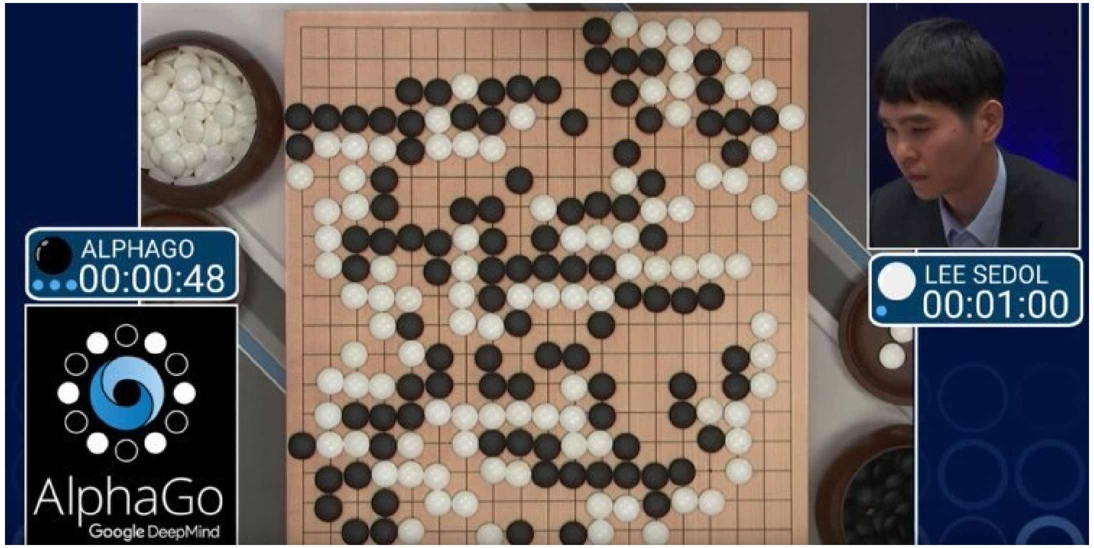
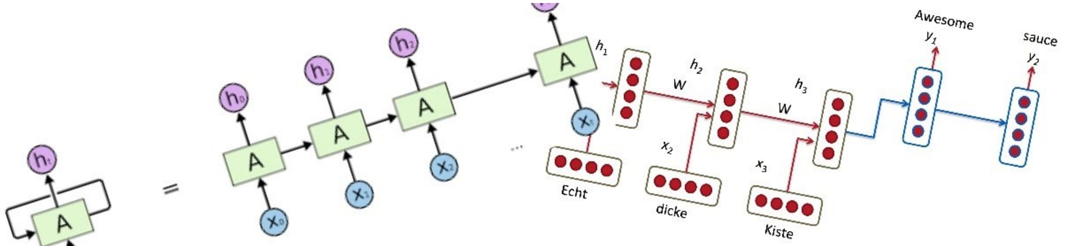
# Deeplearning

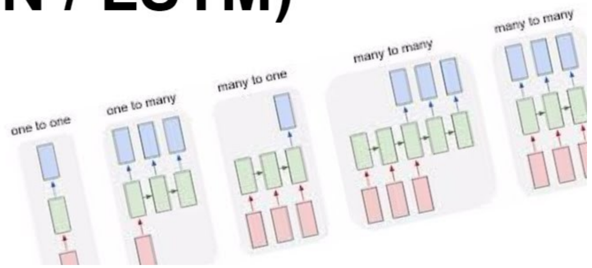
Thực sự thì "trí thông minh nhân tạo"

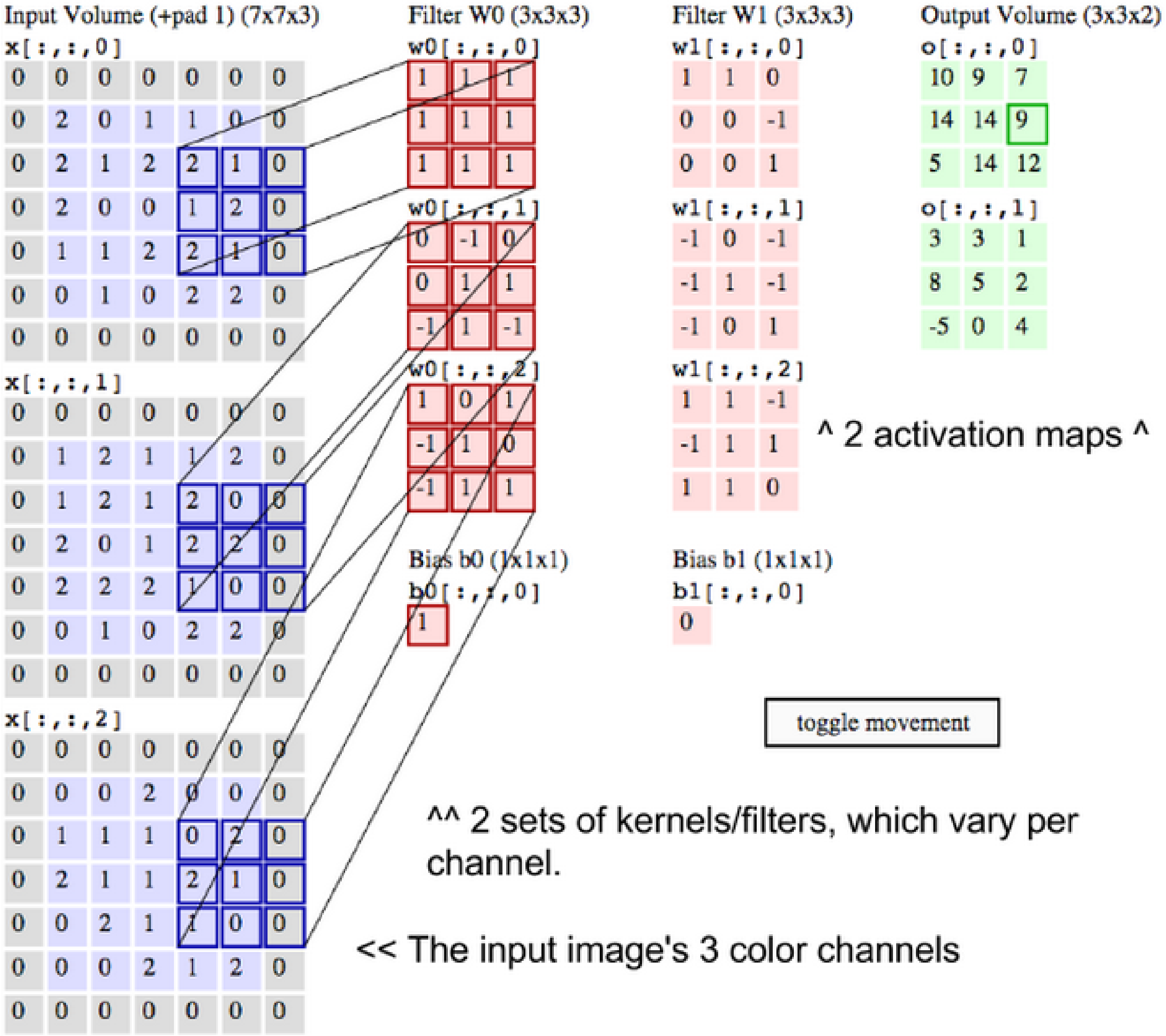
ở thời điểm hiện tại là như thế nào?





# Recurrent Neural Networks (RNN / LSTM)





Input Volume (+pad 1) (7x7x3)

x[:,:,0]

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 0 | 1 | 1 | 0 | 0 |
| 0 | 2 | 1 | 2 | 2 | 1 | 0 |
| 0 | 2 | 0 | 0 | 1 | 2 | 0 |
| 0 | 1 | 1 | 2 | 2 | 1 | 0 |
| 0 | 0 | 1 | 0 | 2 | 2 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

x[:,:,1]

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 2 | 1 | 1 | 2 | 0 |
| 0 | 1 | 2 | 1 | 2 | 0 | 0 |
| 0 | 2 | 0 | 1 | 2 | 2 | 0 |
| 0 | 2 | 2 | 2 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 2 | 2 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

x[:,:,2]

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 2 | 0 |
| 0 | 2 | 1 | 1 | 2 | 1 | 0 |
| 0 | 0 | 2 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 2 | 1 | 2 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Filter W0 (3x3x3)

w0[:,:,0]

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

w0[:,:,1]

| | | |
|---|---|---|
| 0 | -1 | 0 |
| 0 | 1 | 1 |
| -1 | 1 | -1 |

w0[:,:,2]

| | | |
|---|---|---|
| 1 | 0 | 1 |
| -1 | 1 | 0 |
| -1 | 1 | 1 |

Bias b0 (1x1x1)

b0[:,:,0]

| |
|---|
| 1 |

Filter W1 (3x3x3)

w1[:,:,0]

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 0 | 0 | -1 |
| 0 | 0 | 1 |

w1[:,:,1]

| | | |
|---|---|---|
| -1 | 0 | -1 |
| -1 | 1 | -1 |
| -1 | 0 | 1 |

w1[:,:,2]

| | | |
|---|---|---|
| 1 | 1 | -1 |
| -1 | 1 | 1 |
| 1 | 1 | 0 |

Bias b1 (1x1x1)

b1[:,:,0]

| |
|---|
| 0 |

Output Volume (3x3x2)

o[:,:,0]

| | | |
|---|---|---|
| 10 | 9 | 7 |
| 14 | 14 | 9 |
| 5 | 14 | 12 |

o[:,:,1]

| | | |
|---|---|---|
| 3 | 3 | 1 |
| 8 | 5 | 2 |
| -5 | 0 | 4 |

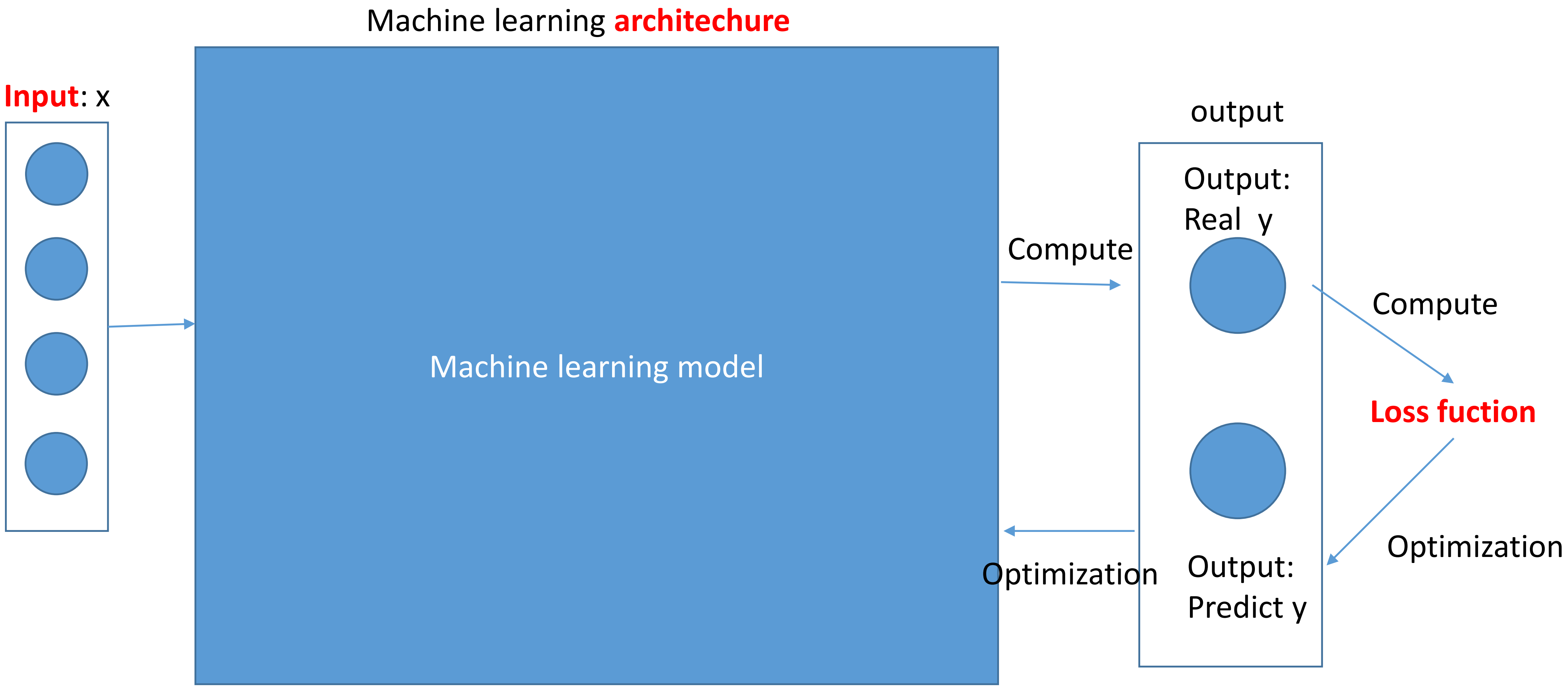
^ 2 activation maps ^

toggle movement

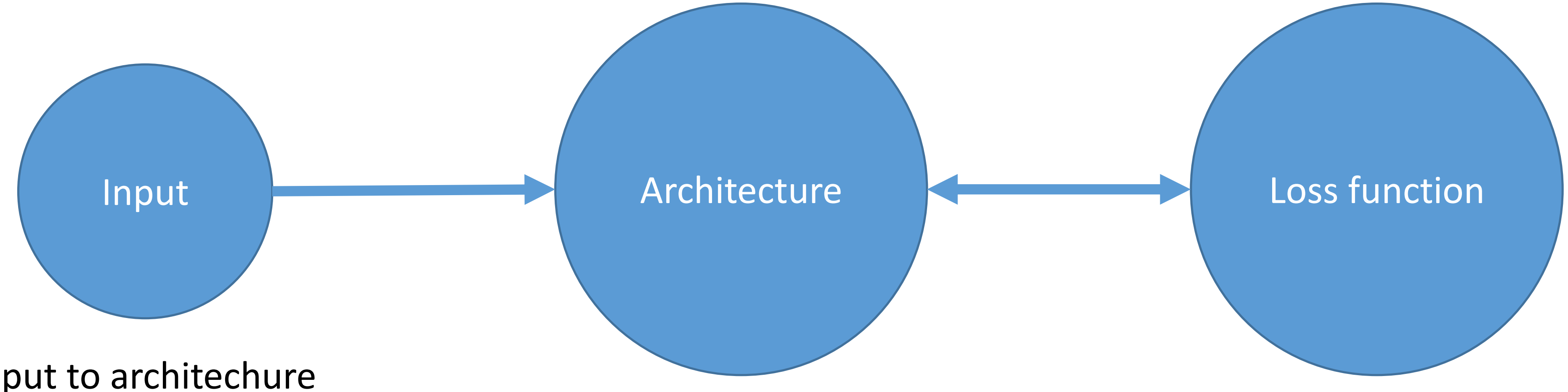
^^ 2 sets of kernels/filters, which vary per channel.

<< The input image's 3 color channels

# Bài toán phân loại



# Bài toán phân loại | Role of each part

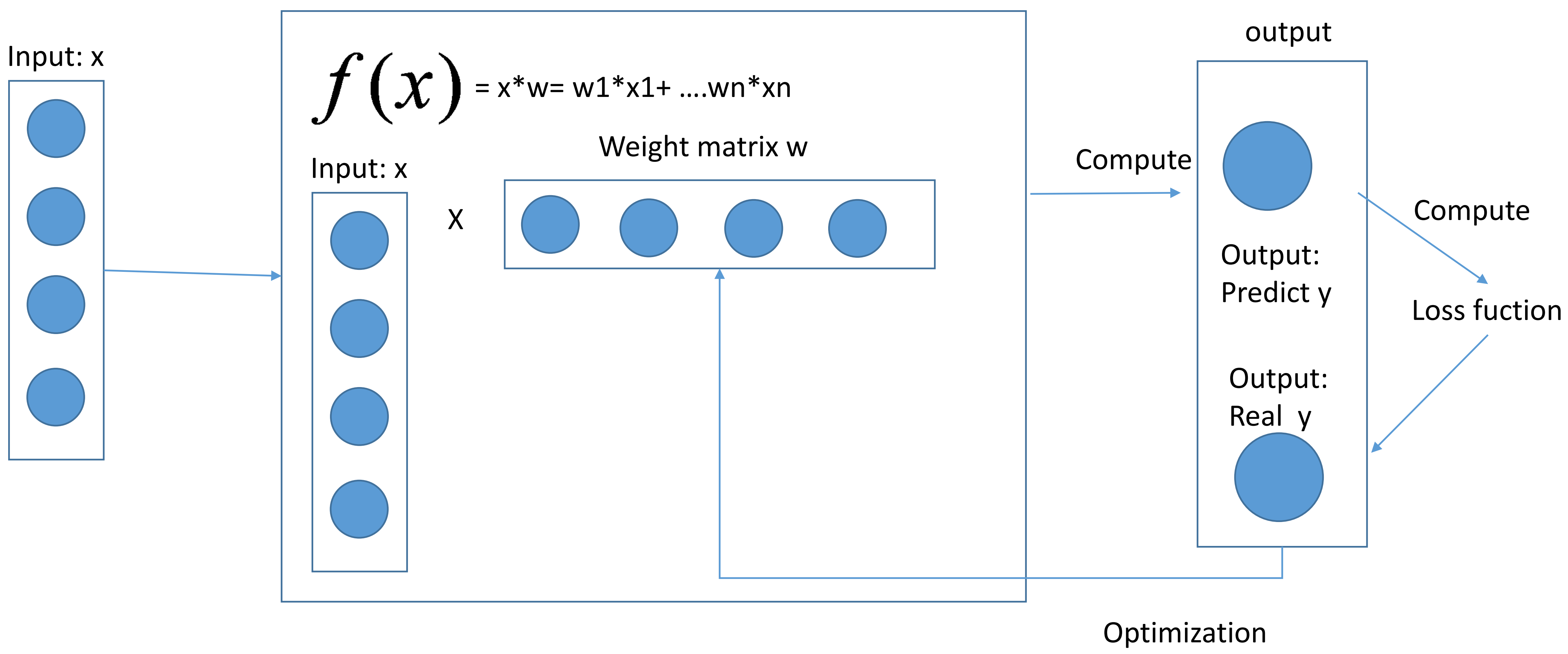


Input to architechure

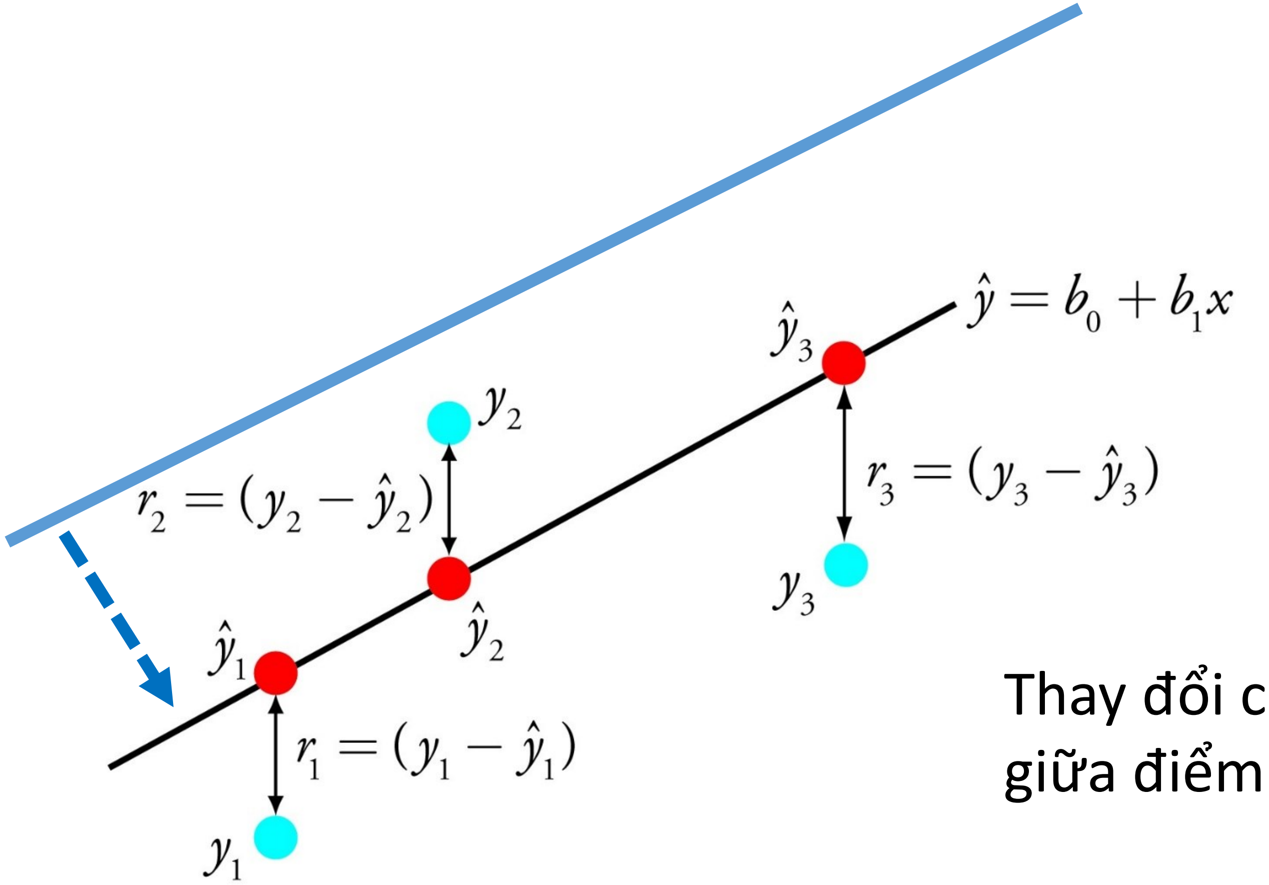
- Initialize weight matrix
- Compute output with input given
- Optimize Weights base on “signal” of loss function

- Compute loss between output and target.
- Tell Architecture how good did it work
- Tell Architechture how to optimize weights

# Bài toán phân loại | Linear regression



# Bài toán phân loại | Linear regression

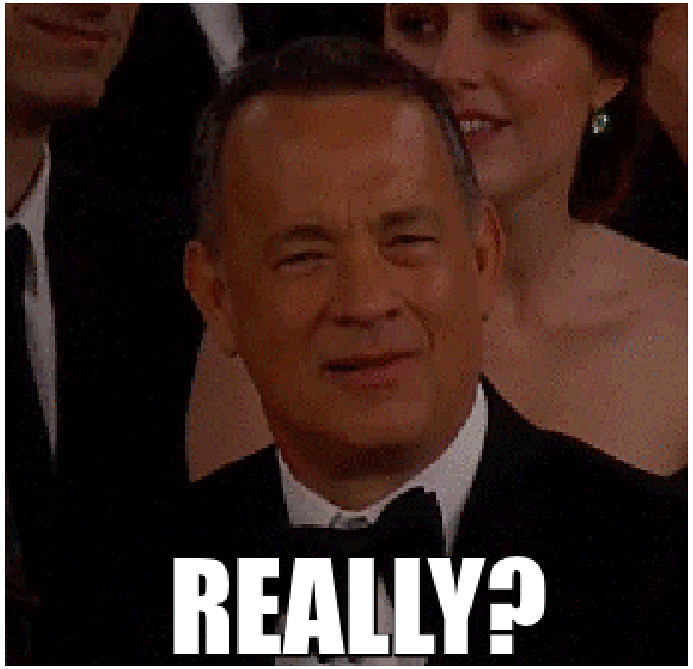

$\hat{y} = b_0 + b_1 x$
$\hat{y}_3$
$y_2$
$r_2 = (y_2 - \hat{y}_2)$
$r_3 = (y_3 - \hat{y}_3)$
$y_3$
$\hat{y}_2$
$\hat{y}_1$
$r_1 = (y_1 - \hat{y}_1)$
$y_1$

Loss fuction

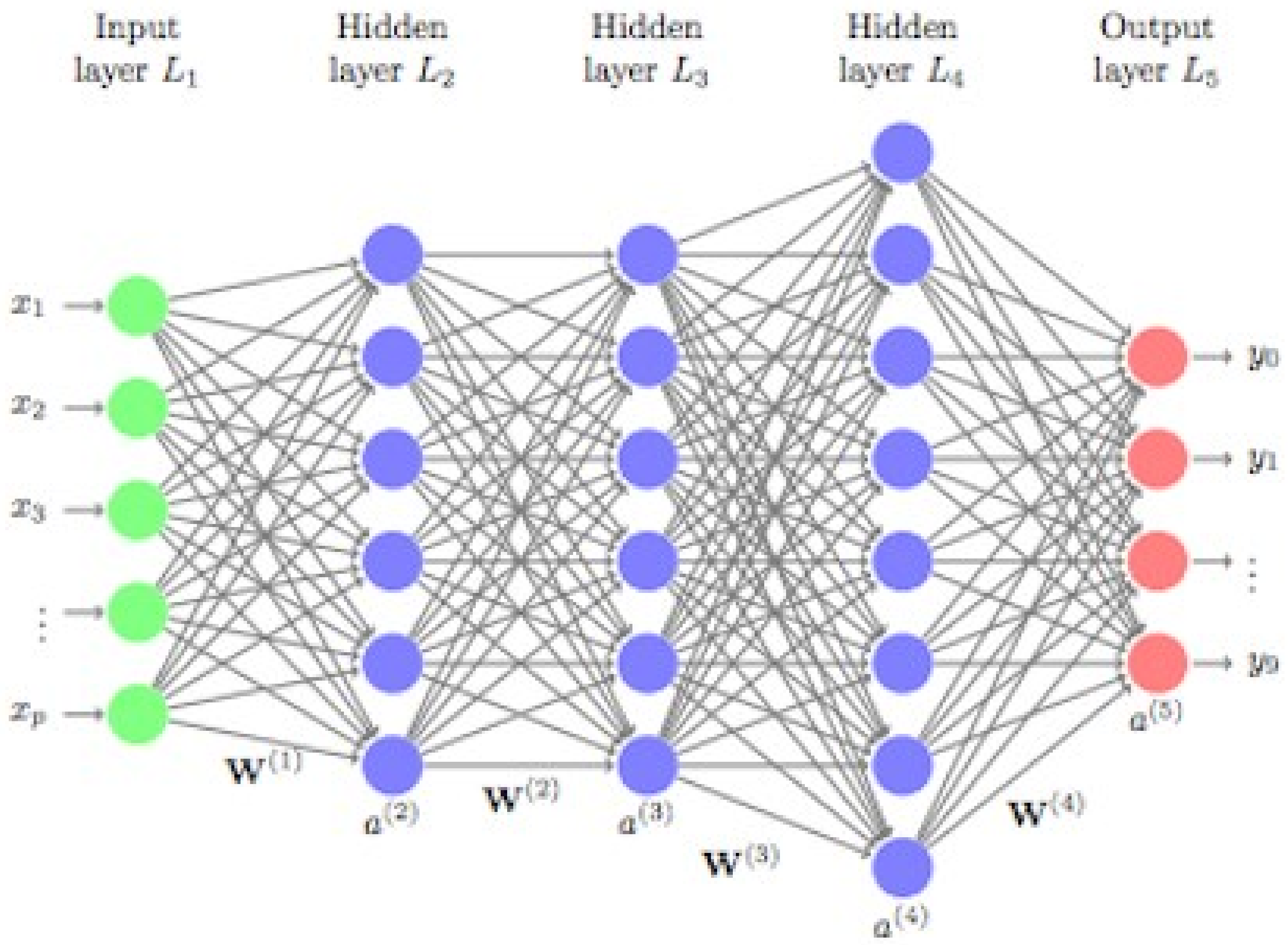
$$Loss(y, \hat{y}) = (y - \hat{y})^2$$

Thay đổi các tham số w sao cho tổng khoảng cách giữa điểm dự đoán và điểm thực tế nhỏ nhất có thể

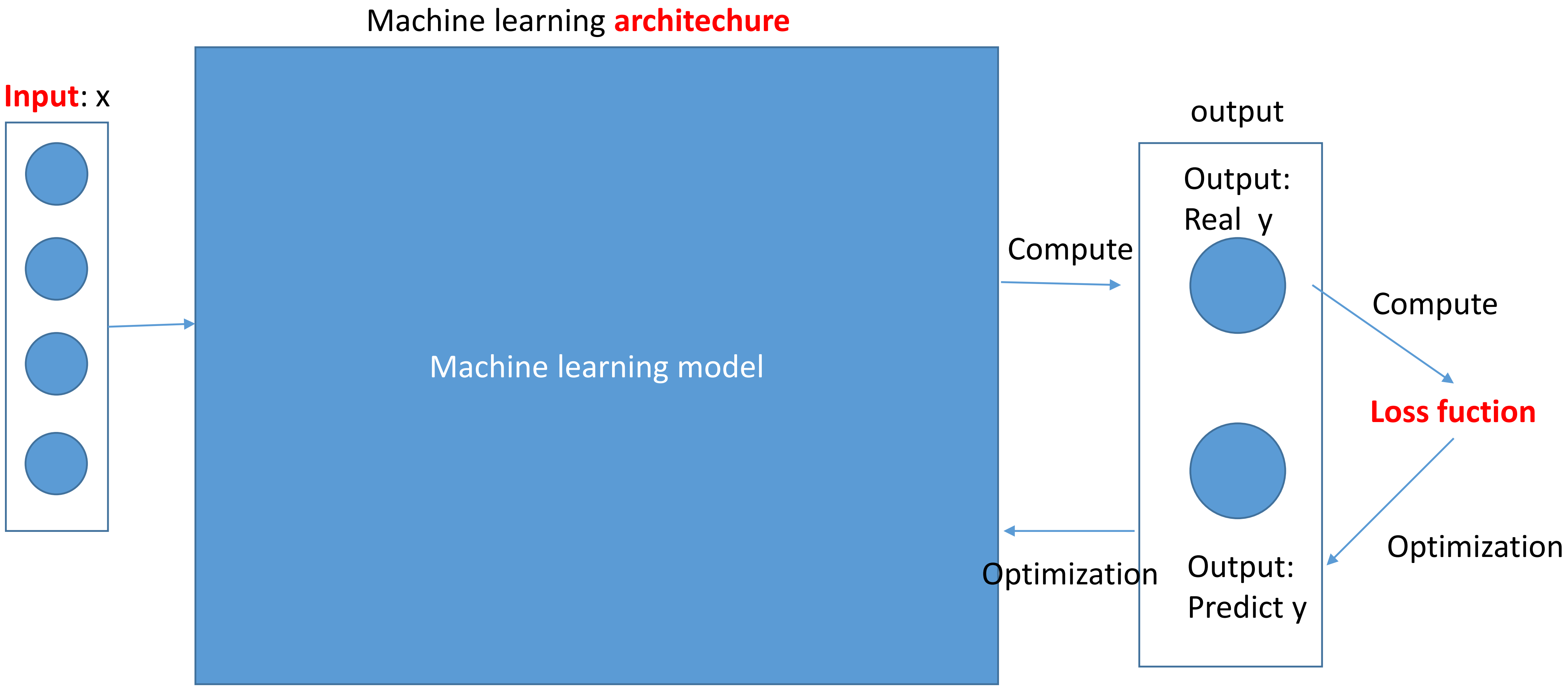
So, we already know all what we need!



# But, what about thing that people call “deep learning”?



# It nearly the same with we saw before



# Lets deep dive in to architechture

## In linear regression, w - weight is a vector

$f(x)$ = x*w = y
= [1,n]*[n,1] = [1,1] -scalar

## If w is a matrix – what happen?

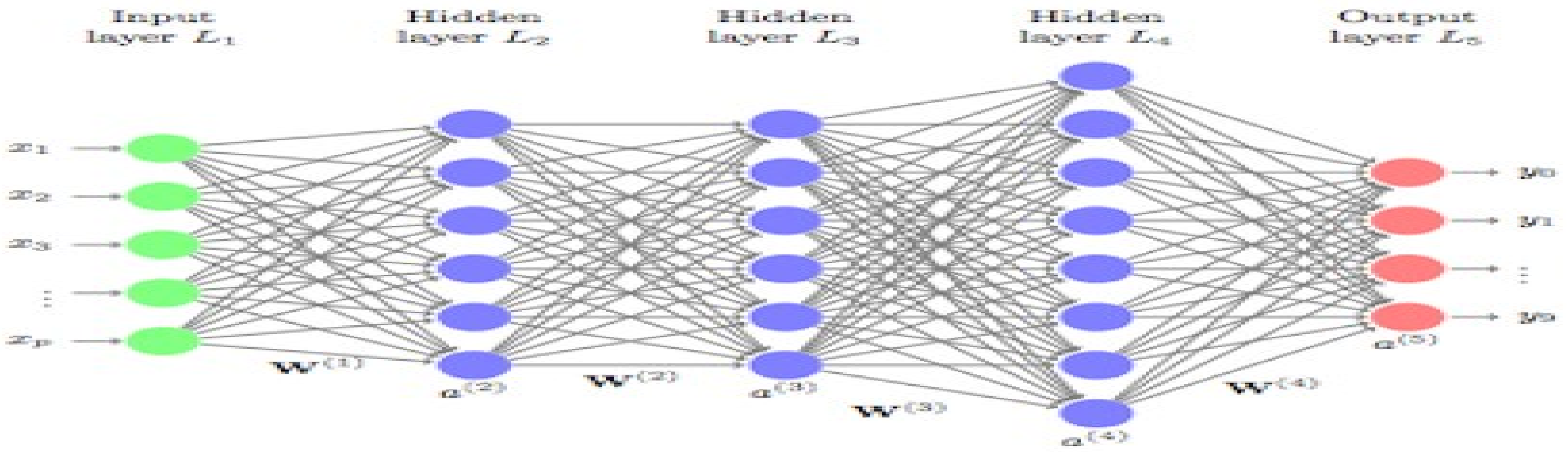
$f(x)$ = x*w = y
= [1,n]*[n,m] = [1,m] - vector

## If w and even x is a matrix – what happen?

$f(x)$ = x*w = y
= [n,m]*[m,k] = [n,k] - matrix

# Lets deep dive in to architechture

## Just simple - “Deep learning” is a nested f(x)



| | | | | |
|---|---|---|---|---|
| X | f(x) | g(f(x)) | h(g(f(x))) | k(h(g(f(x)))) |
| 1x5 | 5x6 | 6x6 | 6x8 | 8x4 |

But what about

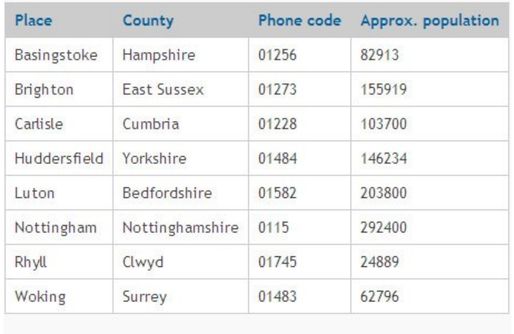
convolutional neural net
Recurrent neural net
Long shot term memory
Sequence-to-sequence
Deep pyramid convolution
Recurrent convolutional net
Very deep long shot term memory
....

It is how we combine f(x) together – like lego

# Let’s begin  with type of input x

X

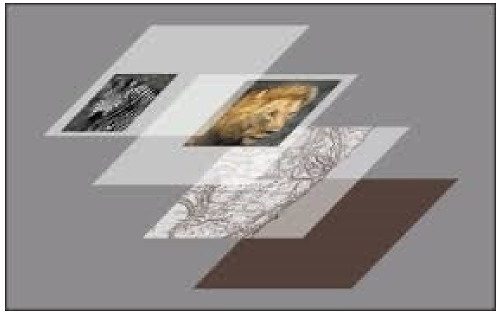
Vector
Fixed size, un-order → Tabular data



| Place | County | Phone code | Approx. population |
|---|---|---|---|
| Basingstoke | Hampshire | 01256 | 82913 |
| Brighton | East Sussex | 01273 | 155919 |
| Carlisle | Cumbria | 01228 | 103700 |
| Huddersfield | Yorkshire | 01484 | 146234 |
| Luton | Bedfordshire | 01582 | 203800 |
| Nottingham | Nottinghamshire | 0115 | 292400 |
| Rhyll | Clwyd | 01745 | 24889 |
| Woking | Surrey | 01483 | 62796 |

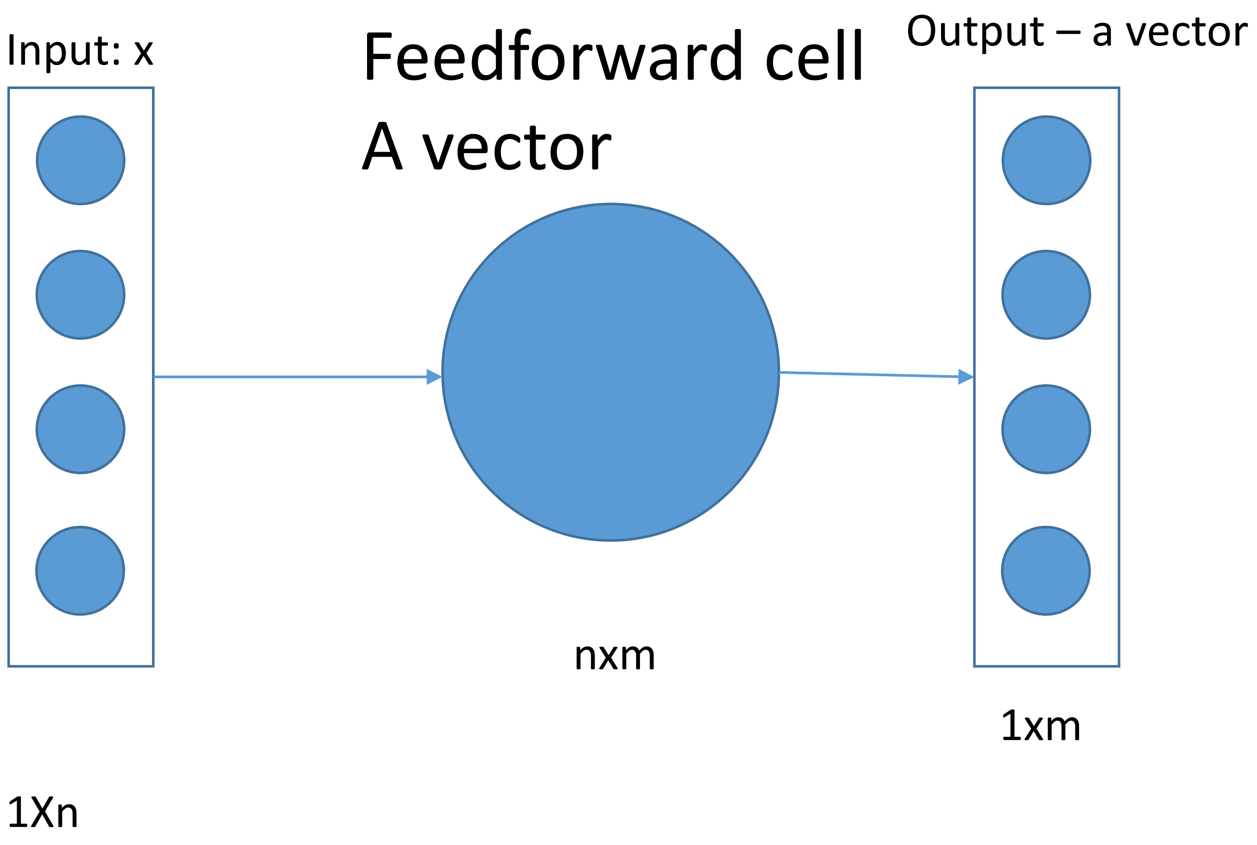
matrix

Fixed size, ordered → Image

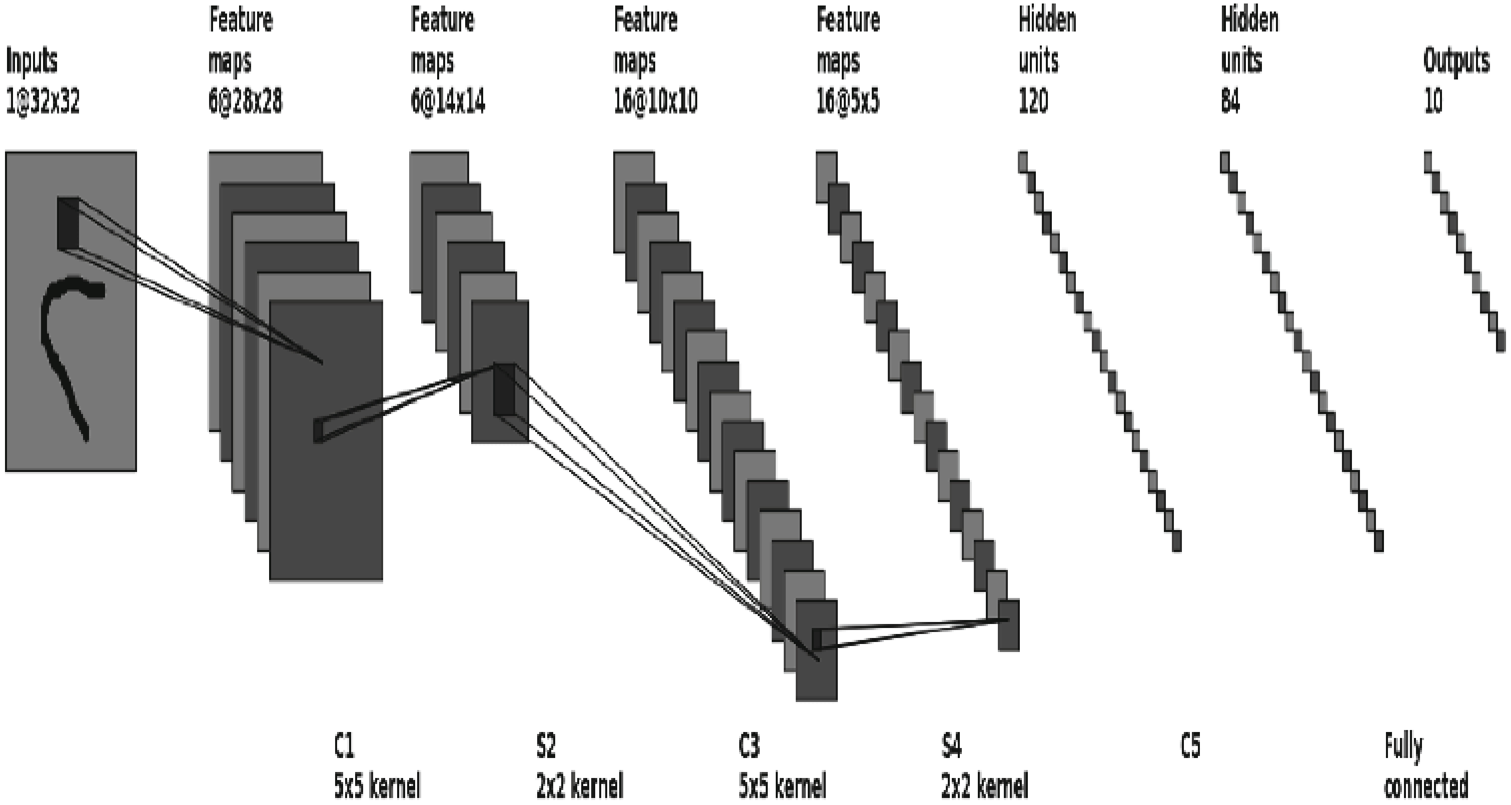


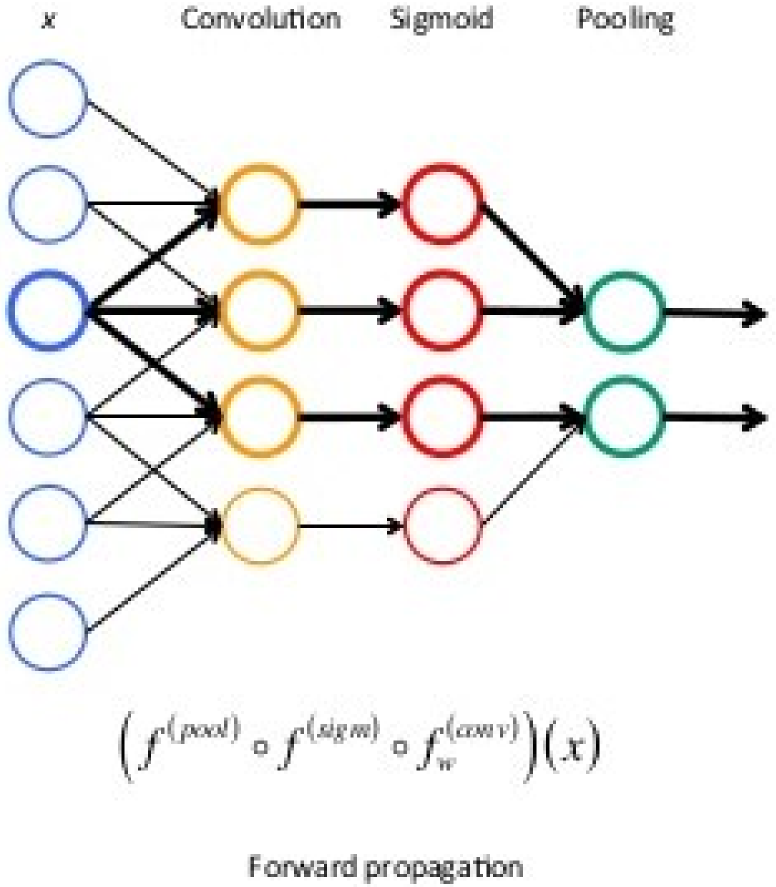
Not Fixed size, ordered → Text, voice, …

# Feed forward cell

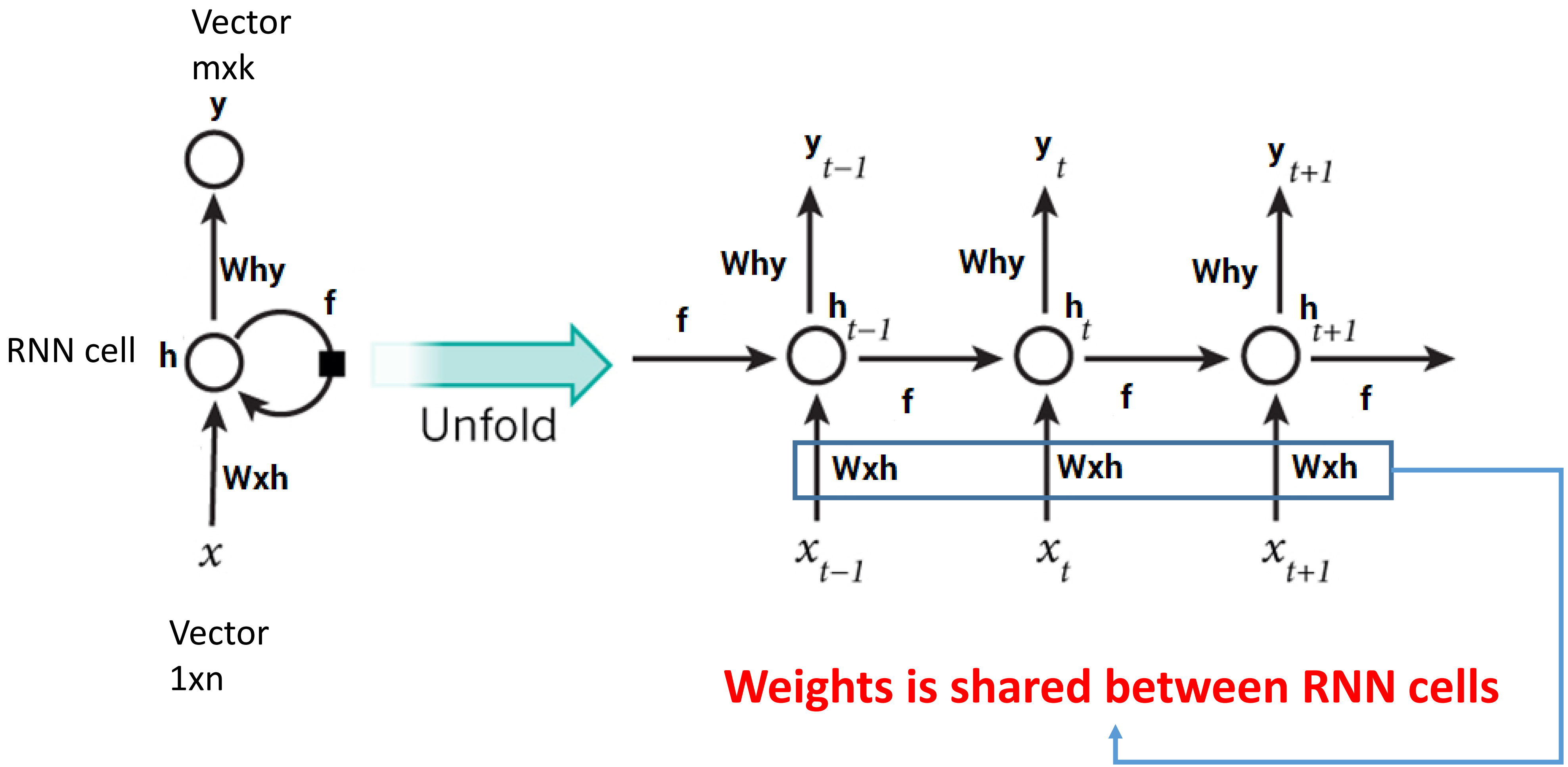


# Convolutional CNN cell – output: tensor



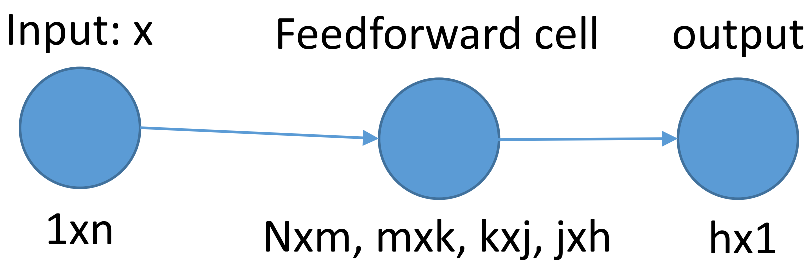

x
Convolution
Sigmoid
Pooling
$\left(f^{(pool)} \circ f^{(sigm)} \circ f_w^{(conv)}\right)(x)$
Forward propagation

# RNN cell – (RNN basic, GRU, LSTM)



Weights is shared between RNN cells

## Feed forward net



Input: x

Feedforward cell

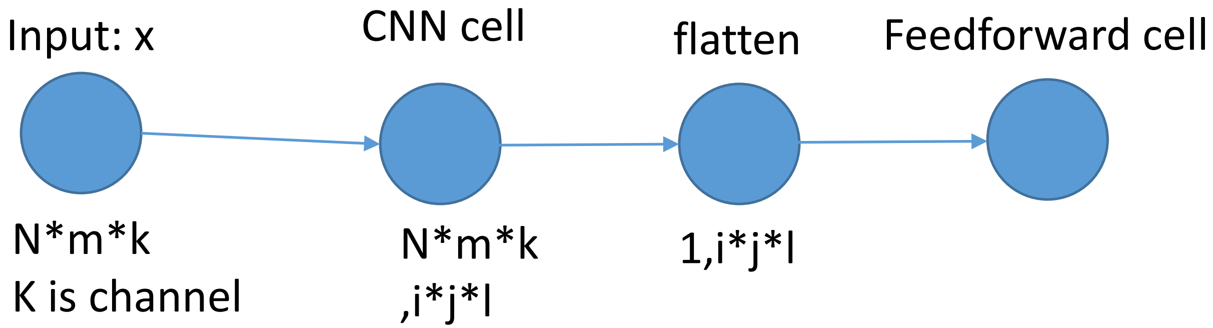
output

1xn

Nxm, mxk, kxj, jxh

hx1

## Convolutional  neuralnet



Input: x

CNN cell

flatten

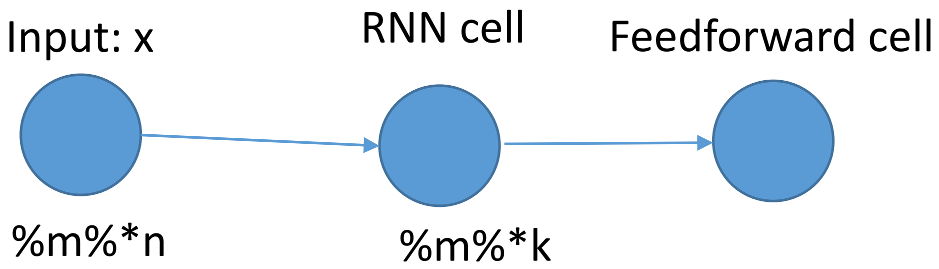
Feedforward cell

N*m*k
K is channel

N*m*k
,i*j*l

1,i*j*l

## Recurrent neural net



Input: x

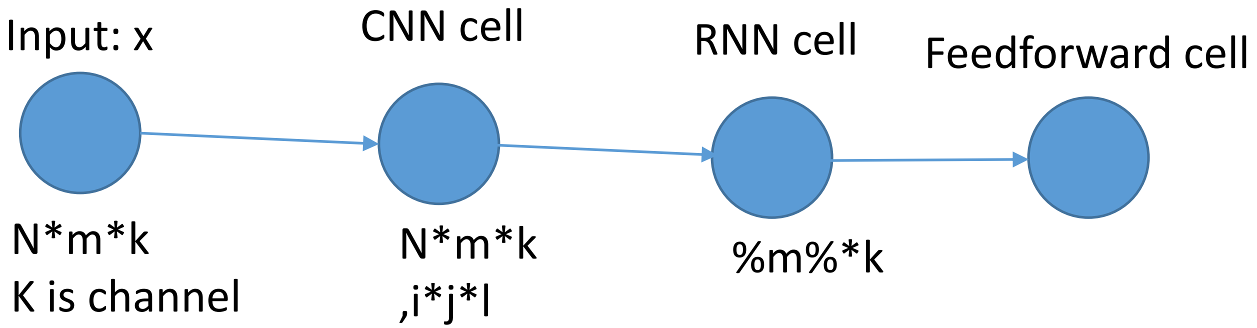
RNN cell

Feedforward cell

%m%*n

%m%*k

## Recurrent - Convolutional neural net



Input: x

CNN cell

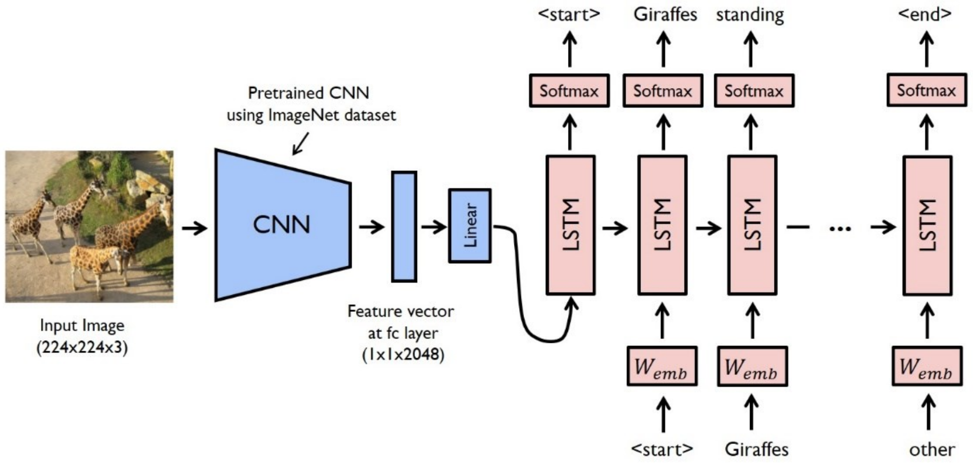
RNN cell

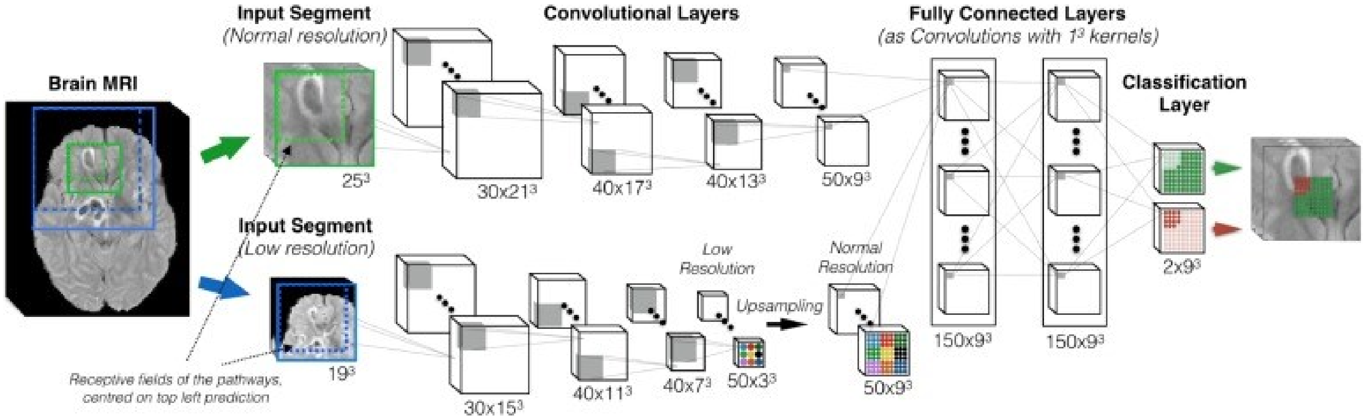
Feedforward cell

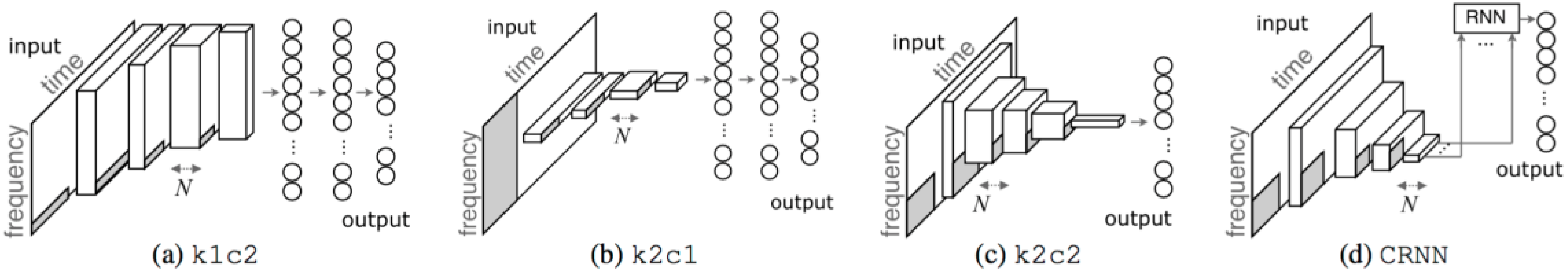
N*m*k
K is channel

N*m*k
,i*j*l

%m%*k







(a) `k1c2` (b) `k2c1` (c) `k2c2` (d) `CRNN`

**Fig. 1**: Block diagrams of `k1c2`, `k2c1`, `k2c2`, and `CRNN`. The grey areas illustrate the convolution kernels. $N$ refers to the number of feature maps of convolutional layers.

# Question?