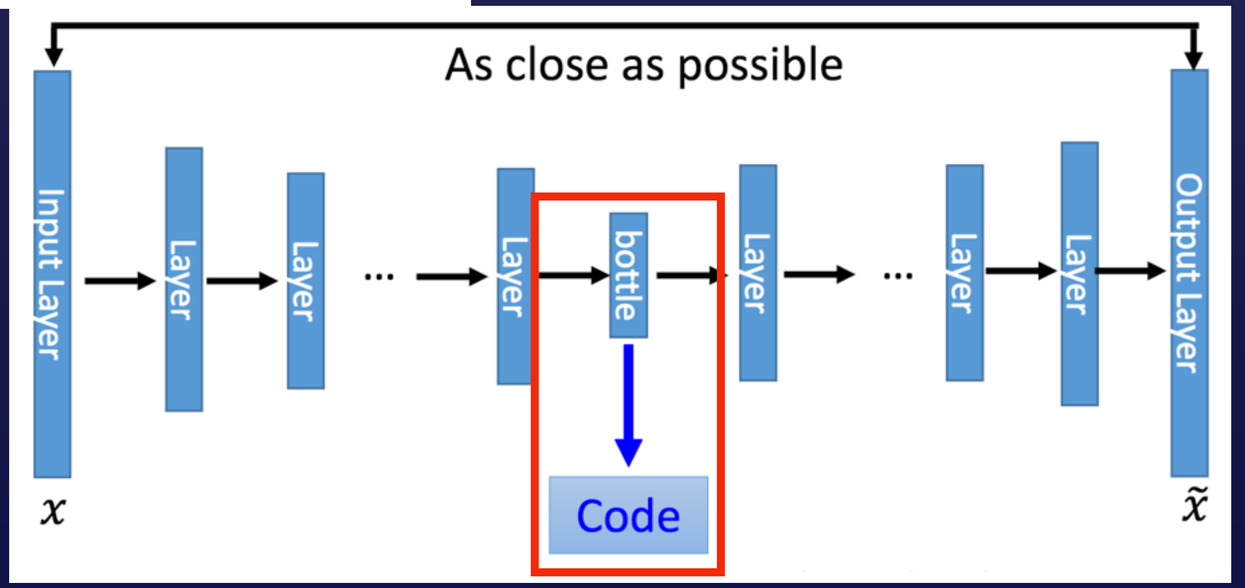
Reference : https://www.tensorflow.org/

DNN Example : Autoencoder

Auto-encoder

- Deep Neural Network as “Encoder” & “Decoder”
- Unsupervised (Dimension reduction)
- “Representation” of the image





Reference : http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD15_2.html

DNN Example : Autoencoder

Encoder

```
def encoder(x, scope, n_input, n_hidden, n_code, n_layer):
    with tf.variable_scope(scope):
        W = tf.get_variable("W_in", [n_input, n_hidden])
        b = tf.get_variable("b_in", [n_hidden])
        layer_i = tf.nn.tanh(tf.add(tf.matmul(x, W), b))

        for i in range(n_layer - 1):
            W = tf.get_variable("W_en_%d" % i, [n_hidden, n_hidden])
            b = tf.get_variable("b_en_%d" % i, [n_hidden])
            layer_i = tf.nn.tanh(tf.add(tf.matmul(layer_i, W), b))

        W = tf.get_variable("W_encode", [n_hidden, n_code])
        b = tf.get_variable("b_encode", [n_code])
        code = tf.nn.tanh(tf.add(tf.matmul(layer_i, W), b))
    return code
```

Reference : https://github.com/AaronYALai/tensorflow_activity_recognition

DNN Example : Autoencoder

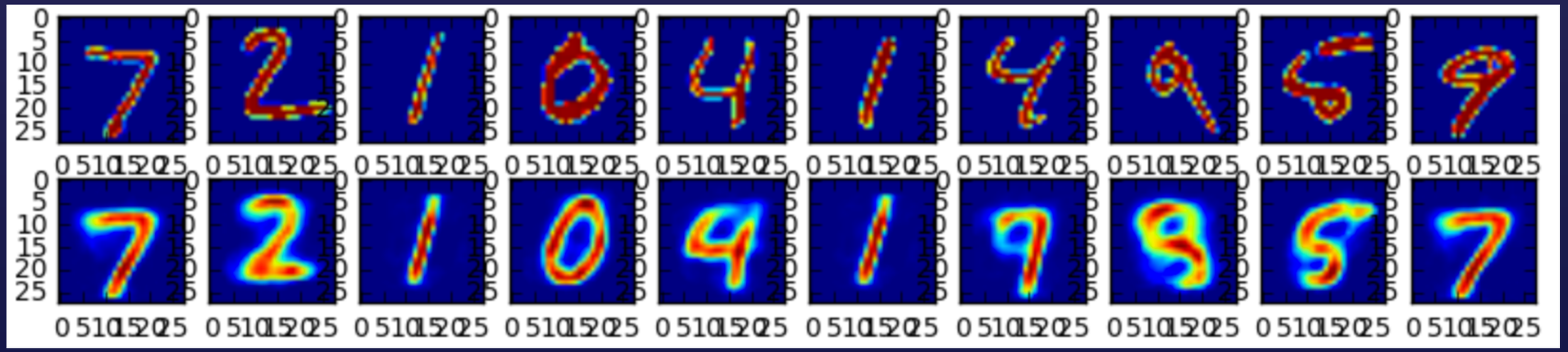
Decoder

```
def decoder(code, scope, n_output, n_hidden, n_code, n_layer):
    with tf.variable_scope(scope):
        W = tf.get_variable("W_decode", [n_code, n_hidden])
        b = tf.get_variable("b_decode", [n_hidden])
        layer_i = tf.nn.tanh(tf.add(tf.matmul(code, W), b))

        for i in range(n_layer - 1):
            W = tf.get_variable("W_de_%d" % i, [n_hidden, n_hidden])
            b = tf.get_variable("b_de_%d" % i, [n_hidden])
            layer_i = tf.nn.tanh(tf.add(tf.matmul(layer_i, W), b))

        W = tf.get_variable("W_out", [n_hidden, n_output])
        b = tf.get_variable("b_out", [n_output])
        output = tf.nn.sigmoid(tf.add(tf.matmul(layer_i, W), b))
    return output
```

Reference : https://github.com/AaronYALai/tensorflow_activity_recognition

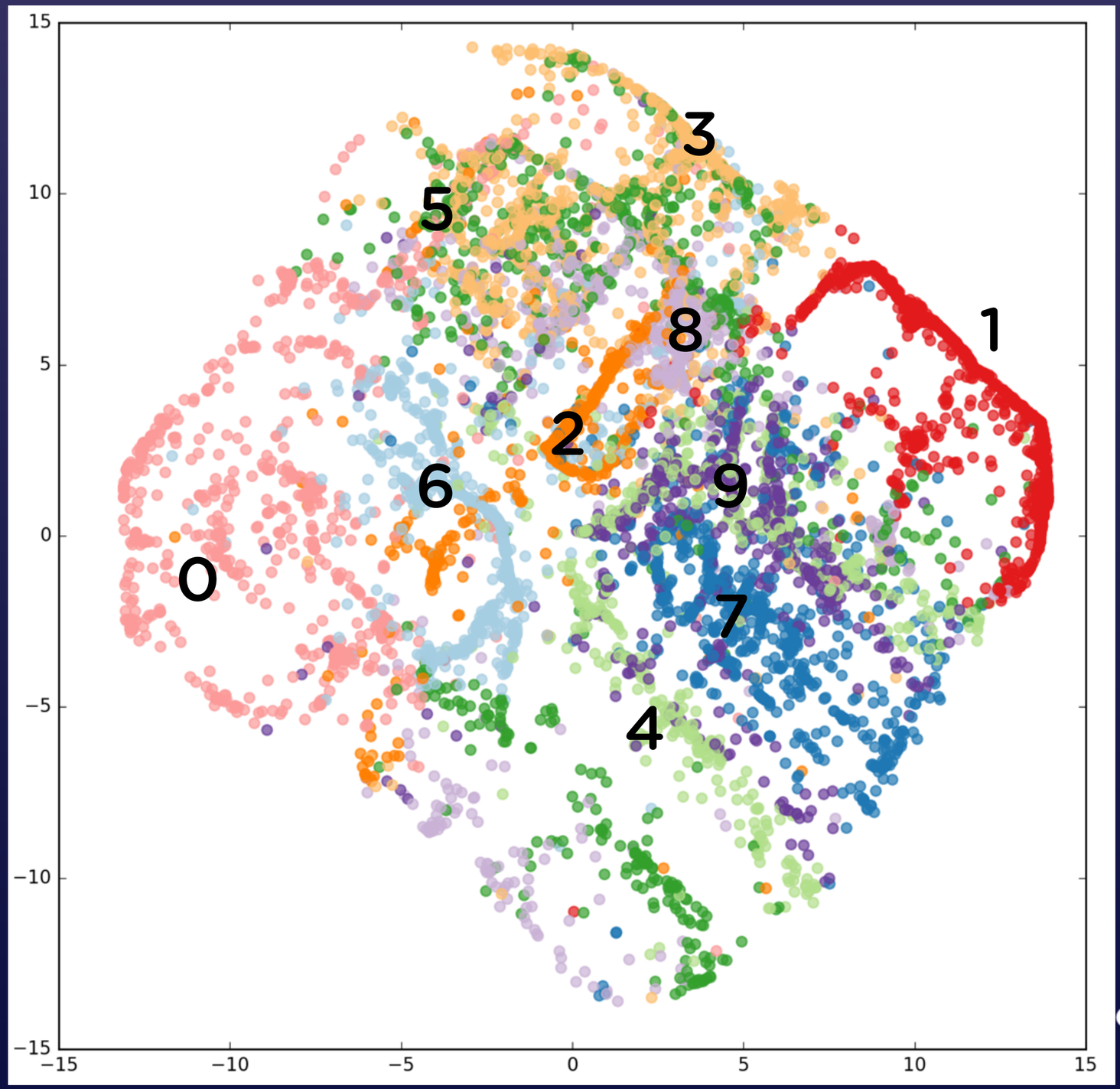
Autoencoder

Results



Reference : https://github.com/AaronYALai/tensorflow_activity_recognition

Autoencoder

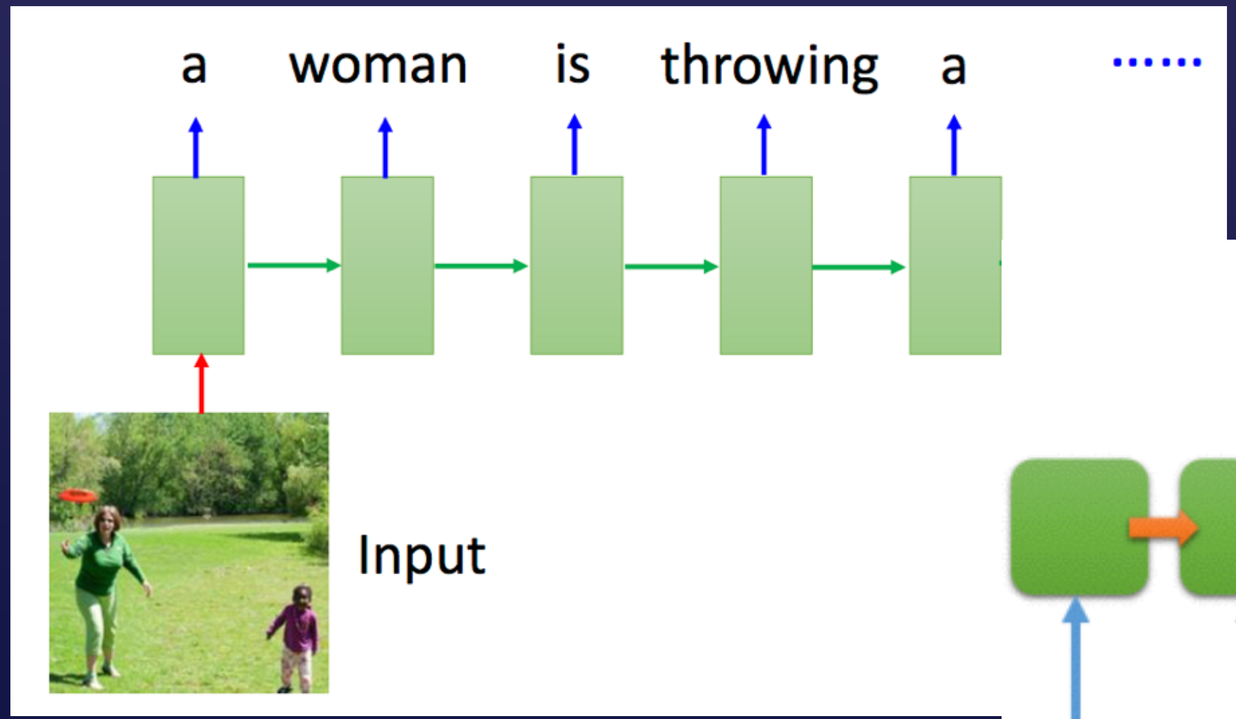


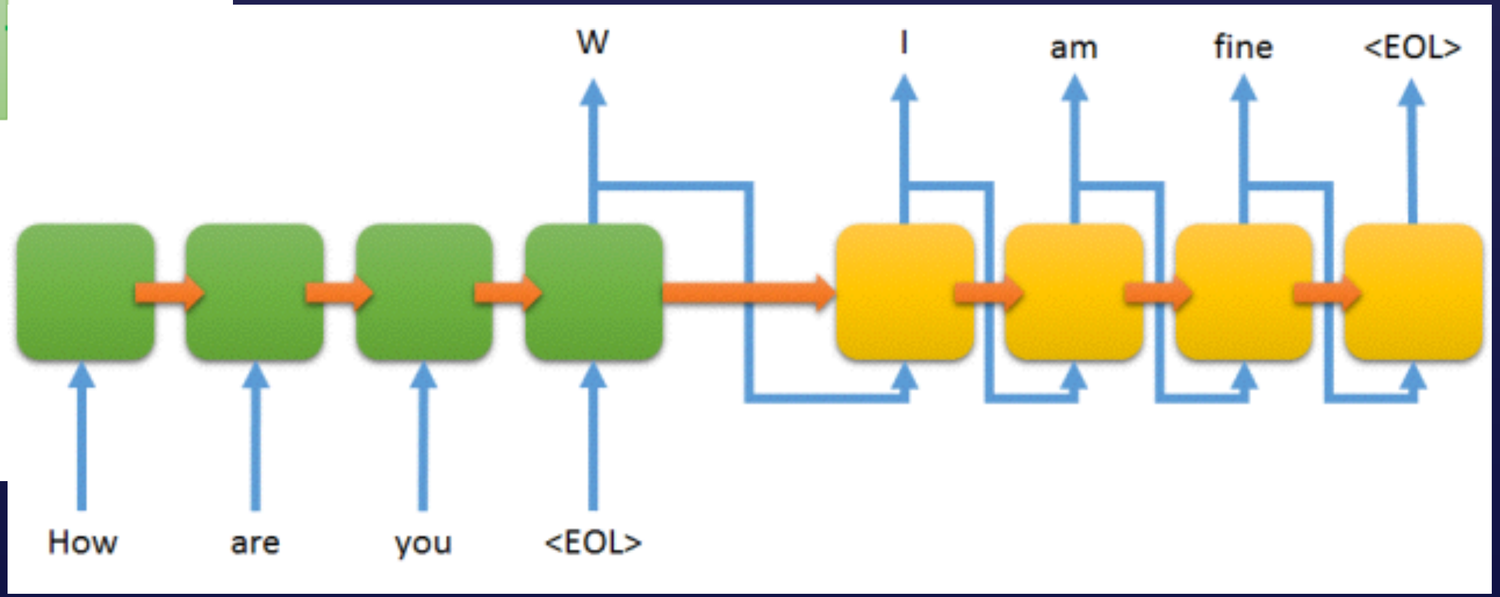
Reference : https://github.com/AaronYALai/tensorflow_activity_recognition

OUD

Neural Network with Memory

Memory is important for sequence tasks



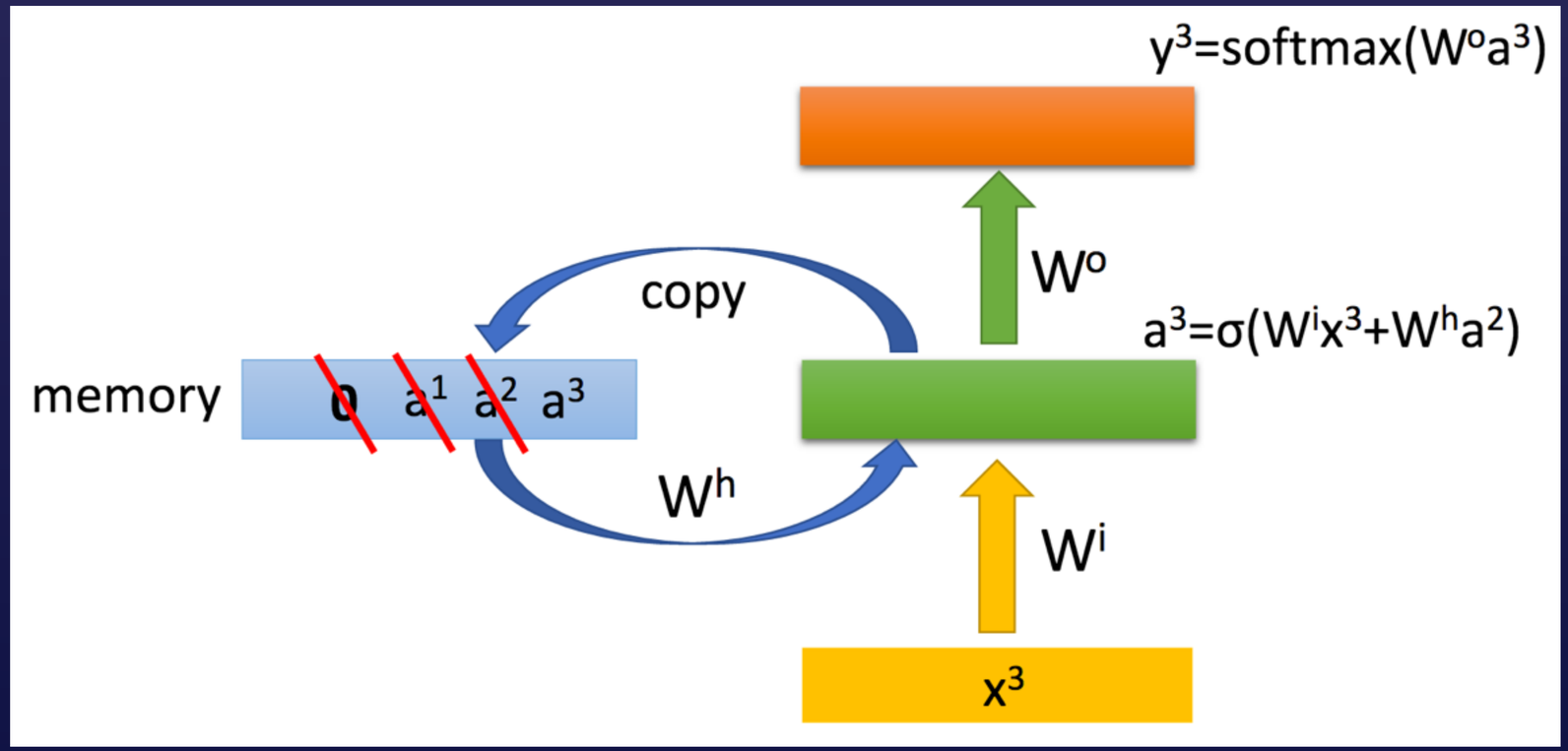


Reference : http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2
https://github.com/farizrahman4u/seq2seq

Neural Network with Memory

Recurrent Neural Network

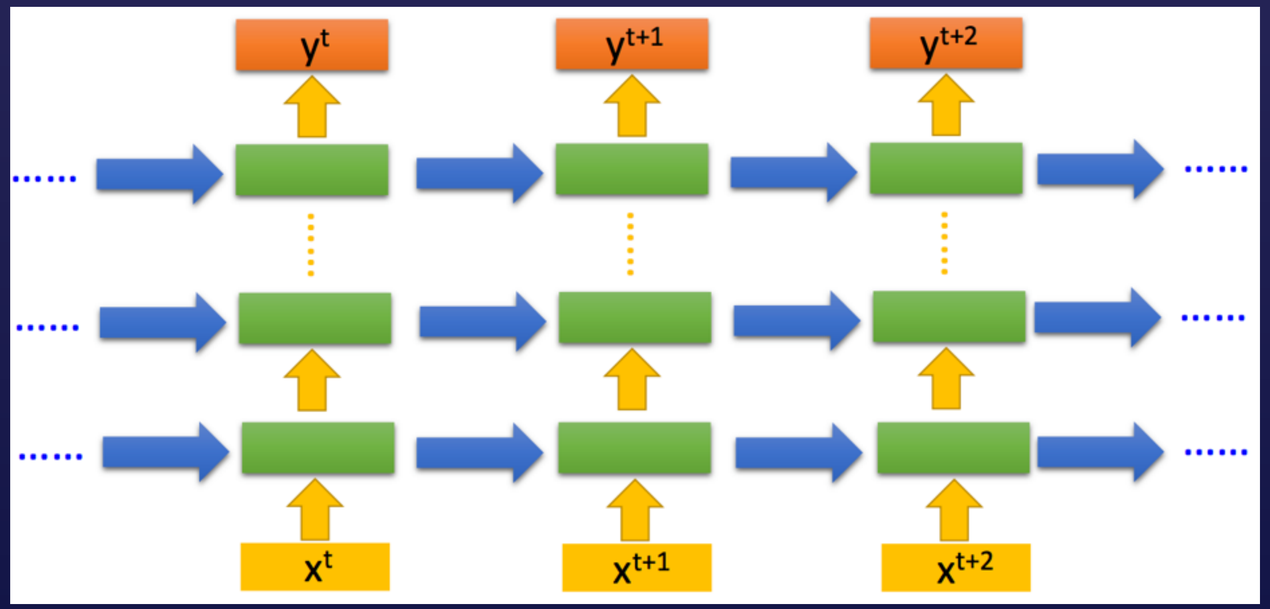
- Memory cell: pass and update at every step
- “Reuse”: W_o, W_i, and W_h are shared
- “Sequence”: feature vectors are input “one by one”
- Hard to Train: df/dW? Autogeadient!



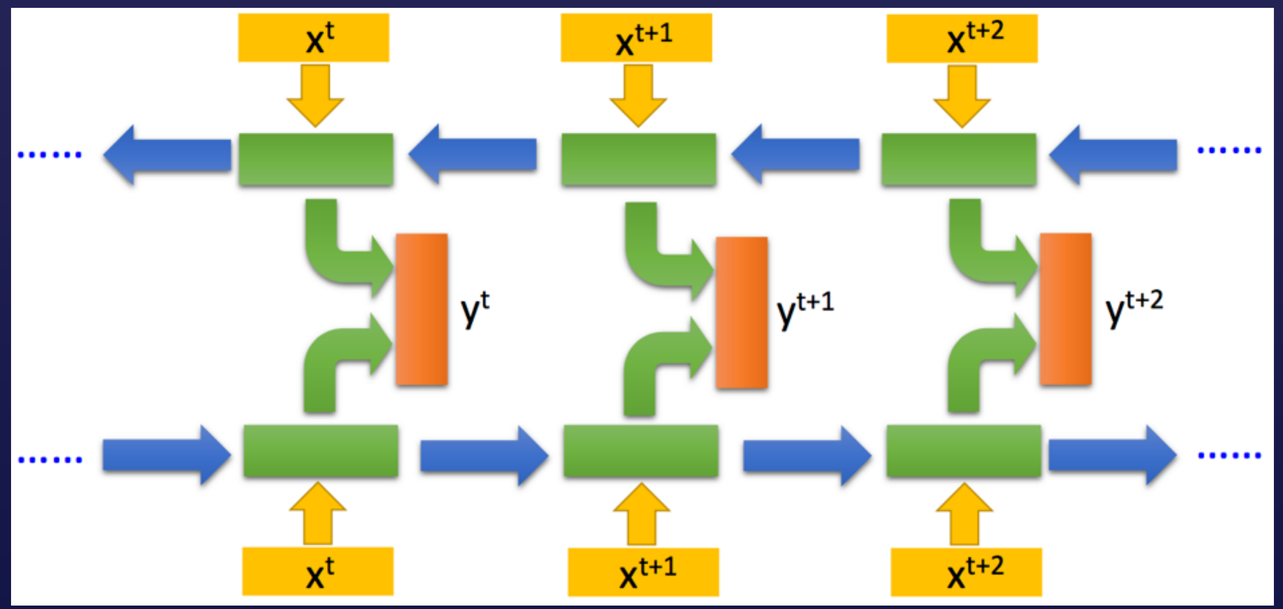
Reference : http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD15_2.html

Recurrent Neural Network

Go Deep!



Bidirectional



Reference : http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD15_2.html

Sample Code

- RNN on MNIST
- Break 28*28 image into 28 vectors (seq)
- Output the last one

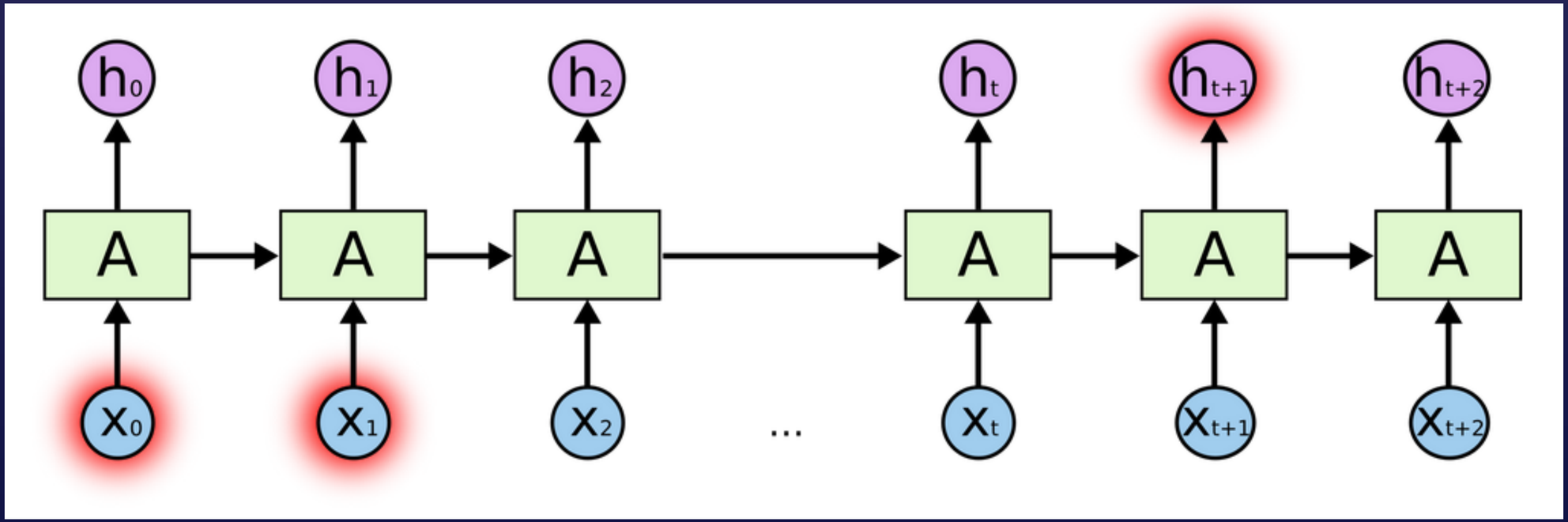
```
Start
Iter 256, Loss= 2.118008, Accuracy= 0.31641
Iter 12800, Loss= 0.369384, Accuracy= 0.89453
Iter 25600, Loss= 0.383506, Accuracy= 0.87109
Iter 38400, Loss= 0.284901, Accuracy= 0.92578
Iter 51200, Loss= 0.220428, Accuracy= 0.92578
Iter 64000, Loss= 0.216666, Accuracy= 0.94141
Iter 76800, Loss= 0.246528, Accuracy= 0.92969
Iter 89600, Loss= 0.224747, Accuracy= 0.92578
Optimization Finished!
Testing Accuracy: 0.96875
```

```python
def RNN(x, n_input, n_hidden, n_classes, n_steps, scope=None):
    with tf.variable_scope(scope or "RNN"):
        # x:[batchsize, n_steps, n_input]; Permuting batch_size and n_steps
        x = tf.transpose(x, [1, 0, 2])
        # Reshape to (n_steps*batch_size, n_input)
        x = tf.reshape(x, [-1, n_input])
        # Split: list of 'n_steps' tensors of shape (batch_size, n_input)
        x = tf.split(0, n_steps, x)

        outputs = []
        memory = tf.get_variable("memory", [1, n_hidden])
        for i in range(len(x)):
            W_o = tf.get_variable("W_out", [n_hidden, n_classes])
            W_h = tf.get_variable("W_mem", [n_hidden, n_hidden])
            W_i = tf.get_variable("W_in", [n_input, n_hidden])

            memory_read = tf.matmul(memory, W_h)
            memory = tf.nn.tanh(tf.add(tf.matmul(x[i], W_i), memory_read))
            outputs.append(tf.matmul(memory, W_o))

            # set the variables to be shared !
            if i == 0:
                tf.get_variable_scope().reuse_variables()

    return outputs[-1]
```

Reference : https://github.com/AaronYALai/tensorflow_activity_recognition

Recurrent Neural Network

However, RNN update memory at each step
=> Cannot “connect the dots”

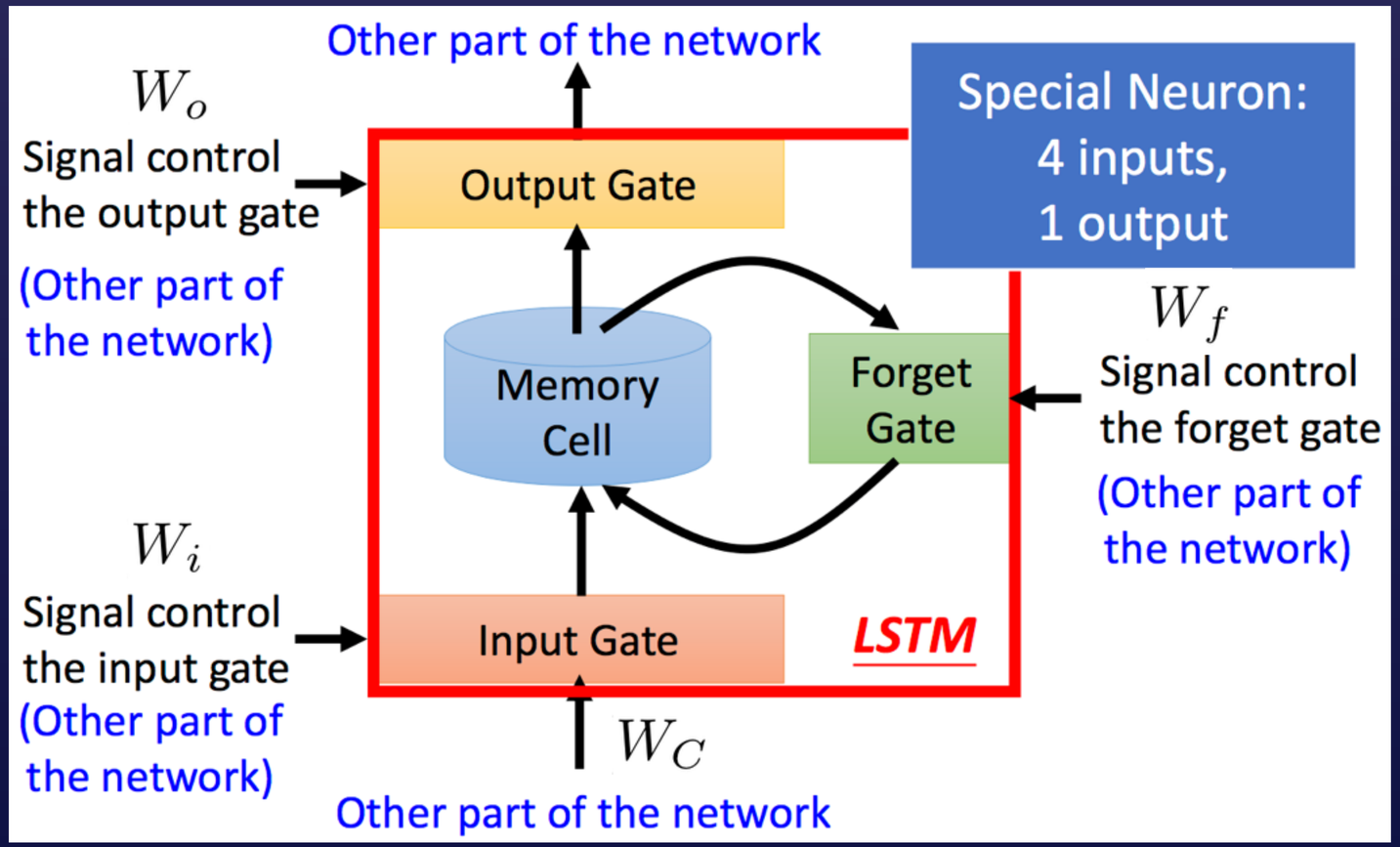


Reference : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Neural Network with Memory

Long Short-Term Memory (LSTM)

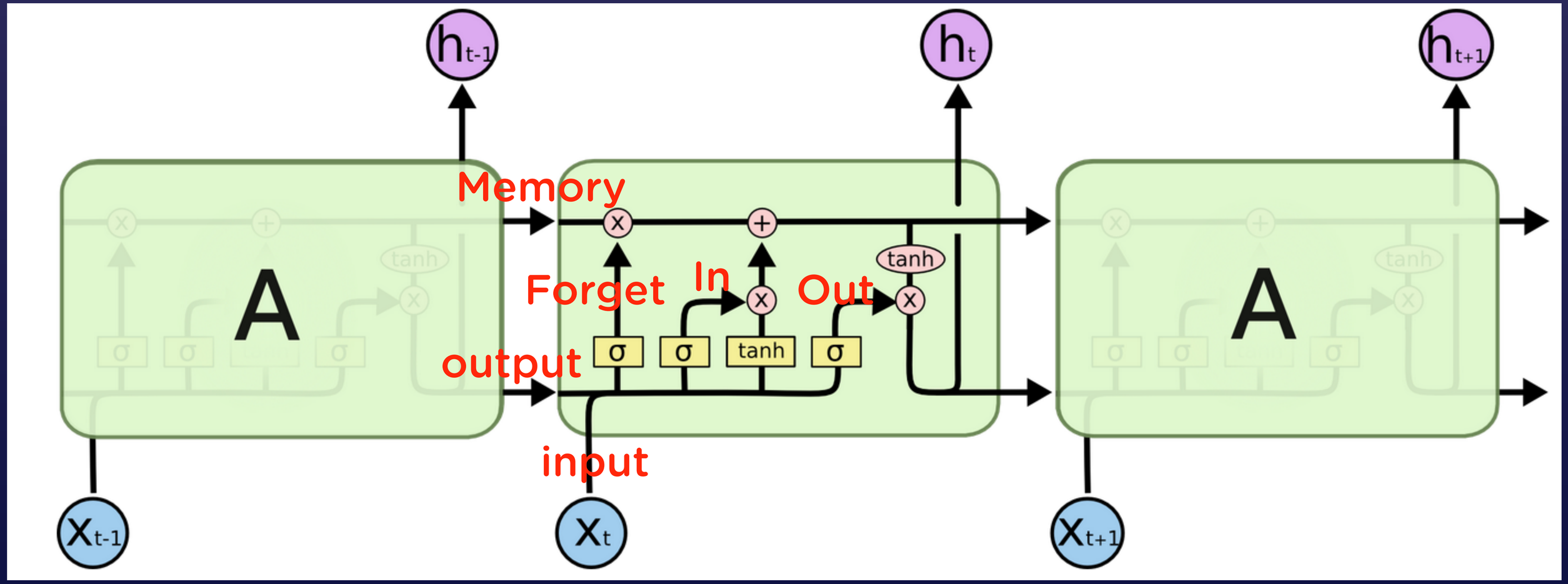
- Gate Mechanism: Control the output, update or forget of the memory
- Memory cell
- “Reuse”: W_o, W_i, W_f and W_c are shared
- “Sequence”: feature vectors are input “one by one”



Reference : http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD15_2.html

Long Short-Term Memory

Another View

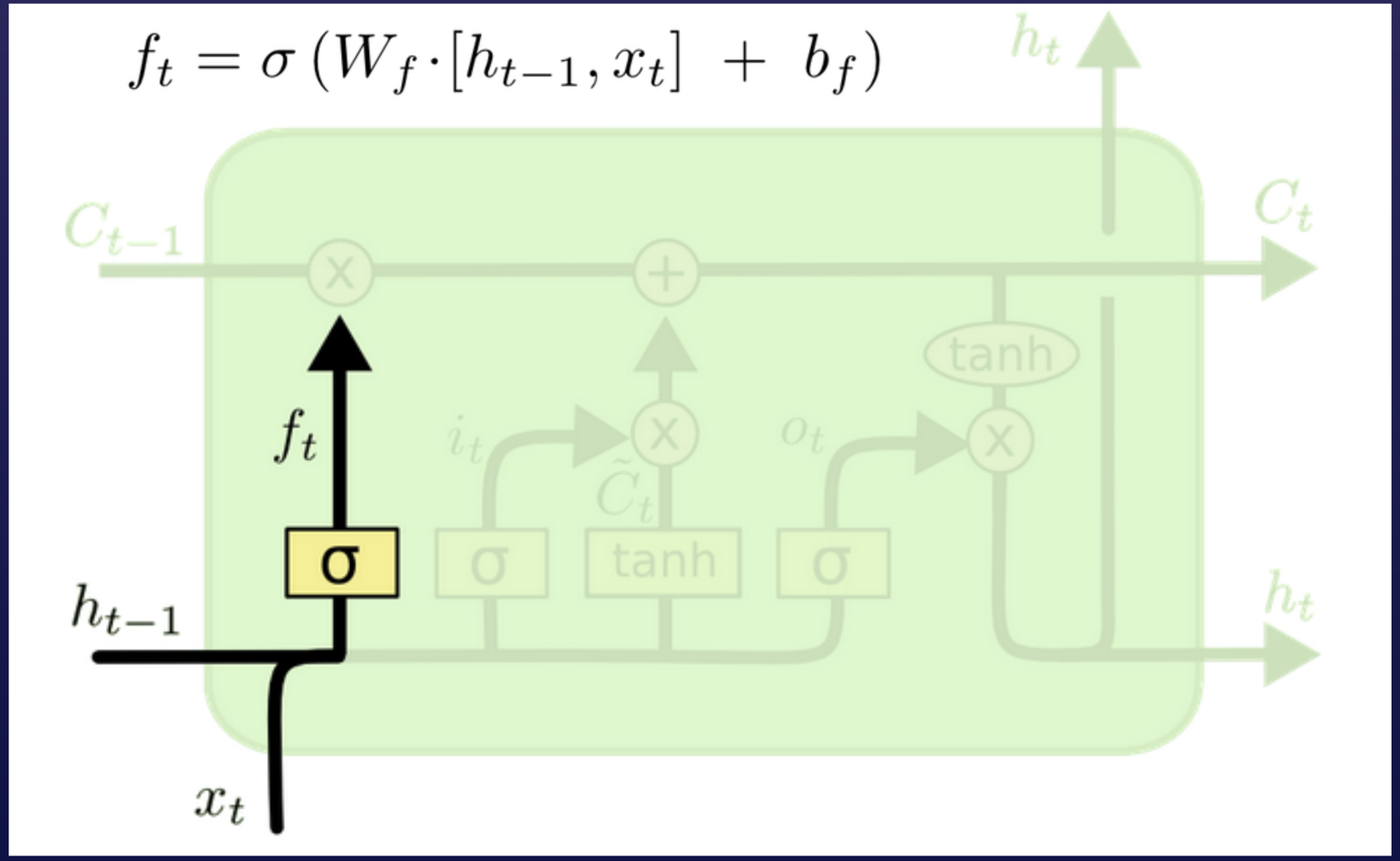


Reference : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory

Forget Step

- Refer to input features of this step and the output of last step
- Decide how much memory to forget !
- C_(t-1) * f_t : memory remained

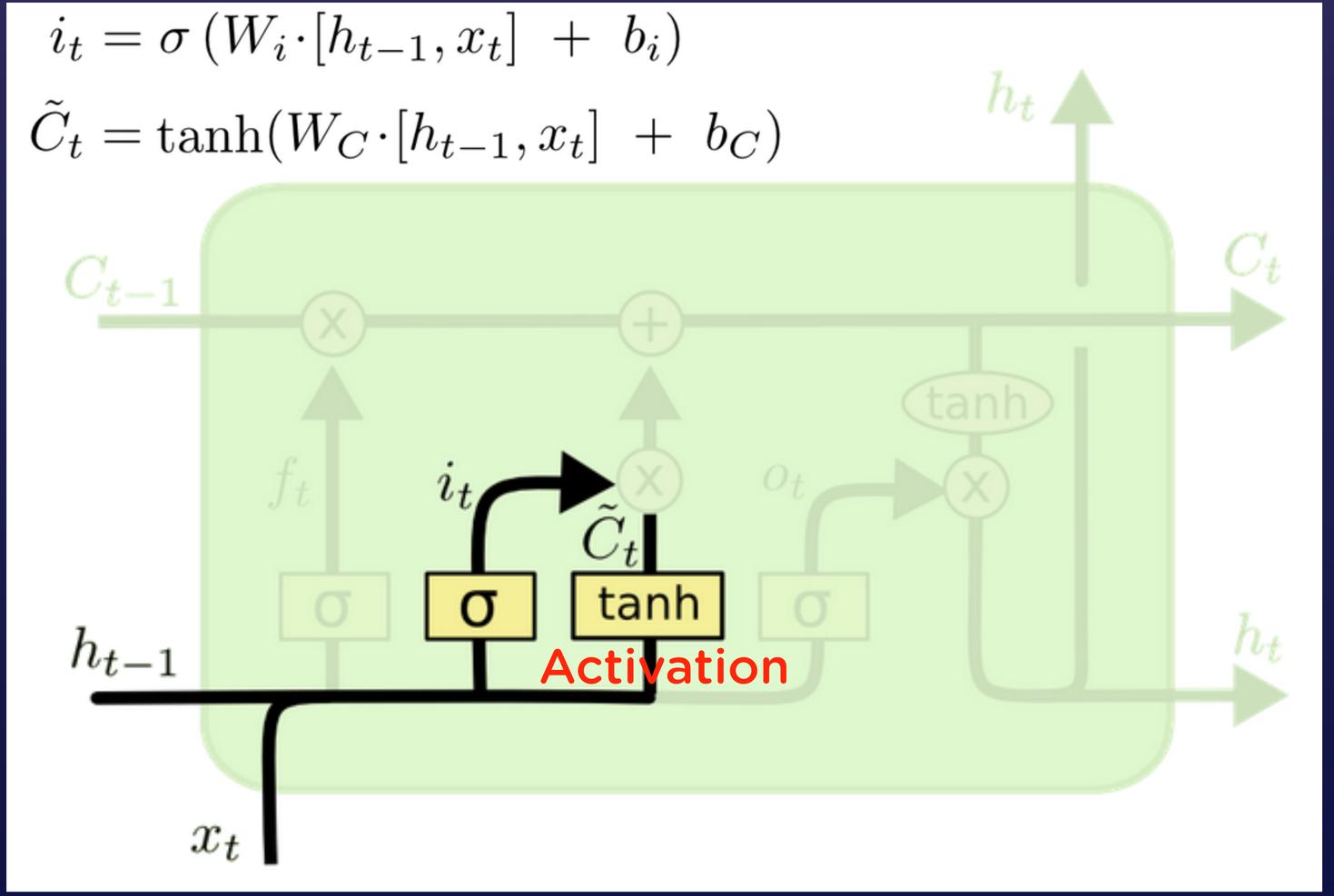


Reference : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory

Input Step

- Refer to x_t and h_(t-1)
- Compute “new” info
- Decide how much new information to get in !
- i_t * C’_t : new info to add to memory

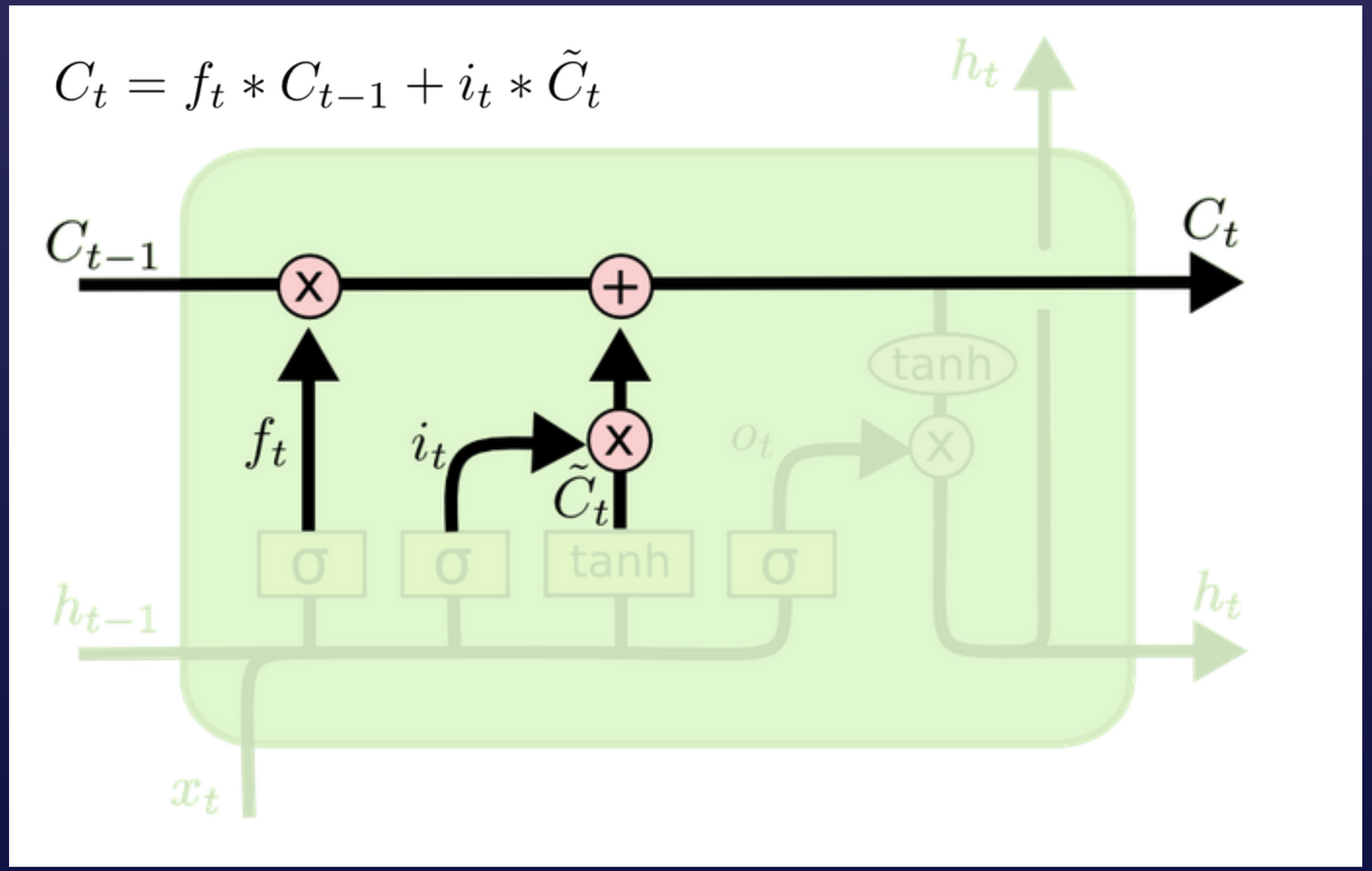


Reference : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory

Update Step

- Update the memory
- Survived “old” memory plus passed “new” info

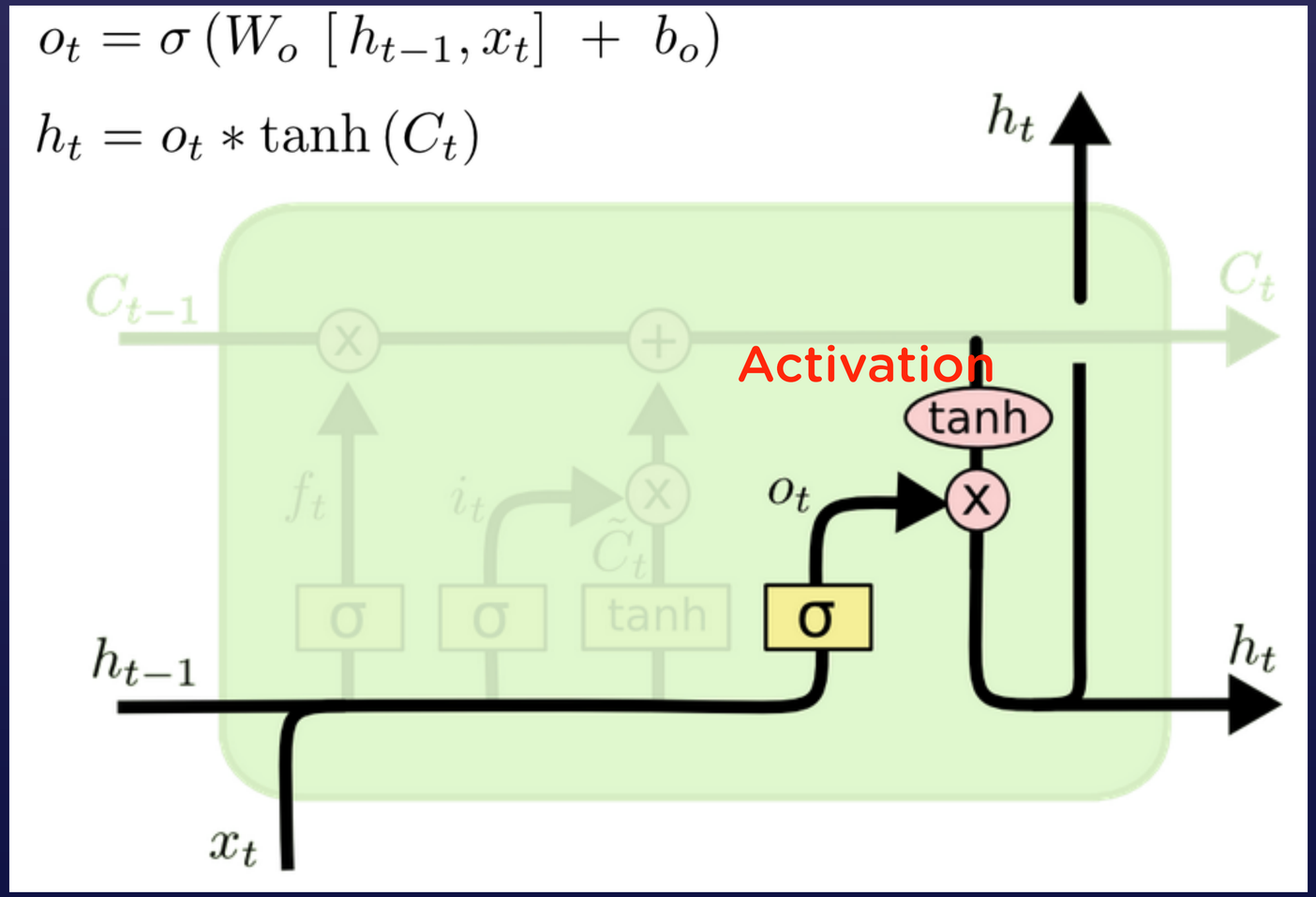


Reference : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory

Output Step

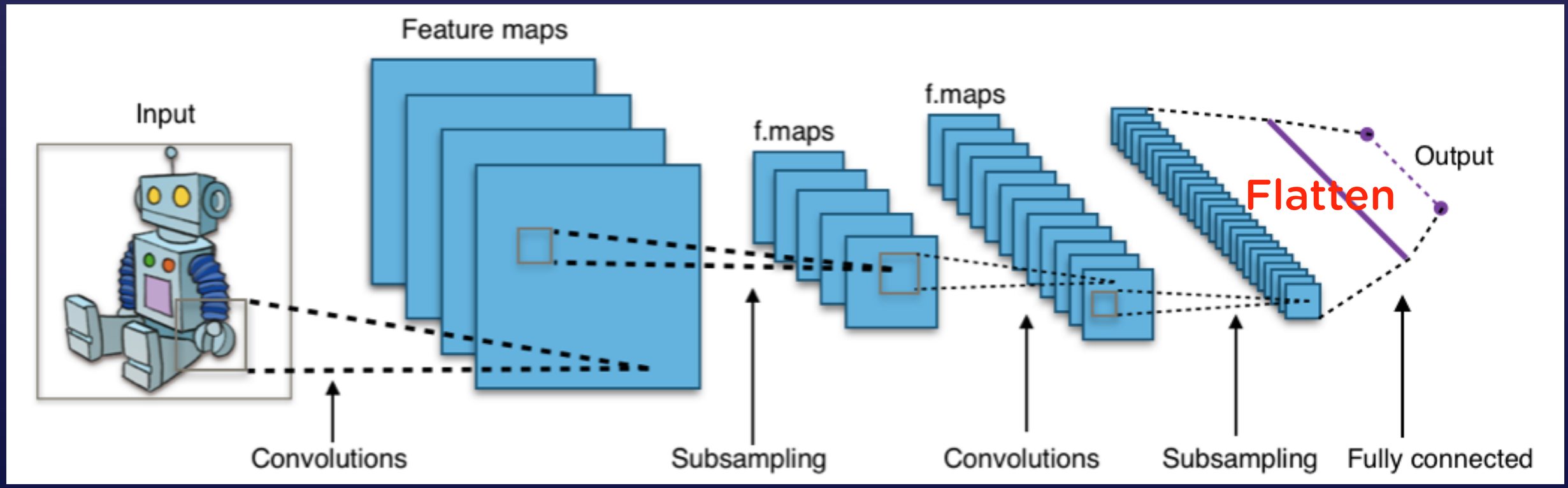
- Refer to x_t and h_(t-1)
- memory + activation: ready to get out
- Decide how much memory info could get out !



Reference : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Convolutional Neural Network

Images : local properties!



Convolution Pooling Repeat 2 steps

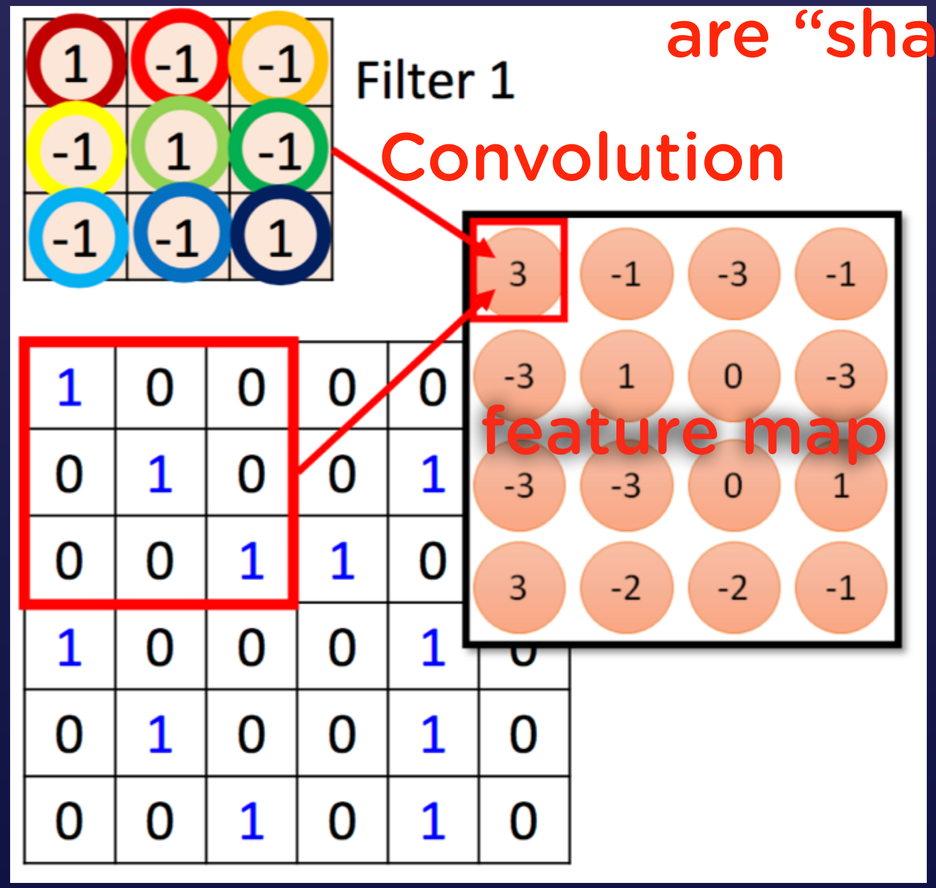
Reference : Wikipedia

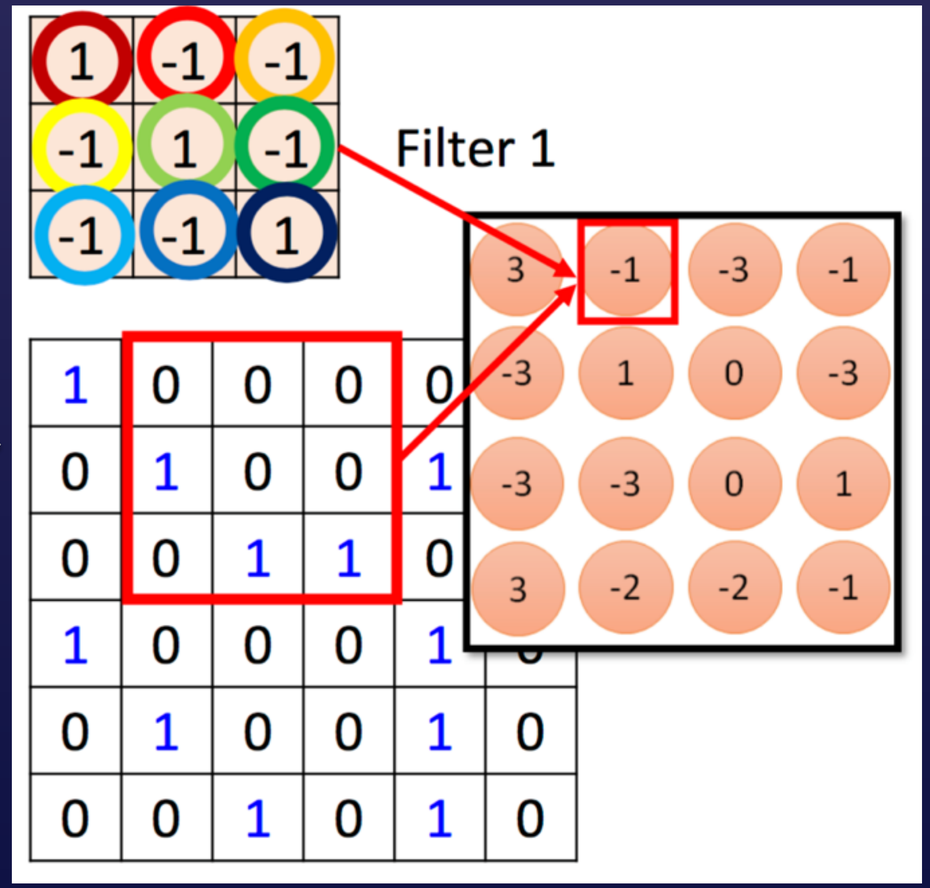
Convolutional Neural Network

Convolution

Parameters are “shared”

Multiple Filters !



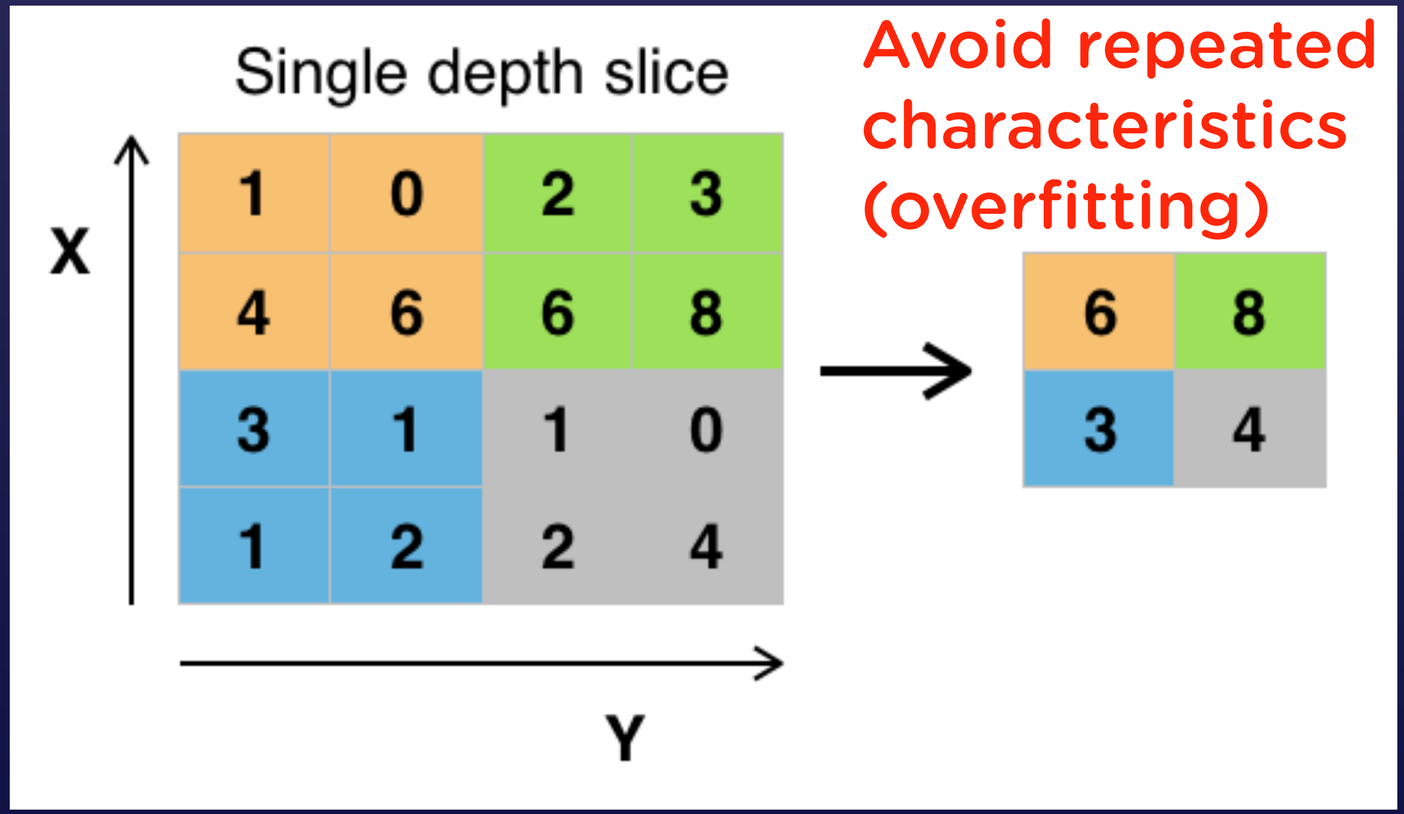


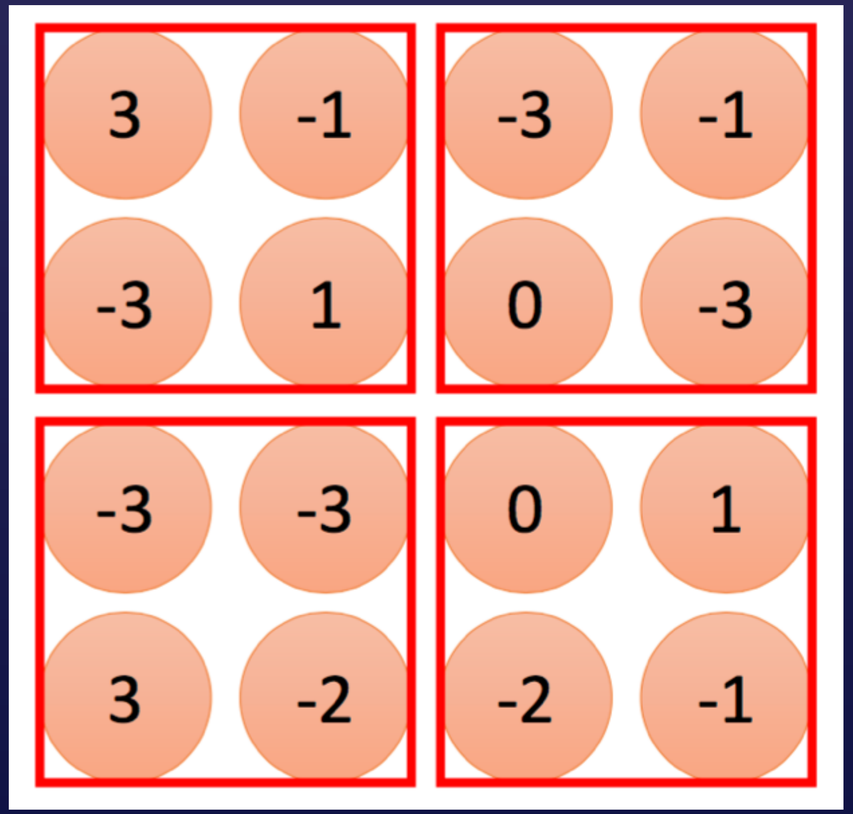
Reference : http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML16.html

GDG DevFest TAIPEI 2016
GLIACLOUD

Convolutional Neural Network

Max Pooling

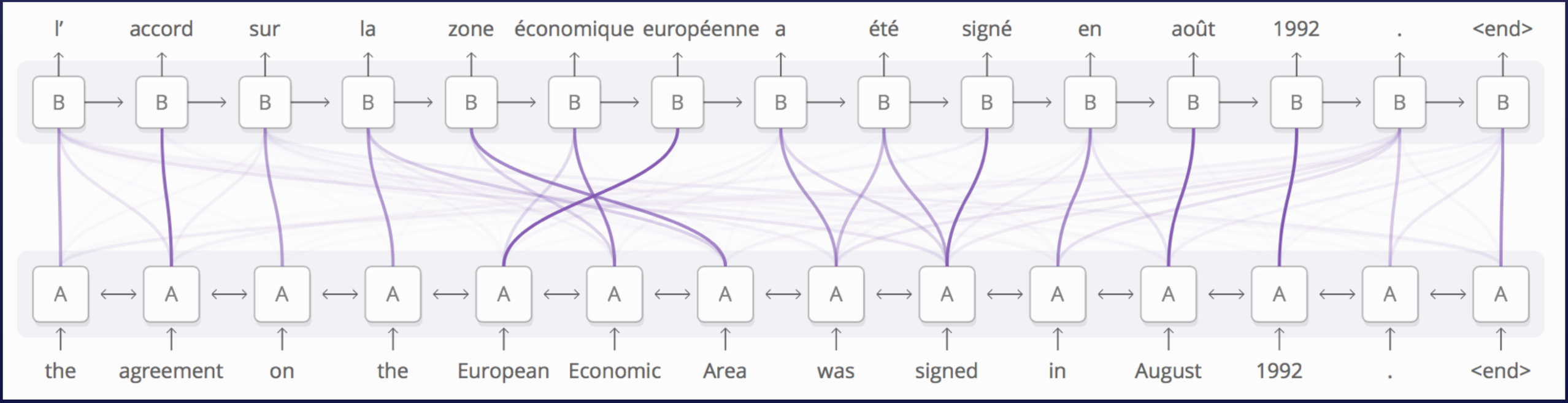




Reference : http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML16.html

Attention Mechanism

Attention : See the bigger picture
and focus on somewhere “important”

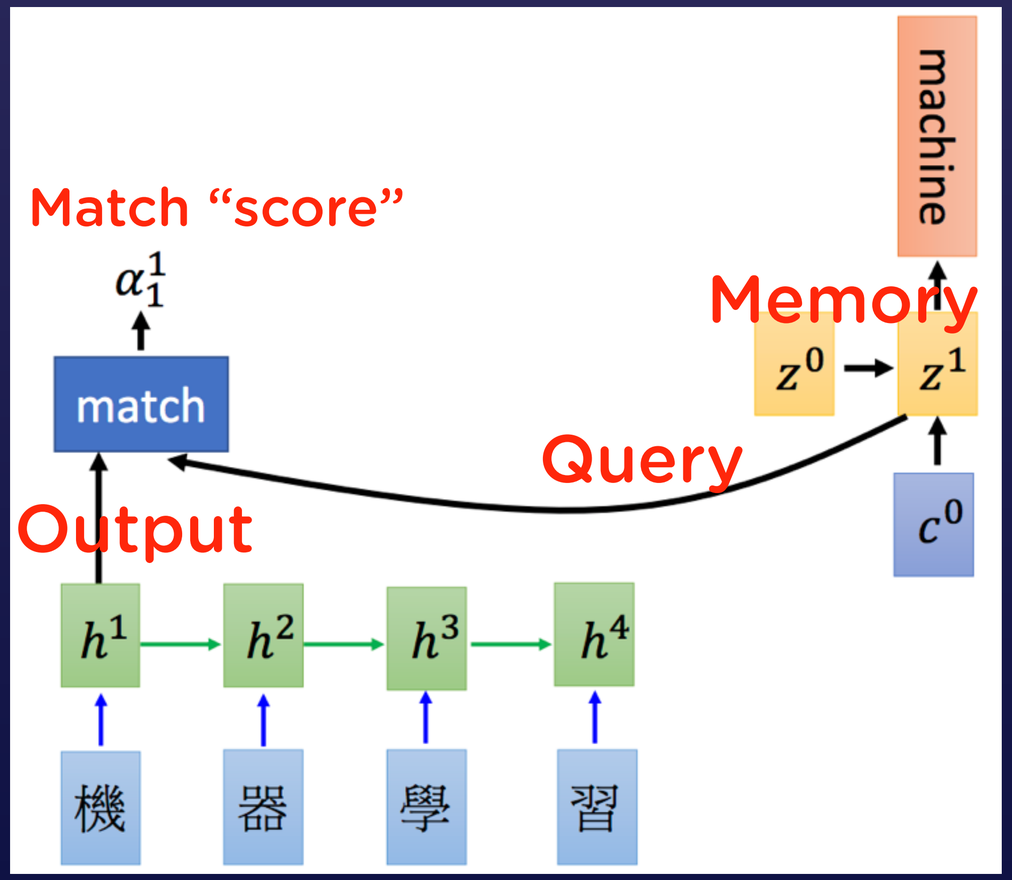


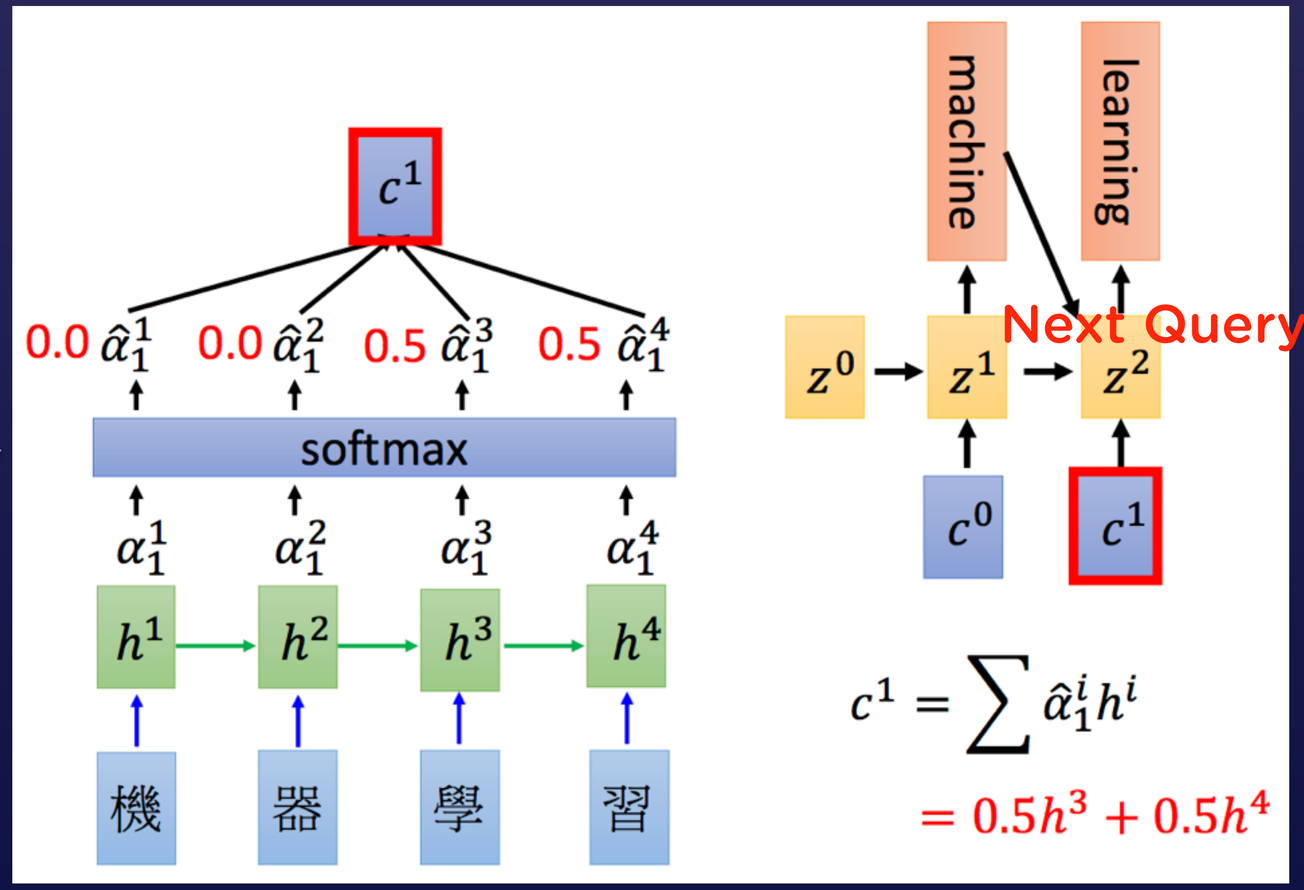
Reference : http://distill.pub/2016/augmented-rnns/

GDG DevFest TAIPEI 2016
GLIACLOUD

Attention Mechanism

Attention How? : “Query” and “Match”





Reference : http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2

Attention Mechanism

Match?

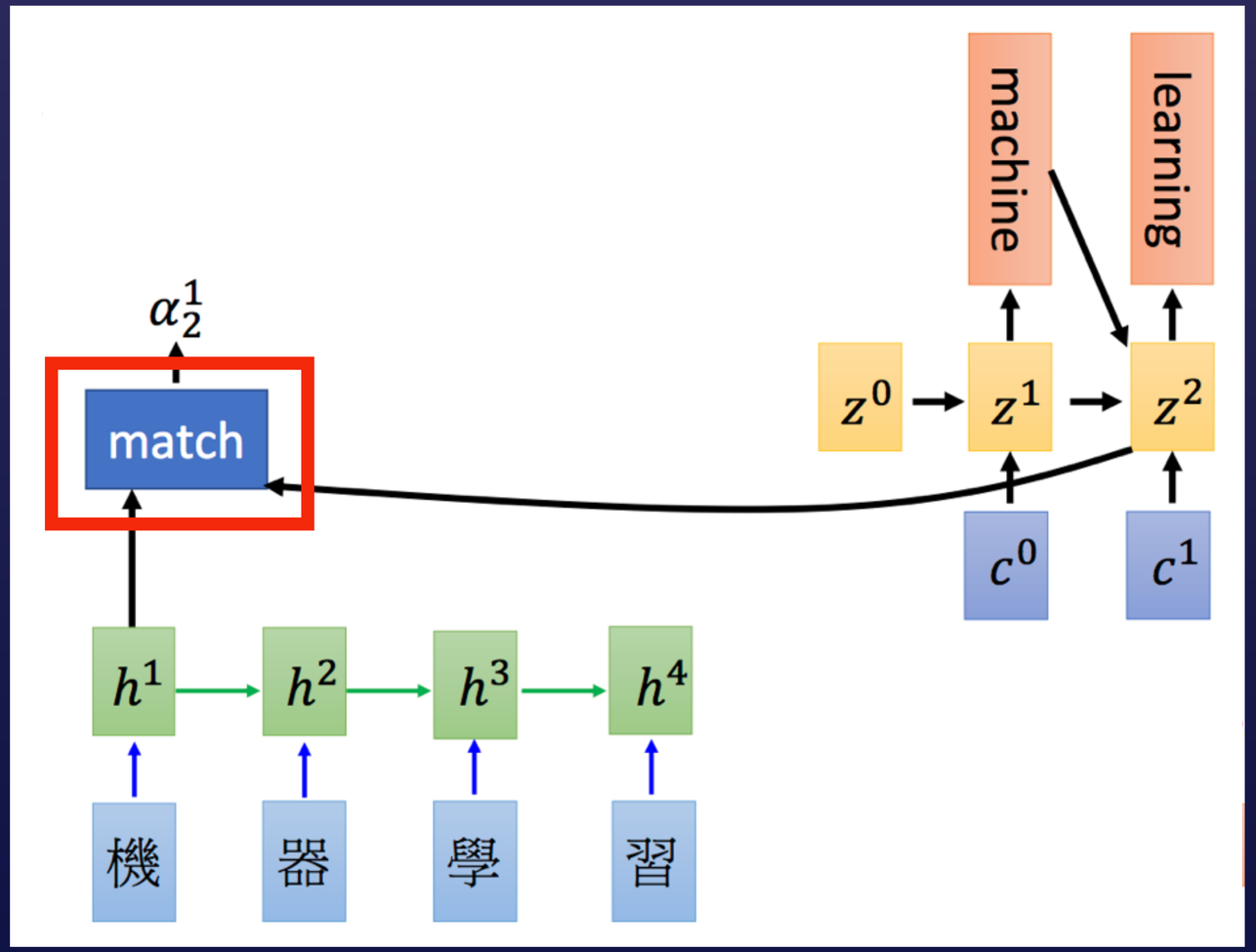
- Matchness of **“h”** and **memory**
- Some kind of “similarity”

Dot product

Cosine similarity

Score : h’Wz

Small neural network



Reference : http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2

Attention Mechanism

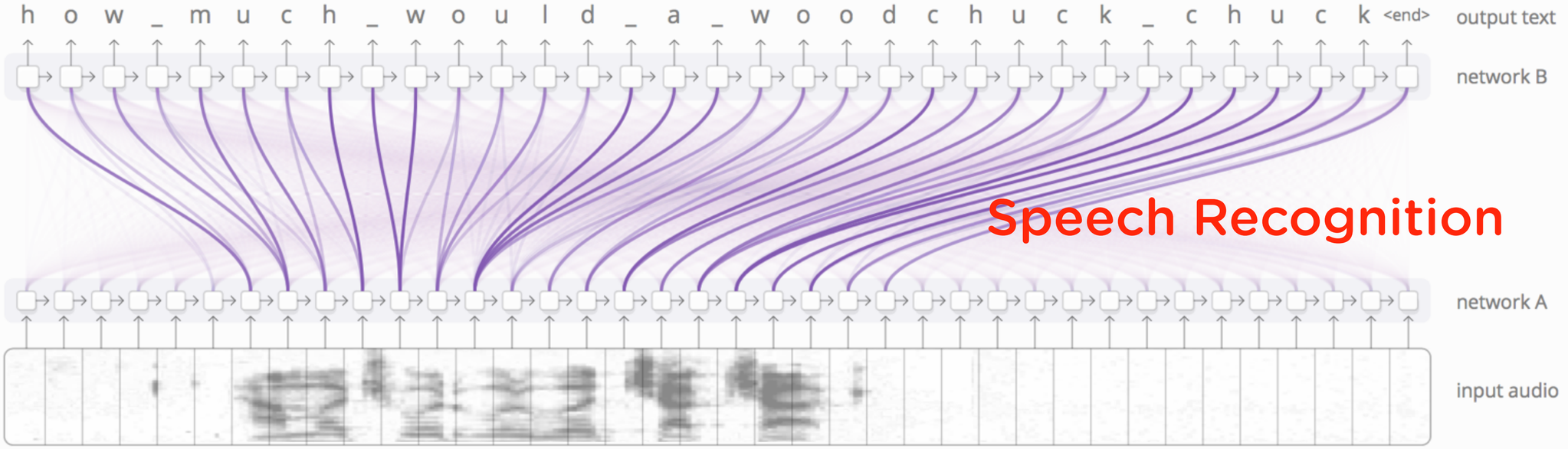
Image Captioning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

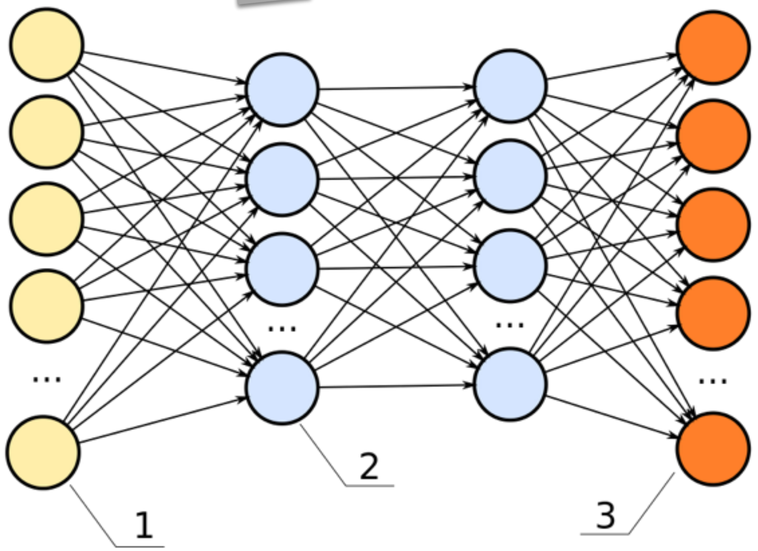
Speech Recognition



Reference : http://distill.pub/2016/augmented-rnns/

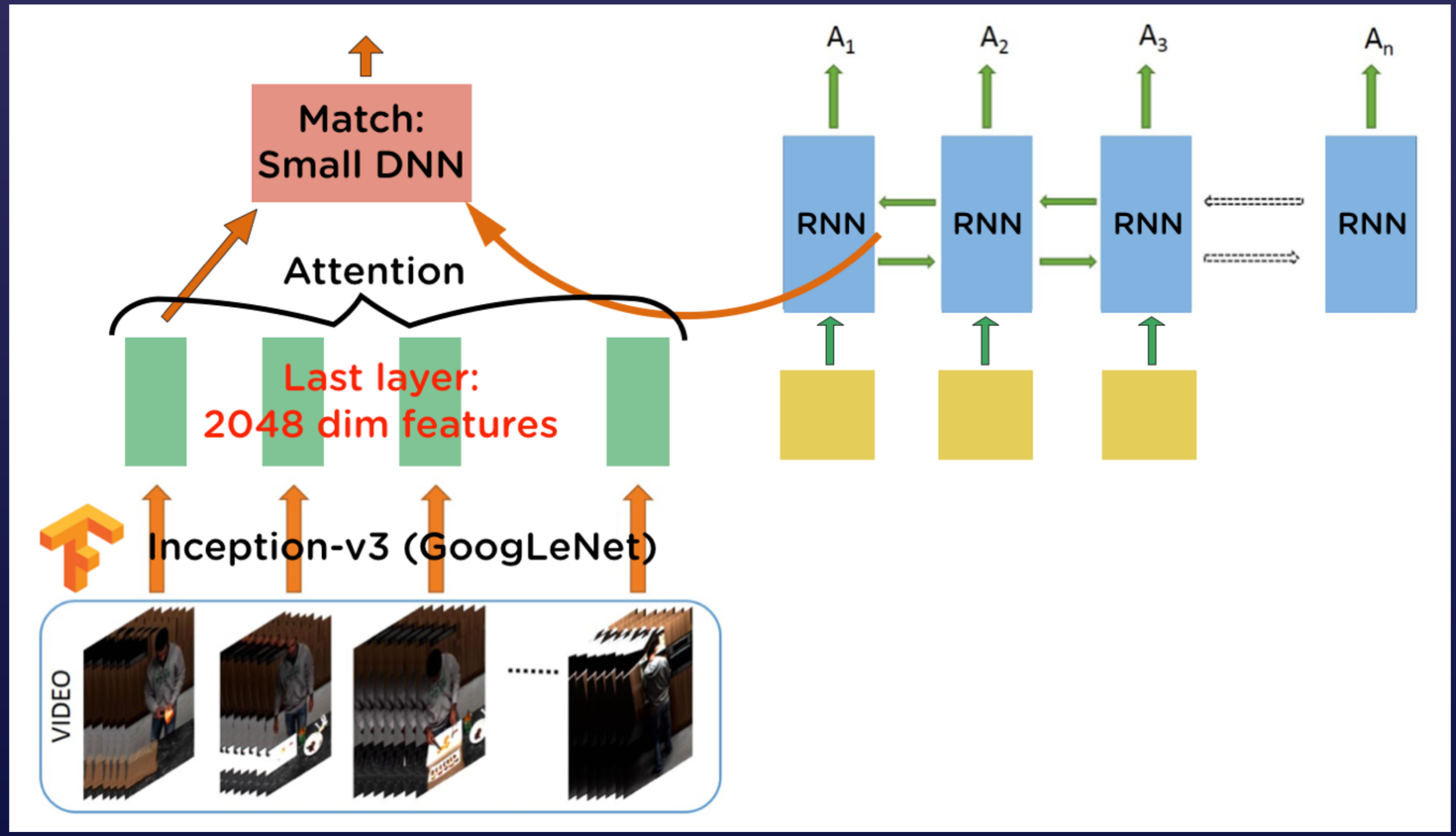
Outline

ACTIVITYNET



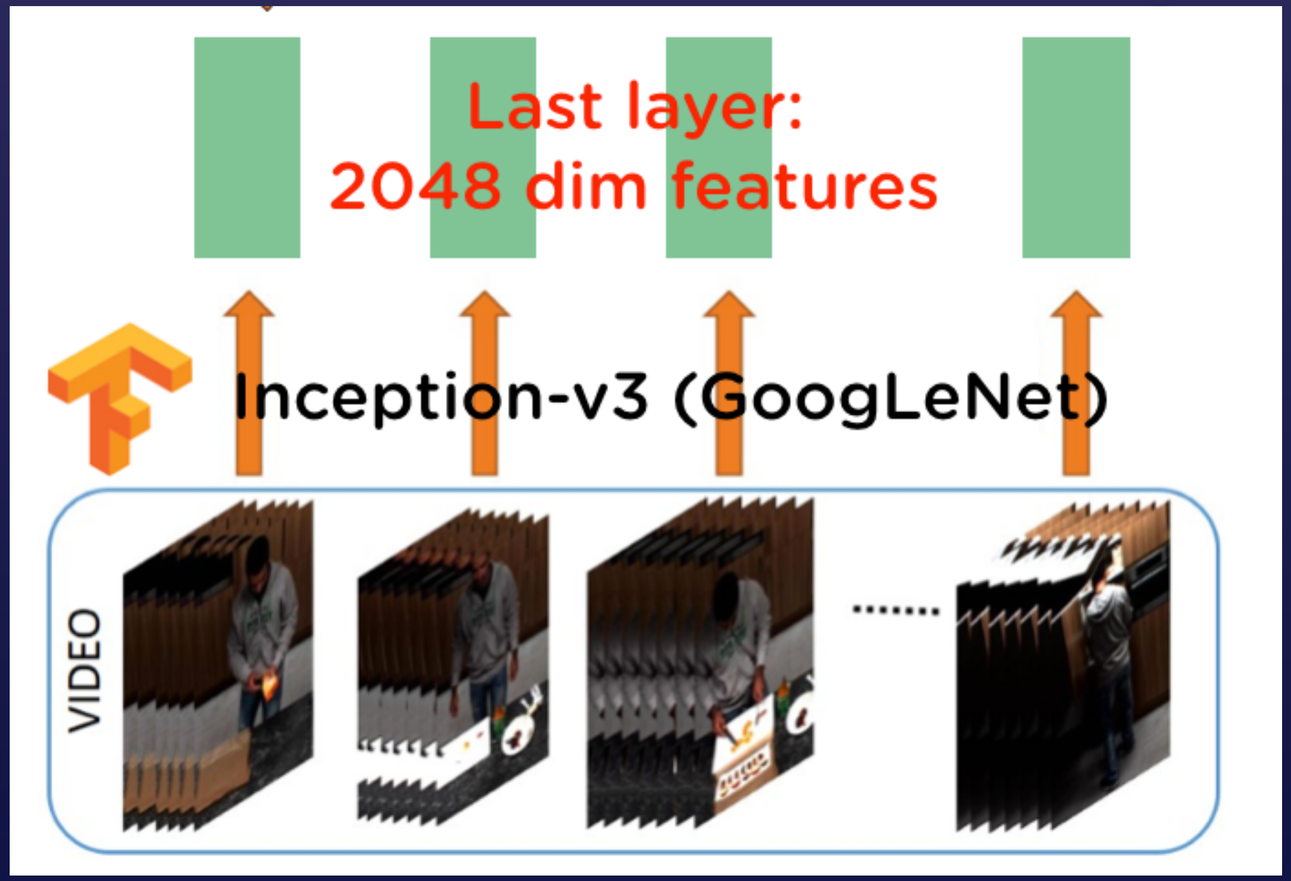
TensorFlow

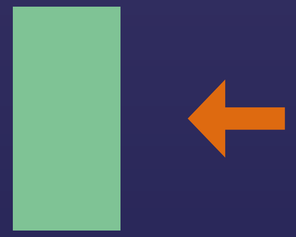
Putting All Things Together



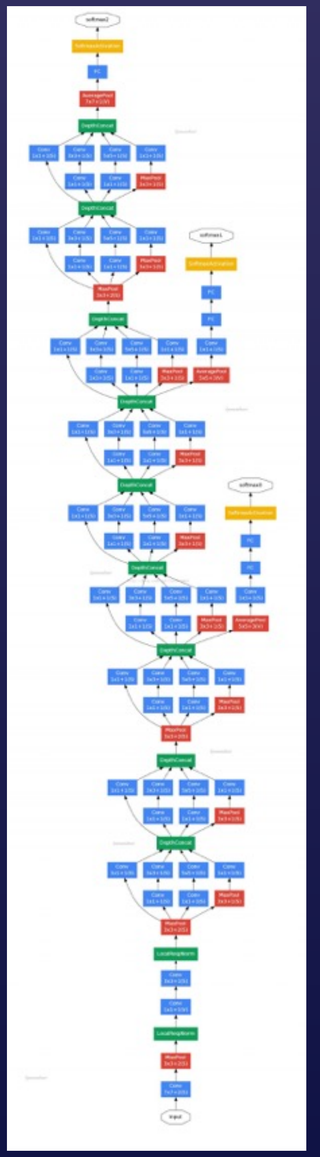
Reference : http://www.cs.umd.edu/~bharat/cvpr2016.pdf

Frame Features





Convolution
Neural Network



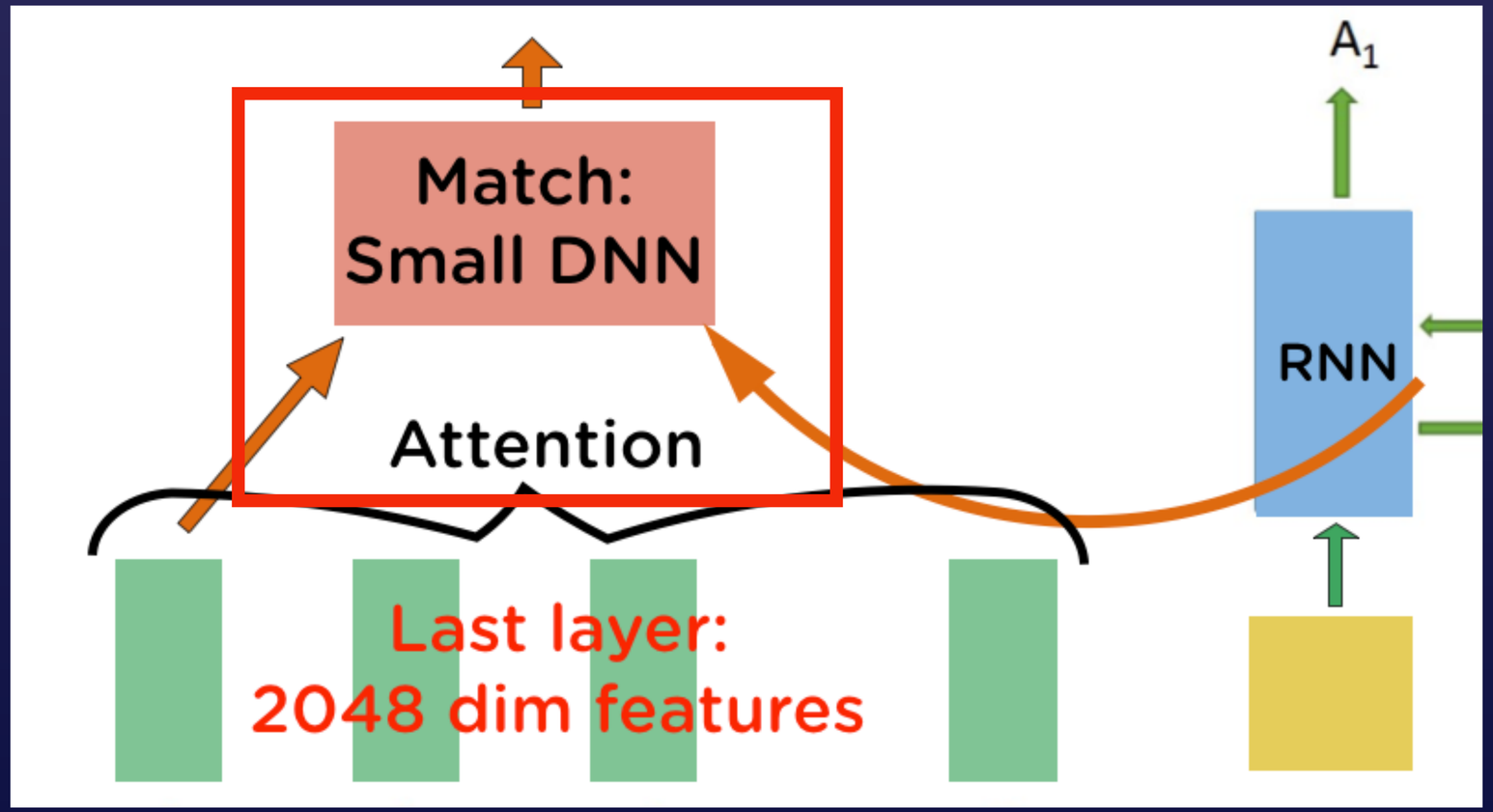
Pre-trained
Inception-v3

Reference : https://www.tensorflow.org/versions/r0.12/tutorials/image_recognition/index.html

Attention and DNN

Attention Mechanism

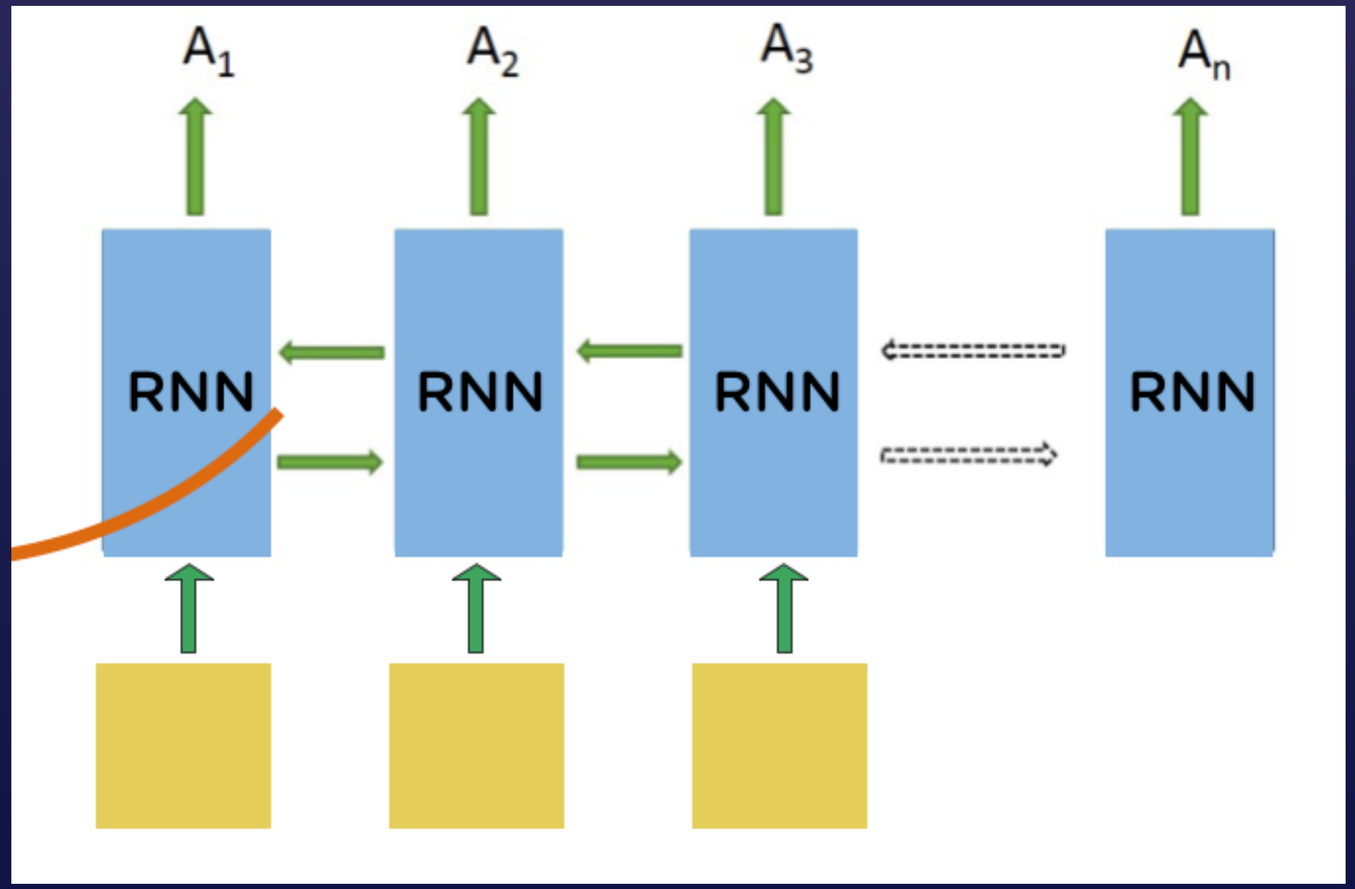
Deep Neural Network



Reference : http://www.cs.umd.edu/~bharat/cvpr2016.pdf

Sequence Labeling

Recurrent Neural Network



Reference : http://www.cs.umd.edu/~bharat/cvpr2016.pdf

Sample Code

Initialize all variables

Attention DNN

RNN Cell

```
def RNN_attention(x, n_input, n_hidden, n_attn_hidden, n_classes, scope=None):
    with tf.variable_scope(scope or "RNN"):
        W_in = tf.get_variable("Win_atten", [n_input + n_hidden, n_attn_hidden])
        b_in = tf.get_variable("bin_atten", [n_attn_hidden])
        W2 = tf.get_variable("W2_atten", [n_attn_hidden, n_attn_hidden])
        b2 = tf.get_variable("b2_atten", [n_attn_hidden])
        W_out = tf.get_variable("Wout_atten", [n_attn_hidden, 1])
        b_out = tf.get_variable("bout_atten", [1])
        d_0 = tf.get_variable("d_0", [1, n_input + n_hidden])

        W_h = tf.get_variable("W_mem", [n_hidden, n_hidden])
        W_i = tf.get_variable("W_in", [n_input, n_hidden])
        b_i = tf.get_variable("b_in", [n_hidden])
        W_o = tf.get_variable("W_out", [n_hidden, n_classes])
        b_o = tf.get_variable("b_out", [n_classes])
        state = tf.get_variable("state0", [1, n_hidden])
```

Output Scalar

Reference : https://github.com/AaronYALai/tensorflow_activity_recognition

Sample Code

Attention

```python
# attention mechaism
def attention(query, x, scope):
    with tf.variable_scope(scope):
        tf.get_variable_scope().reuse_variables()

        d_0 = tf.get_variable("d_0")
        dnn_input = tf.scan(lambda _, b: tf.concat(1, [b, query]),
                            tf.transpose(x, [1, 0, 2]), initializer=d_0)
        dnn_input = tf.reshape(dnn_input, [-1, n_input + n_hidden])

        W_in = tf.get_variable("Win_atten")
        b_in = tf.get_variable("bin_atten")
        W2 = tf.get_variable("W2_atten")
        b2 = tf.get_variable("b2_atten")
        W_out = tf.get_variable("Wout_atten")
        b_out = tf.get_variable("bout_atten")

        # attention "match" DNN
        a_attn = tf.nn.relu(tf.add(tf.matmul(dnn_input, W_in), b_in))
        a_attn2 = tf.nn.relu(tf.add(tf.matmul(a_attn, W2), b2))
        attn_score = tf.add(tf.matmul(a_attn2, W_out), b_out)
        attn = tf.nn.softmax(tf.reshape(attn_score, [-1]))

        return attn
```

DNN

Reference : https://github.com/AaronYALai/tensorflow_activity_recognition

Sample Code

RNN Cell

```
    # RNN cell
    def cell(memory, _):
        tf.get_variable_scope().reuse_variables()
        W_h = tf.get_variable("W_mem")
        W_i = tf.get_variable("W_in")
        b_i = tf.get_variable("b_in")

        mem = tf.matmul(memory, W_h)
        attn = attention(memory, x, scope)

        x_t = tf.transpose(x[0], [1, 0])
        atten_x = tf.reshape(tf.reduce_sum(tf.mul(x_t, attn), 1), [1, n_input])

        memory = tf.nn.tanh(tf.add(tf.add(tf.matmul(atten_x, W_i), b_i), mem))
        return memory

    out_seq = tf.scan(cell, tf.transpose(x, [1, 2, 0]), initializer=state)
    out_seq = tf.reshape(out_seq, [-1, n_hidden])
    outputs = tf.nn.softmax(tf.add(tf.matmul(out_seq, W_o), b_o))

return outputs
```

One Step of RNN

Reference : https://github.com/AaronYALai/tensorflow_activity_recognition

Conclusions

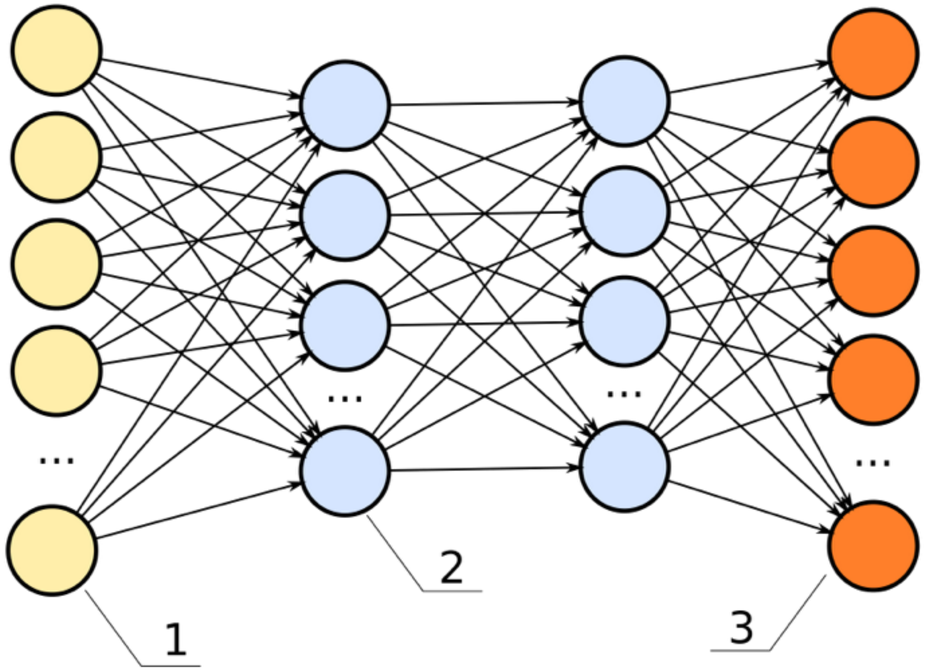
Still Young

Many Tricks

Be Powerful on Cloud Soon

More Examples

https://github.com/aymericdamien/TensorFlow-Examples



Resources

- Machine Learning and having it deep and structured(HungYi Lee): http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD15_2.html
- TensorFlow Deep Learning Examples: https://github.com/aymericdamien/TensorFlow-Examples
- Long Short-Term Memery: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- “Deep Learning” by Ian Goodfellow, Yoshua Bengio and Aaron Courville: http://www.deeplearningbook.org/
- Gradient Descent Optimization Algorithms: http://sebastianruder.com/optimizing-gradient-descent/index.html

GDG DevFest
TAIPEI 2016

Thanks for Listening !

GLIACLOUD