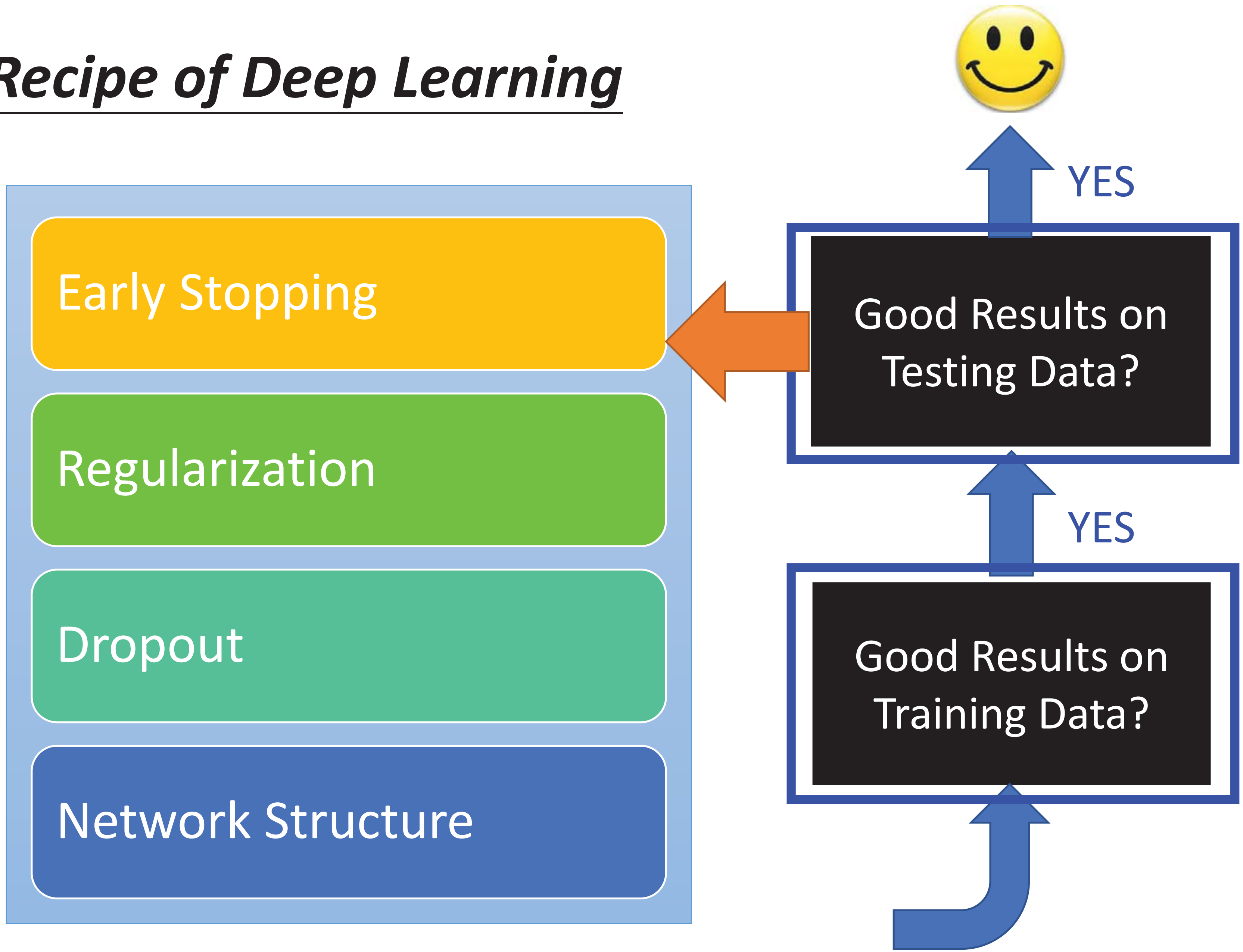
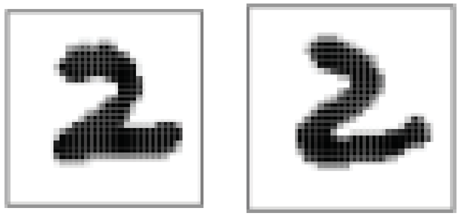
Recipe of Deep Learning



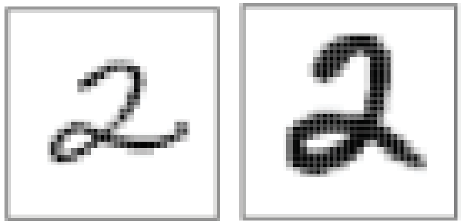
Why Overfitting?

- Training data and testing data can be different.

Training Data:



Testing Data:



Learning target is defined by the training data.

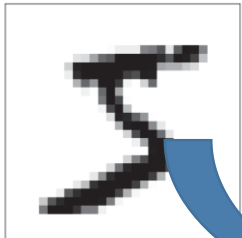
The parameters achieving the learning target do not necessary have good results on the testing data.

Panacea for Overfitting

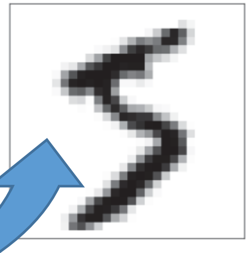
- Have more training data
- ***Create*** more training data (?)

Handwriting recognition:

Original
Training Data:



Created
Training Data:



Shift 15 °

Why Overfitting?

- For experiments, we added some noises to the testing data

```
        -1.36230370e-01,   1.03749340e-01,   1.15432226e-01,
         2.58670464e-01,   1.48774333e+00,   1.92885328e+00,
         1.70038673e+00,   2.46242981e+00,   1.21244572e+00,
        -9.28660713e-01,   3.63209761e-01,  -1.81327713e+00,
        -1.97910760e-01,   4.32874592e-01,  -5.40565788e-01,
         2.95630655e-01,   2.07984424e+00,  -1.84243292e+00,
        -5.11166017e-01,  -5.80935128e-01,   1.06273647e+00,
         1.80551097e-02,   2.27983997e-02,  -1.67979148e+00,
         8.12423001e-01,  -6.25888706e-01,  -1.25027082e+00,
         6.15135458e-01,  -1.21394611e-01,  -1.28089527e+00,
         3.24609806e-01,   6.70569391e-01,   1.49161323e-01,
         8.01573609e-01,   6.43116741e-01,  -9.37629233e-02,
         1.74677366e+00,   6.80996008e-01,  -7.03717611e-01,
         1.02079749e-01,   1.19505614e+00,  -2.77959386e-01,
        -5.21652916e-02,   3.53683601e-01,  -4.08310762e-01,
        -1.81042967e+00,  -9.03308062e-01,   1.05404509e+00,
        -9.80876877e-01,   3.52078891e-01,   6.65981840e-01,
         1.06550150e+00,  -2.28433613e-01,   3.64483904e-01,
        -1.51484666e+00,  -7.52612872e-02,  -2.97058082e-01,
        -7.27414382e-01,  -2.45875340e-01,  -1.27948942e-01,
        -3.69310620e-01,  -2.62300428e+00,   2.11585073e+00,
         6.85561585e-01,  -1.57443985e-01,   1.38128777e+00,
         6.84265587e-02,   3.12536292e-01,   4.54253185e-01,
        -7.88471875e-01,  -6.58403343e-02,  -1.41847985e+00,
        -1.39753340e-01,  -5.55354856e-01,  -5.01917779e-01,
         6.93118522e-01,  -2.45360497e-01,  -1.26943186e+00,
        -2.62323855e-01])

n [3]: x test[0]
```

Why Overfitting?

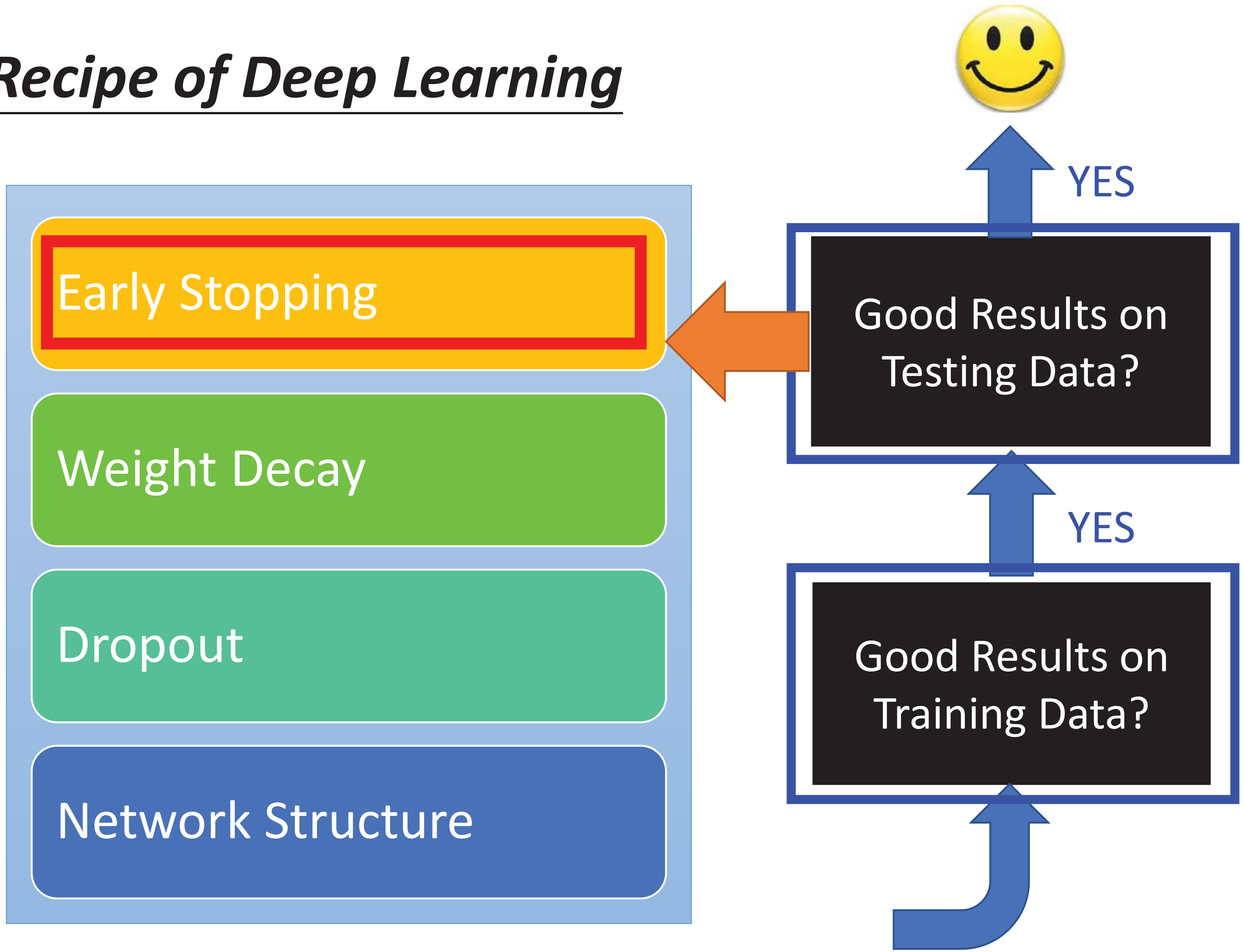
- For experiments, we added some noises to the testing data

Testing:

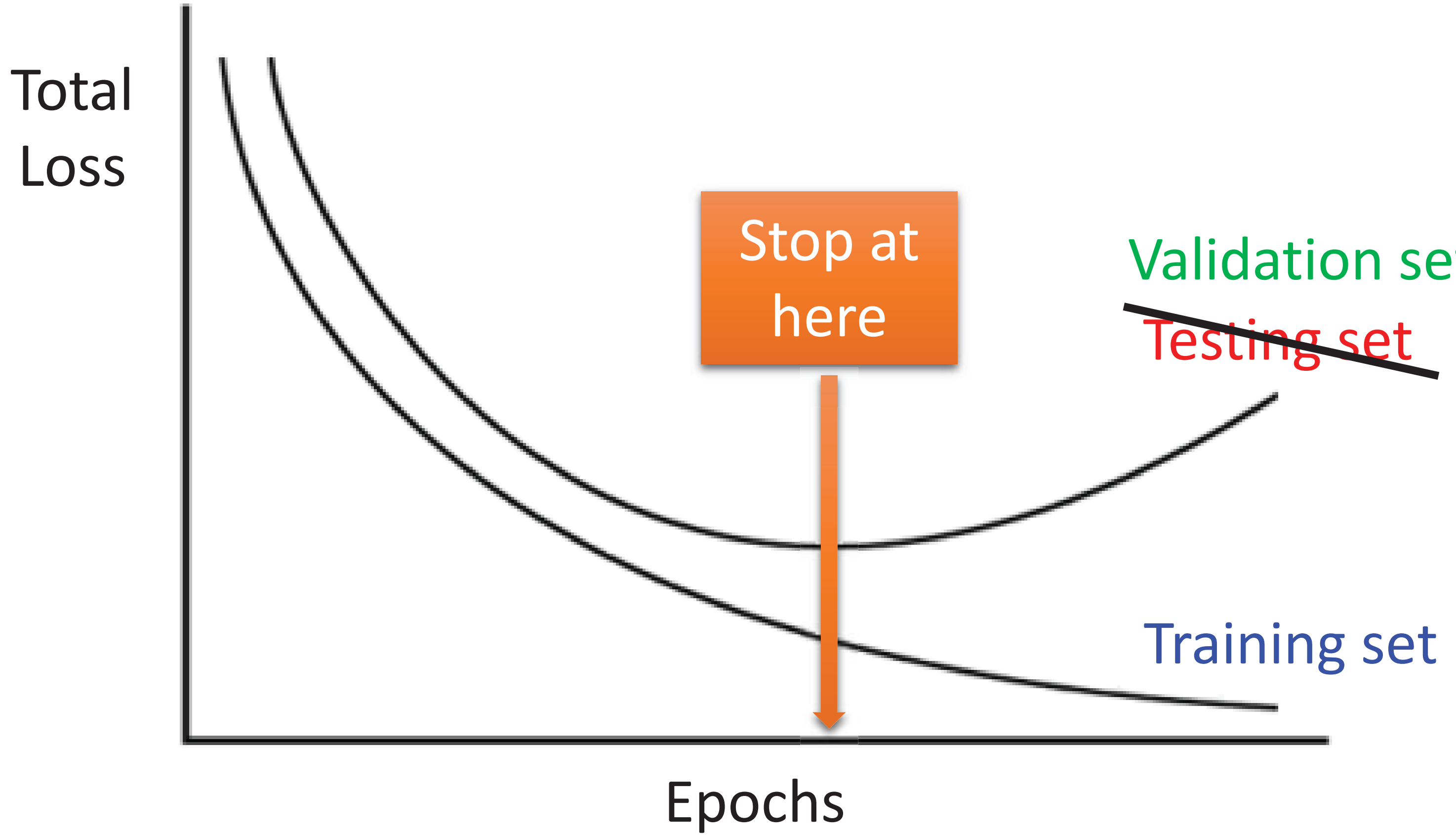
	Accuracy
Clean	0.97
Noisy	0.50

Training is not influenced.

Recipe of Deep Learning

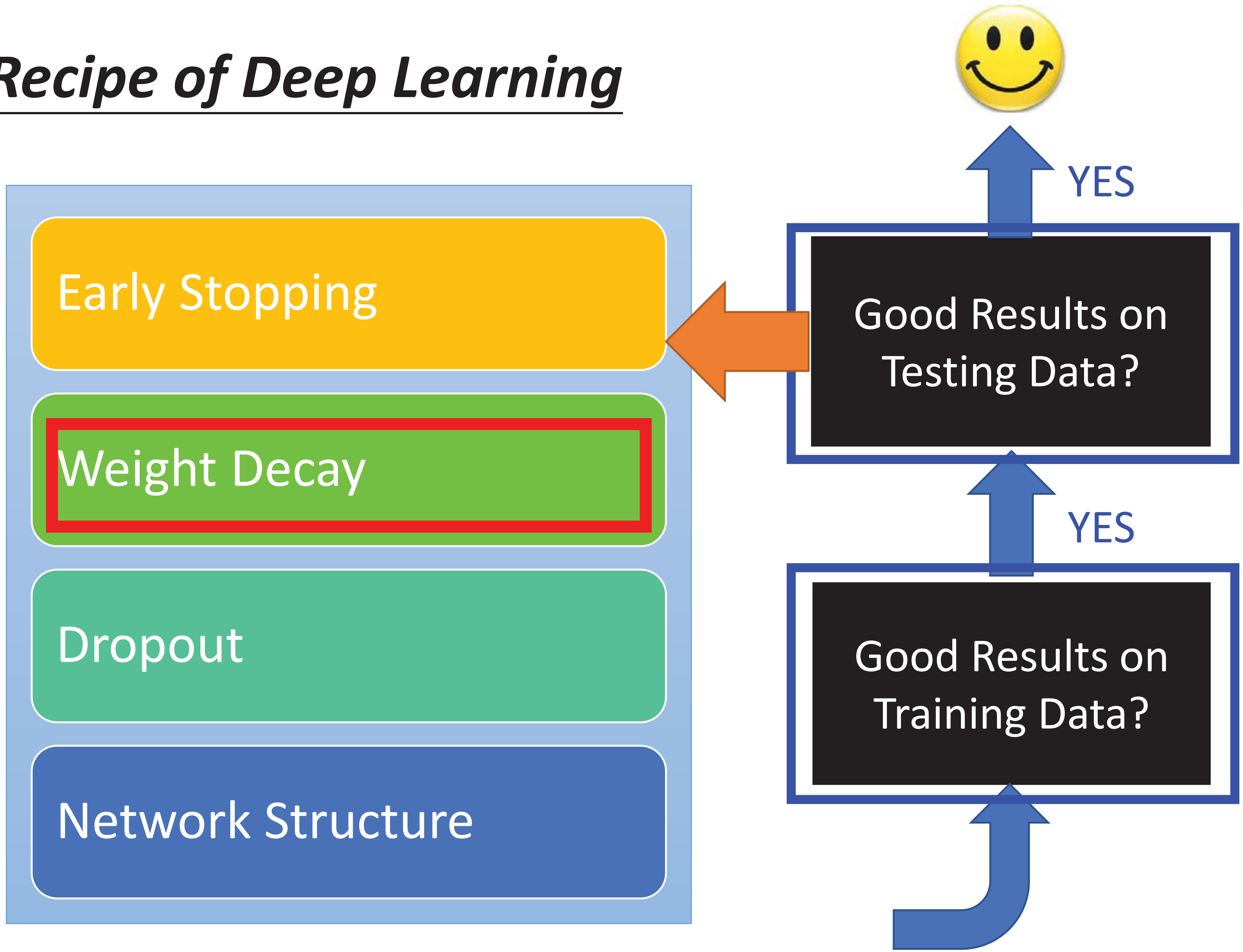


Early Stopping



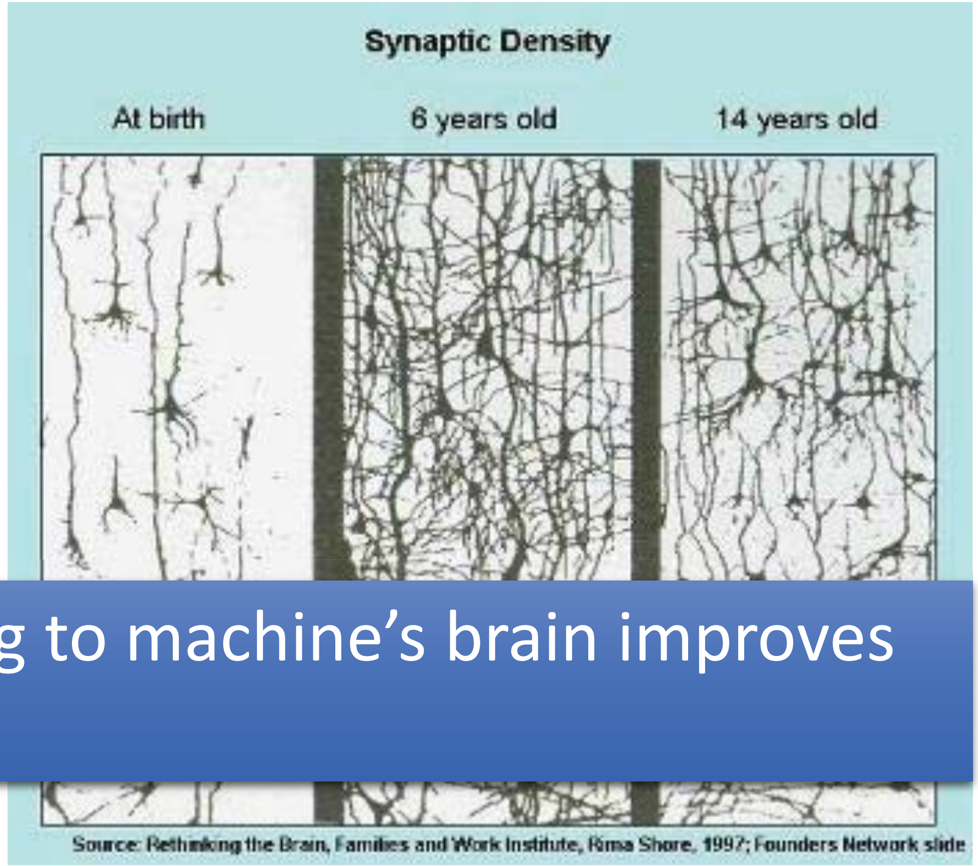
Keras: http://keras.io/getting-started/faq/#how-can-i-interrupt-training-when-the-validation-loss-isnt-decreasing-anymore

Recipe of Deep Learning



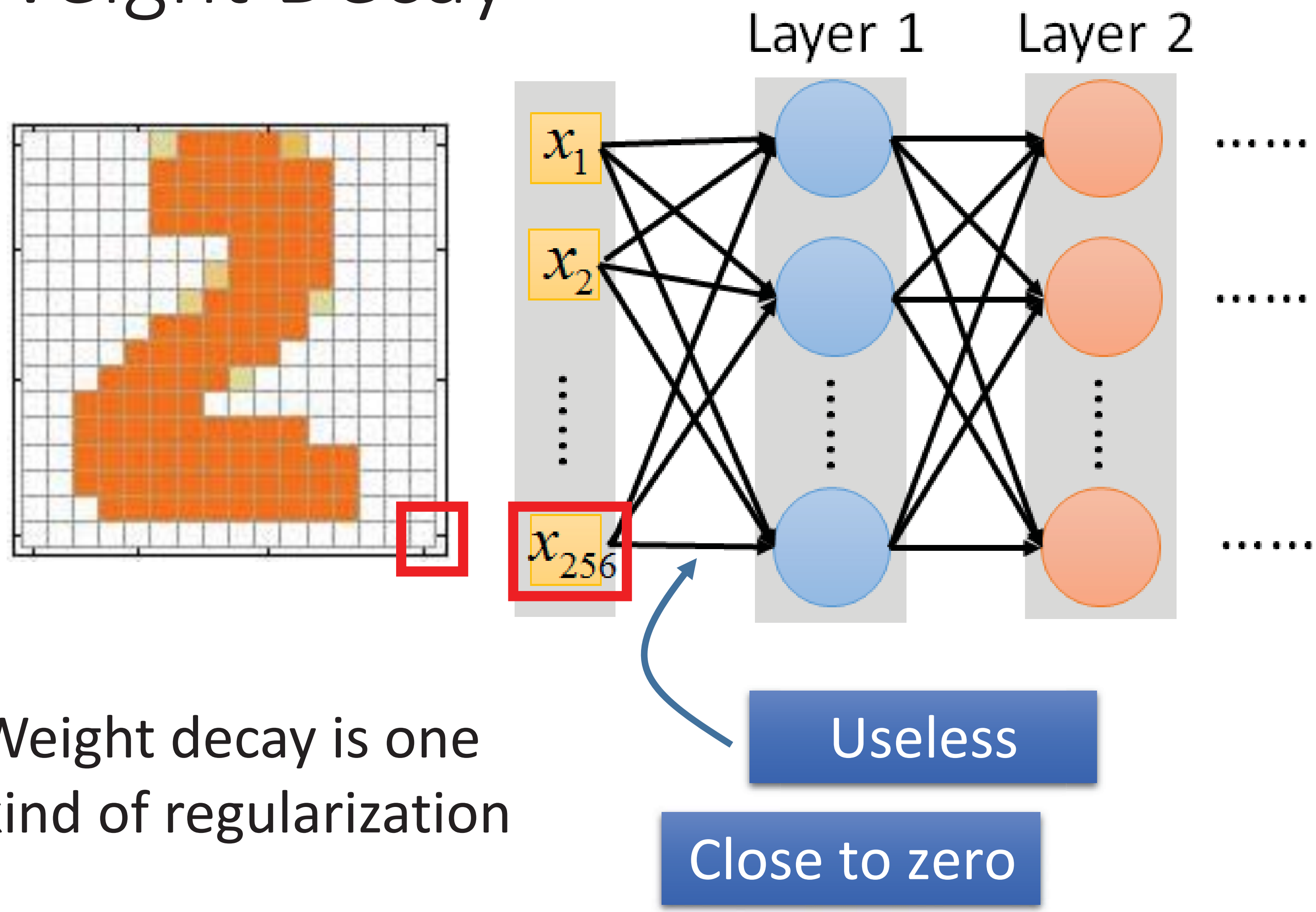
Weight Decay

- Our brain prunes out the useless link between neurons.



Doing the same thing to machine's brain improves the performance.

Weight Decay



Weight decay is one kind of regularization

Weight Decay

- Implementation

Original: $w \leftarrow w - \eta \frac{\partial L}{\partial w}$

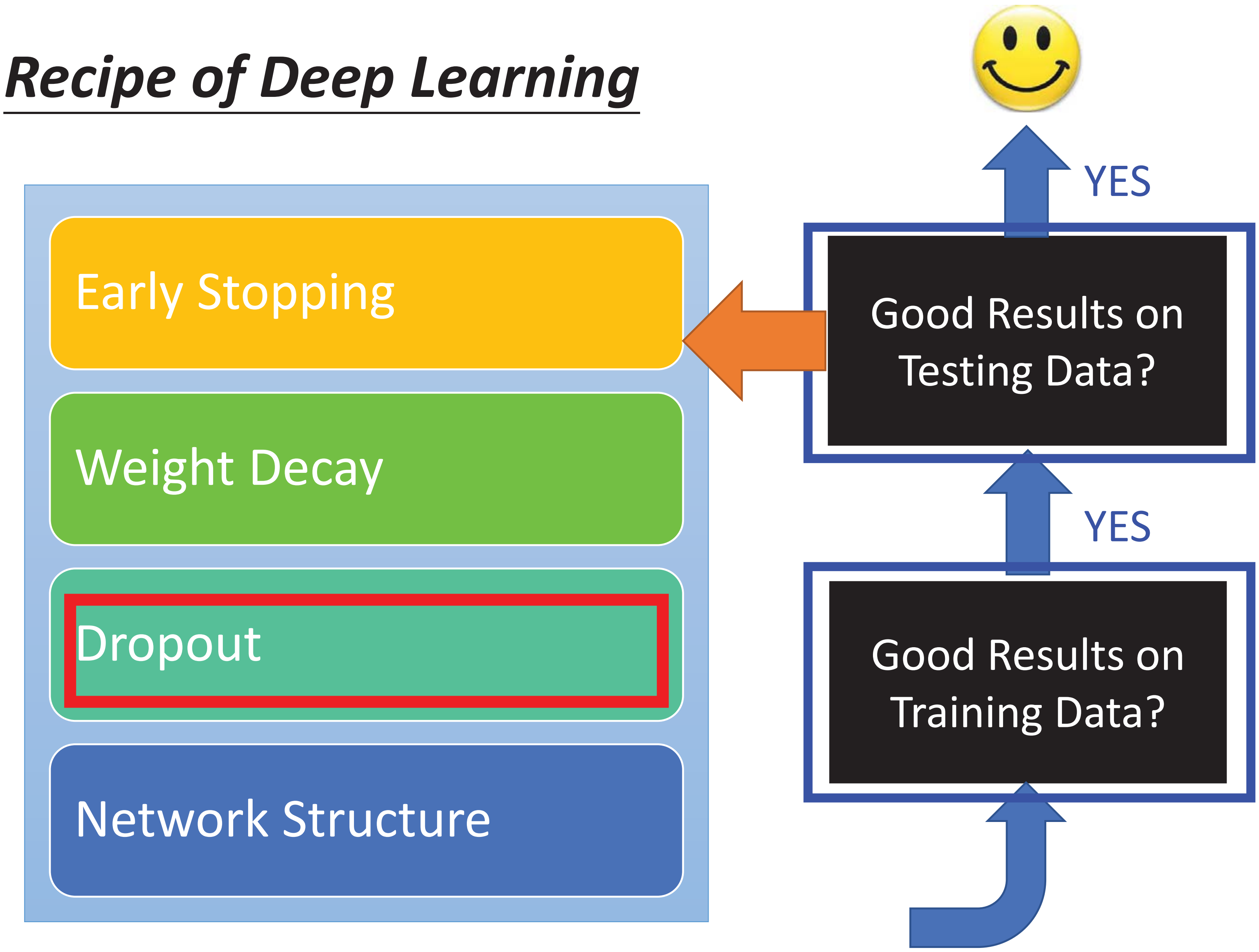
$$\lambda = 0.01$$

Weight Decay: $w \leftarrow \boxed{0.99}\, w - \eta \frac{\partial L}{\partial w}$

Smaller and smaller

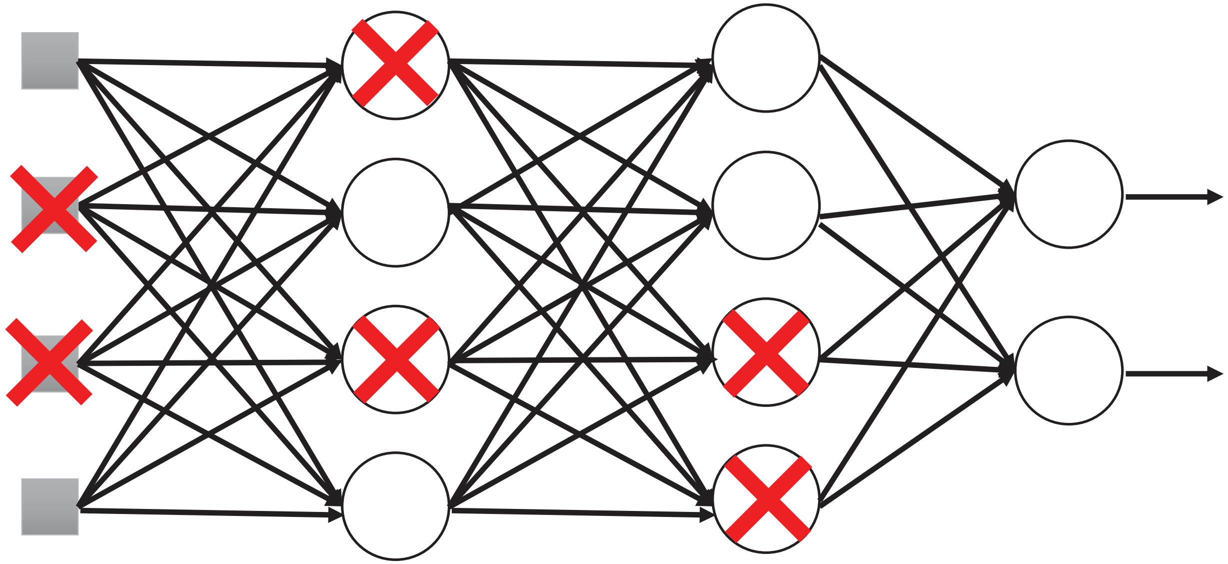
Keras: http://keras.io/regularizers/

Recipe of Deep Learning



Dropout

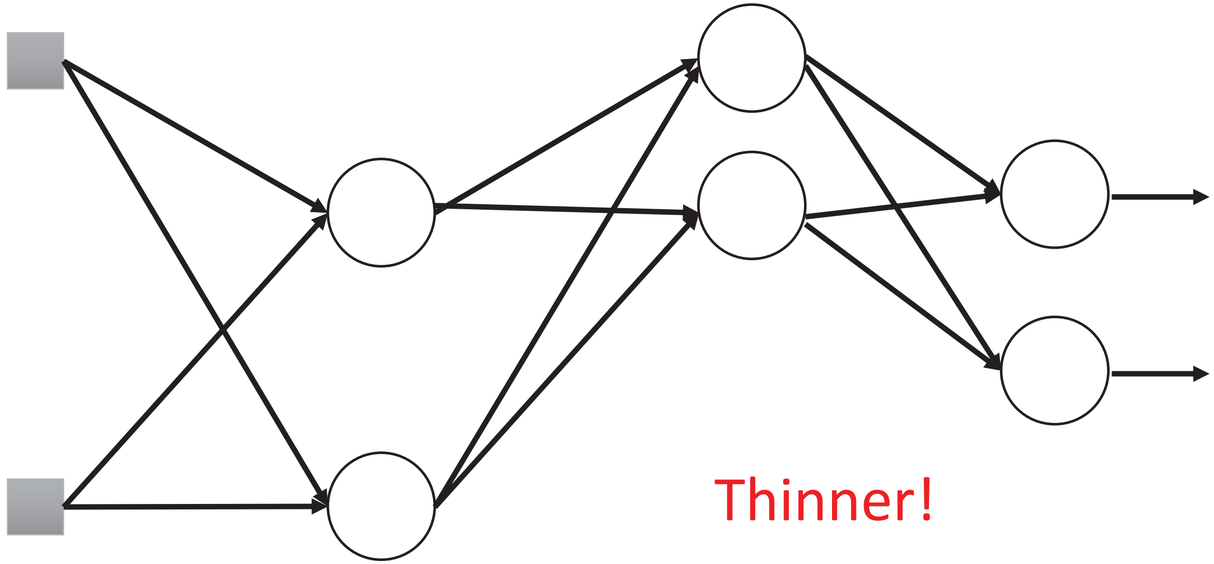
Training:



- ➢ **Each time before updating the parameters**
  - Each neuron has p% to dropout

Dropout

Training:

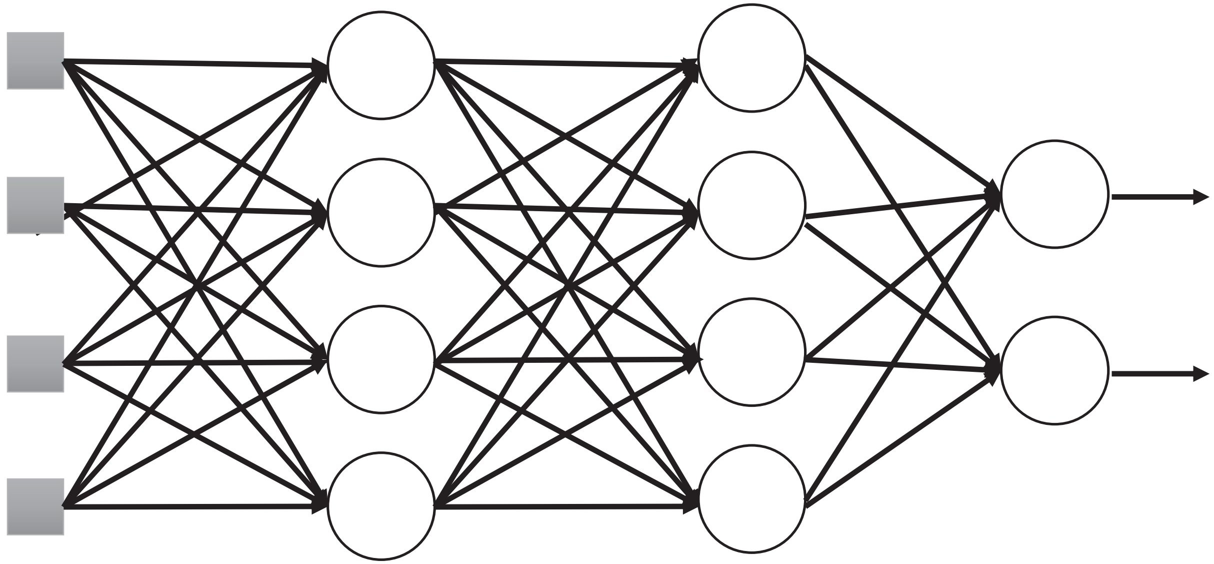


- **Each time before updating the parameters**
  - Each neuron has p% to dropout
    - **The structure of the network is changed.**
  - Using the new network for training

For each mini-batch, we resample the dropout neurons

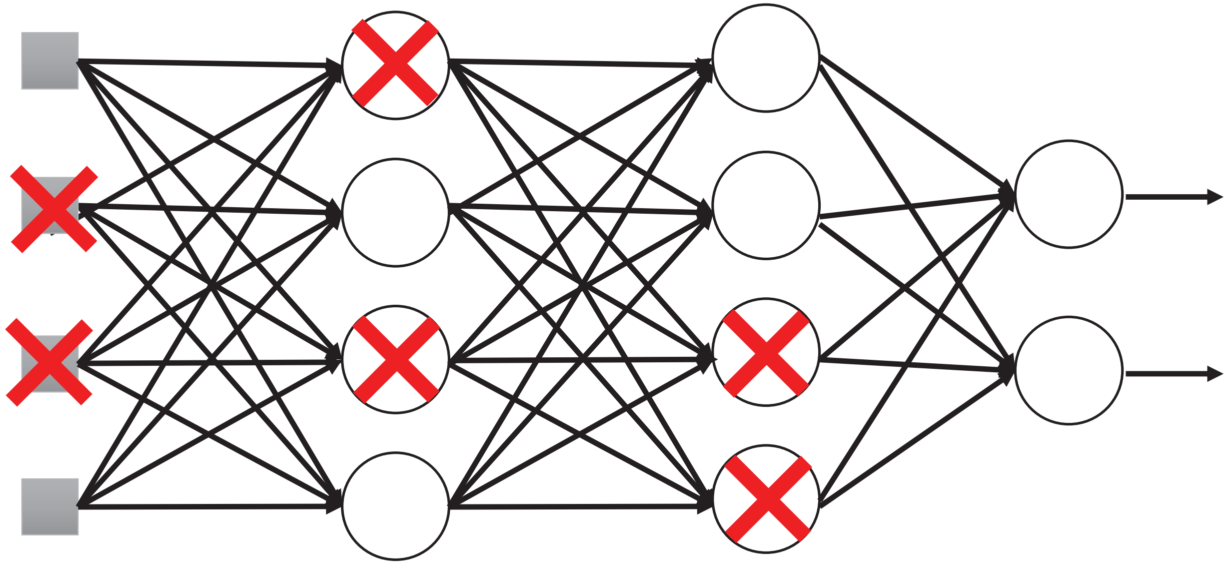
Dropout

Testing:



- **No dropout**
  - If the dropout rate at training is p%, all the weights times (1-p)%
  - Assume that the dropout rate is 50%. If a weight $w = 1$ by training, set $w = 0.5$ for testing.

Dropout - Intuitive Reason



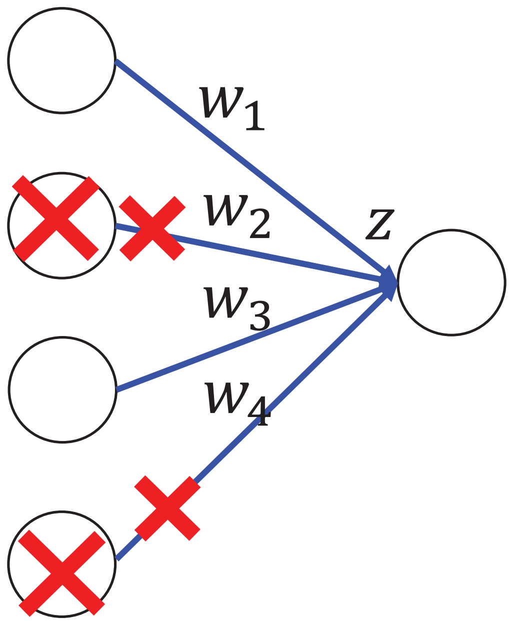
- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

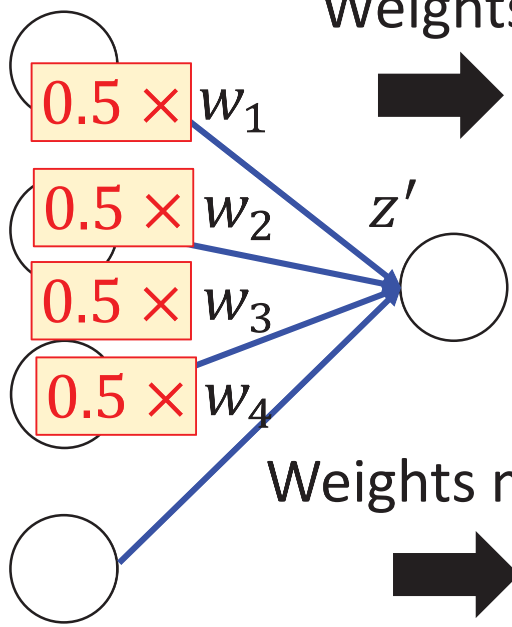
<u>Training of Dropout</u>

Assume dropout rate is 50%



<u>Testing of Dropout</u>

No dropout



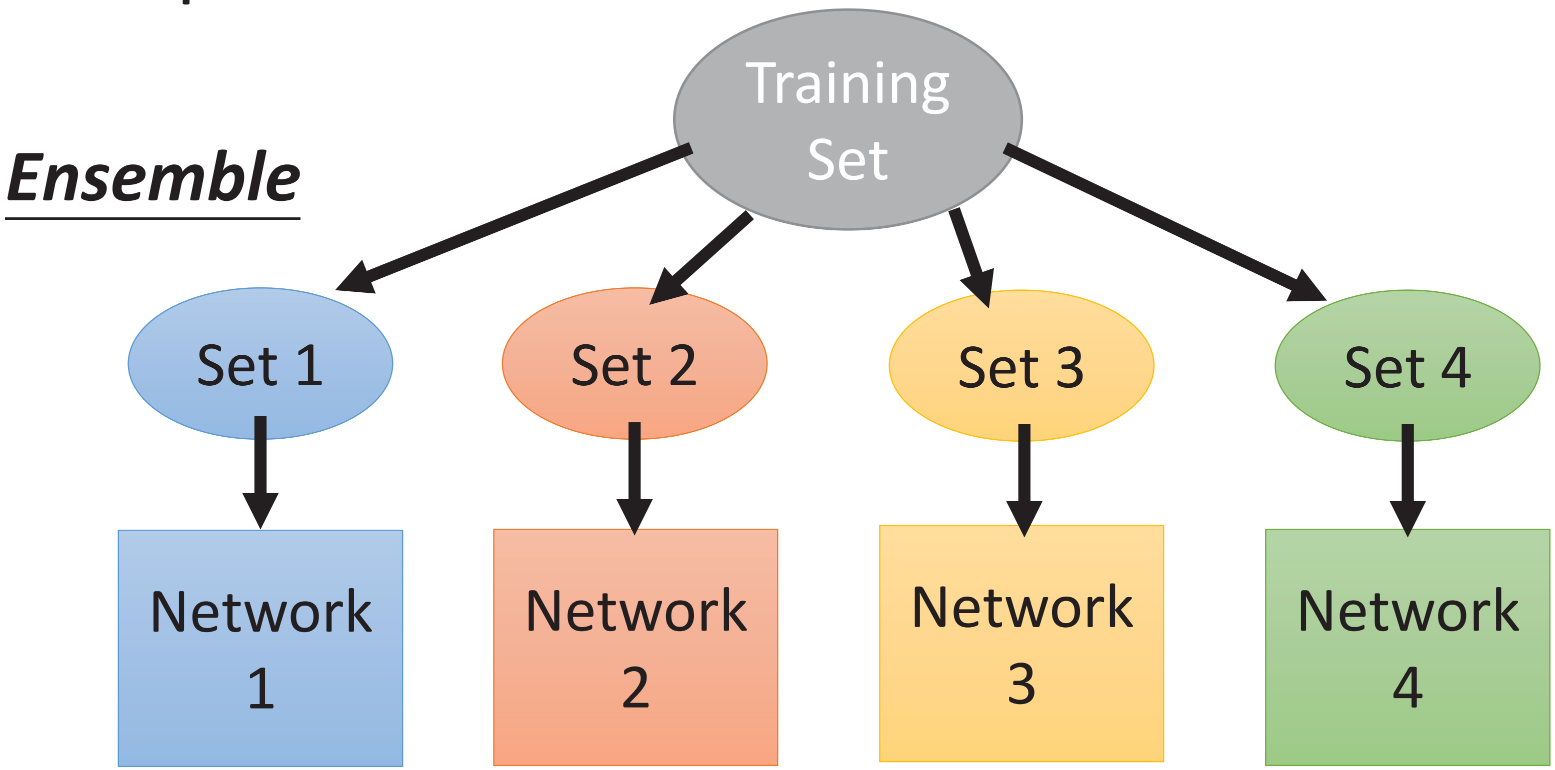
Weights from training

$\Rightarrow z' \approx 2z$

Weights multiply (1-p)%

$\Rightarrow z' \approx z$

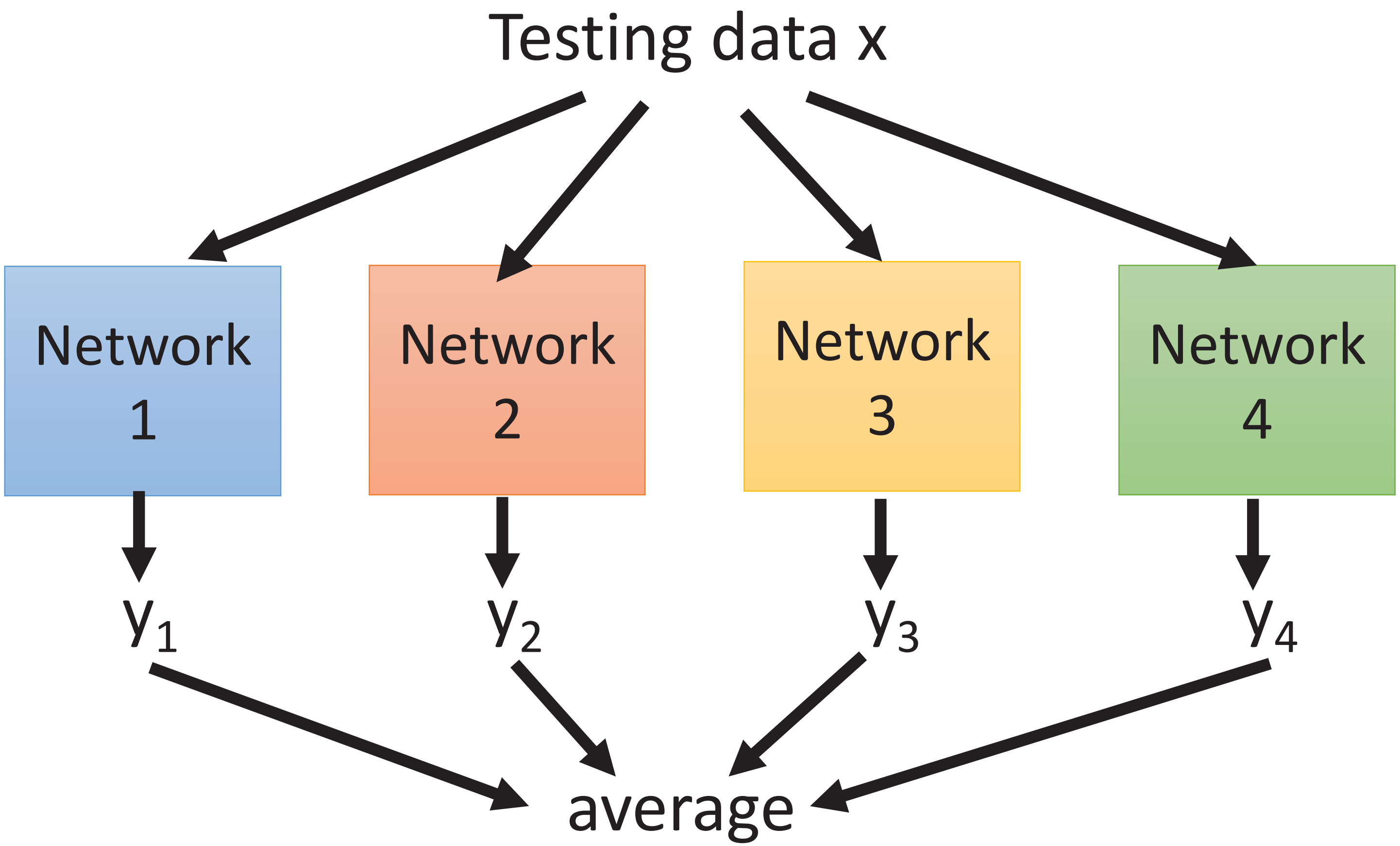
Dropout is a kind of ensemble.



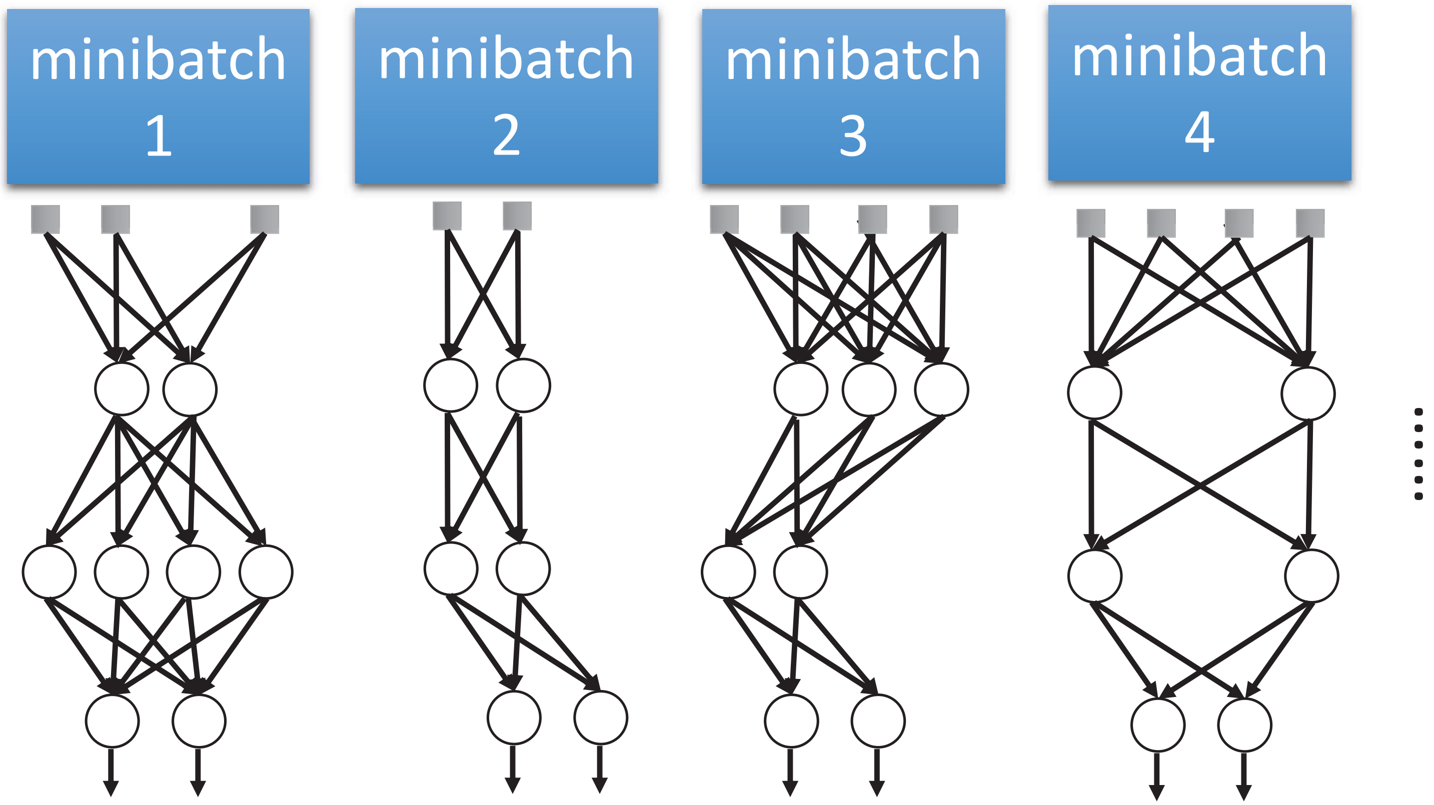
Train a bunch of networks with different structures

Dropout is a kind of ensemble.

Ensemble



Dropout is a kind of ensemble.



Training of Dropout

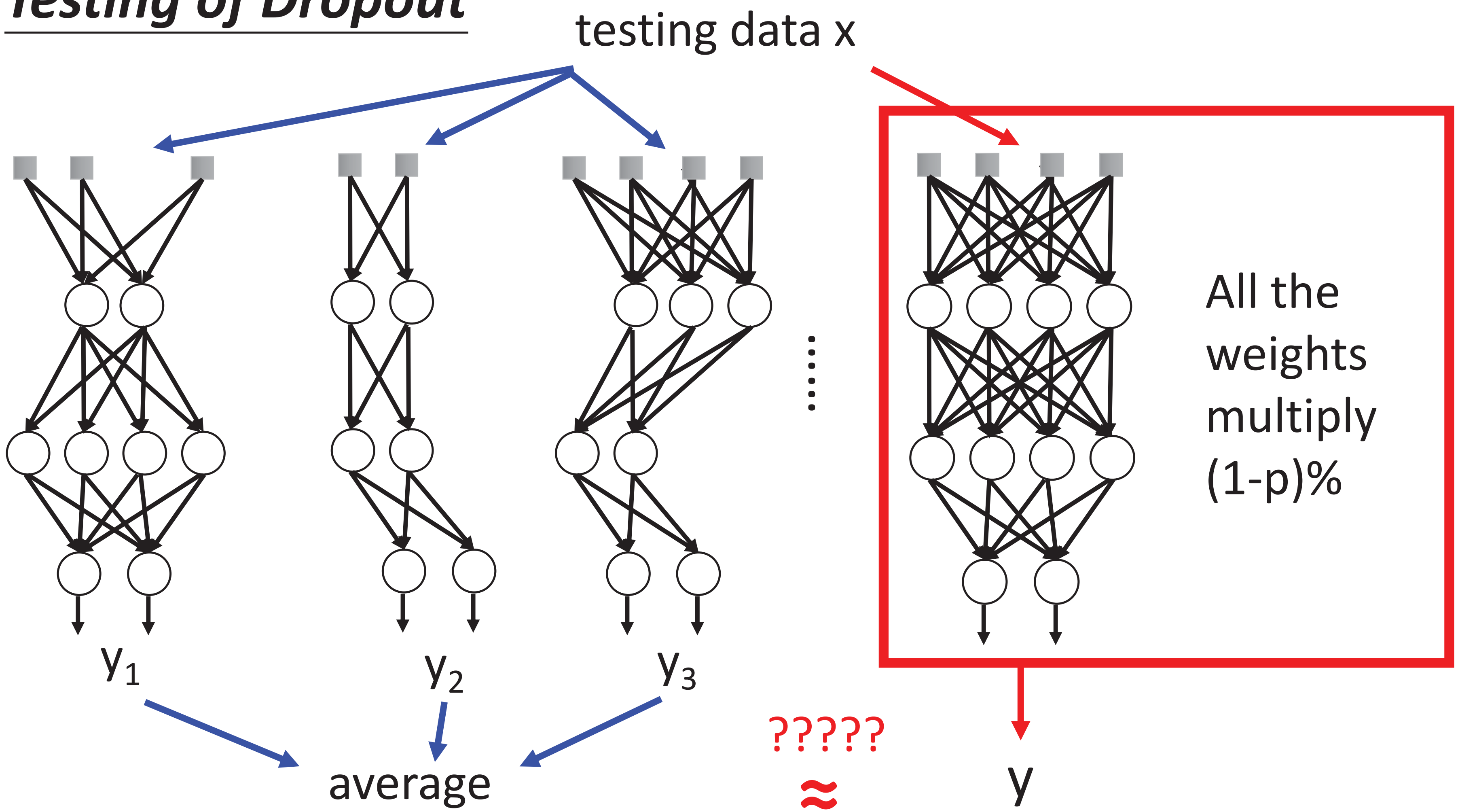
M neurons

2^M possible networks

- ➢Using one mini-batch to train one network
- ➢Some parameters in the network are shared

Dropout is a kind of ensemble.

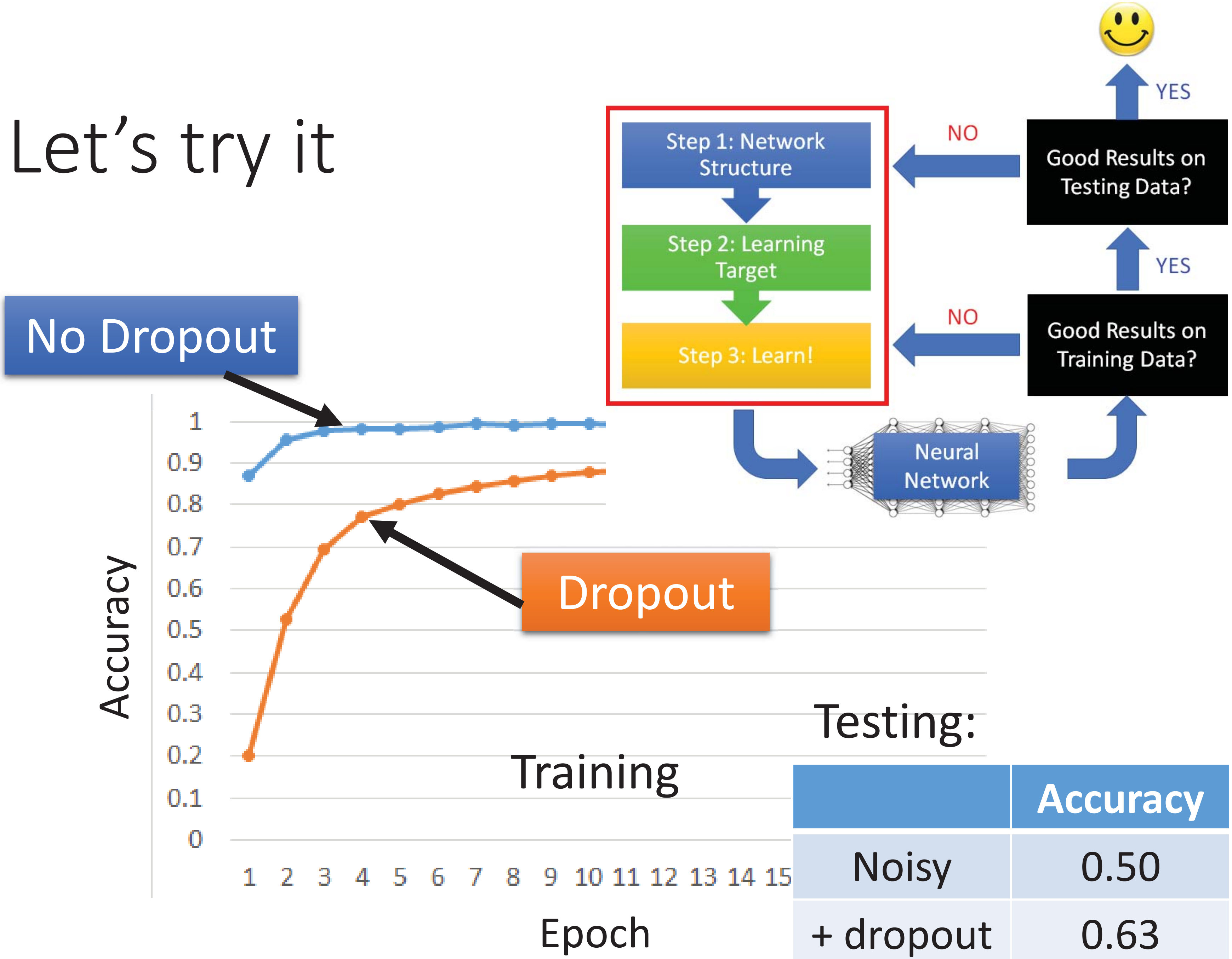
Testing of Dropout



More about dropout

- More reference for dropout [Nitish Srivastava, JMLR’14] [Pierre Baldi, NIPS’13][Geoffrey E. Hinton, arXiv’12]
- Dropout works better with Maxout [Ian J. Goodfellow, ICML’13]
- Dropconnect [Li Wan, *ICML’13*]
  - Dropout delete neurons
  - Dropconnect deletes the connection between neurons
- Annealed dropout [S.J. Rennie, SLT’14]
  - Dropout rate decreases by epochs
- Standout [J. Ba, NISP’13]
  - Each neural has different dropout rate

Let's try it



Testing:

	Accuracy
Noisy	0.50
+ dropout	0.63

Variants of Neural Networks

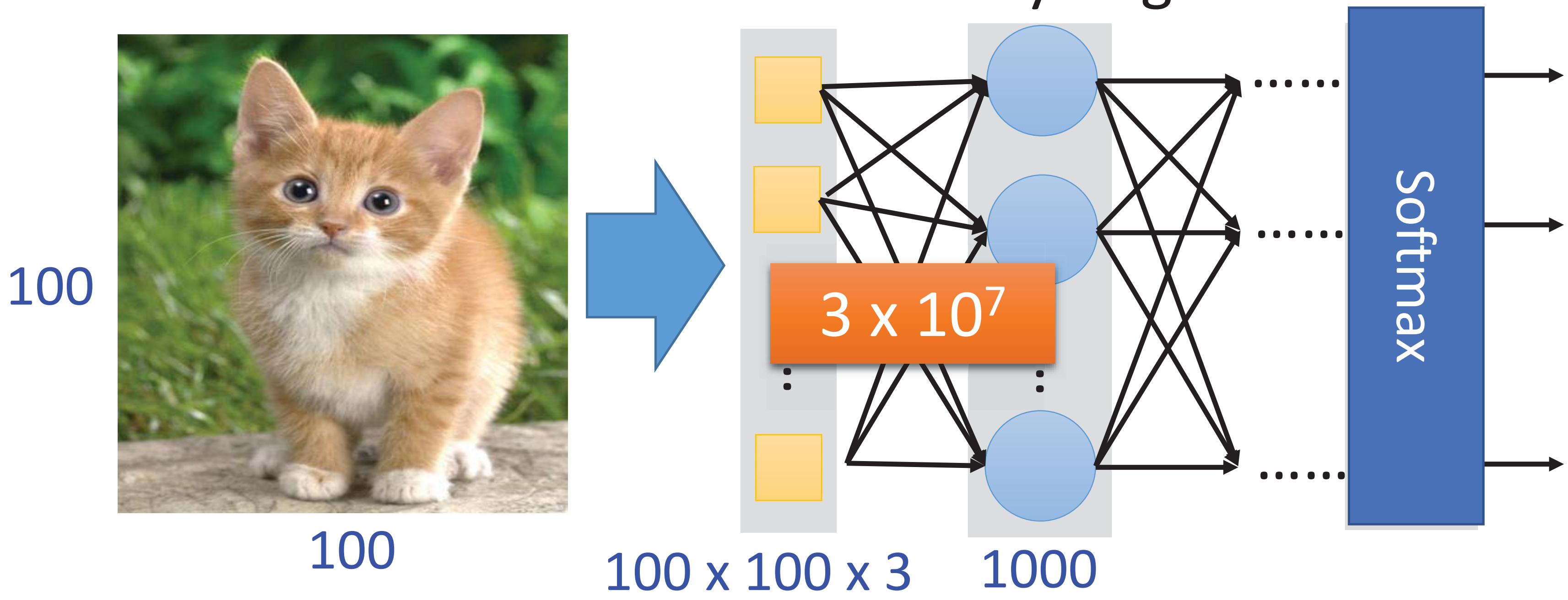
Convolutional Neural Network (CNN)

Widely used in image processing

Recurrent Neural Network (RNN)

Why CNN for Image?

- When processing image, the first layer of fully connected network would be very large



Can the fully connected network be simplified by considering the properties of image recognition?

Why CNN for Image

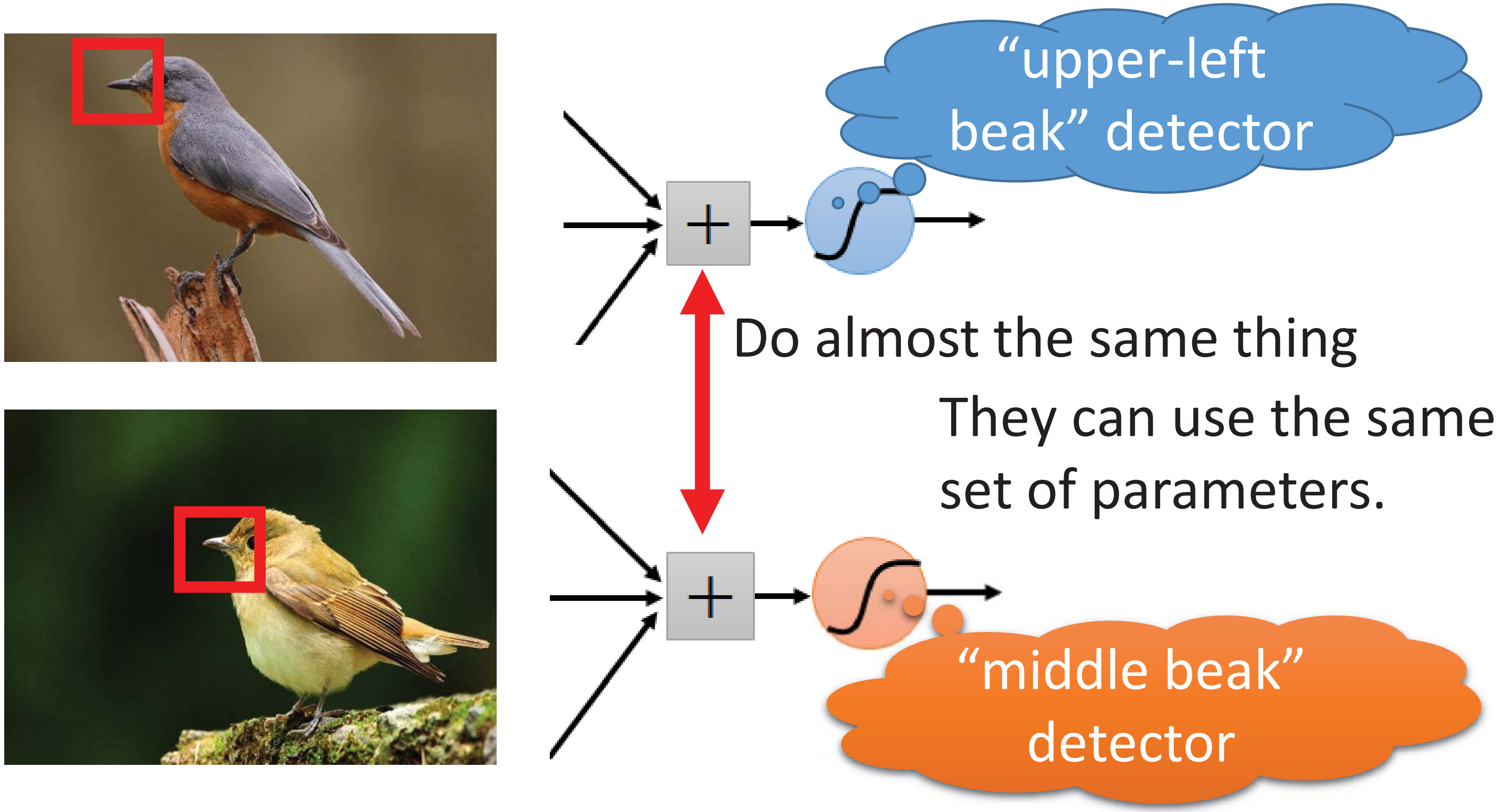
- Some patterns are much smaller than the whole image

A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters

Why CNN for Image

- The same patterns appear in different regions.



Why CNN for Image

- Subsampling the pixels will not change the object

bird

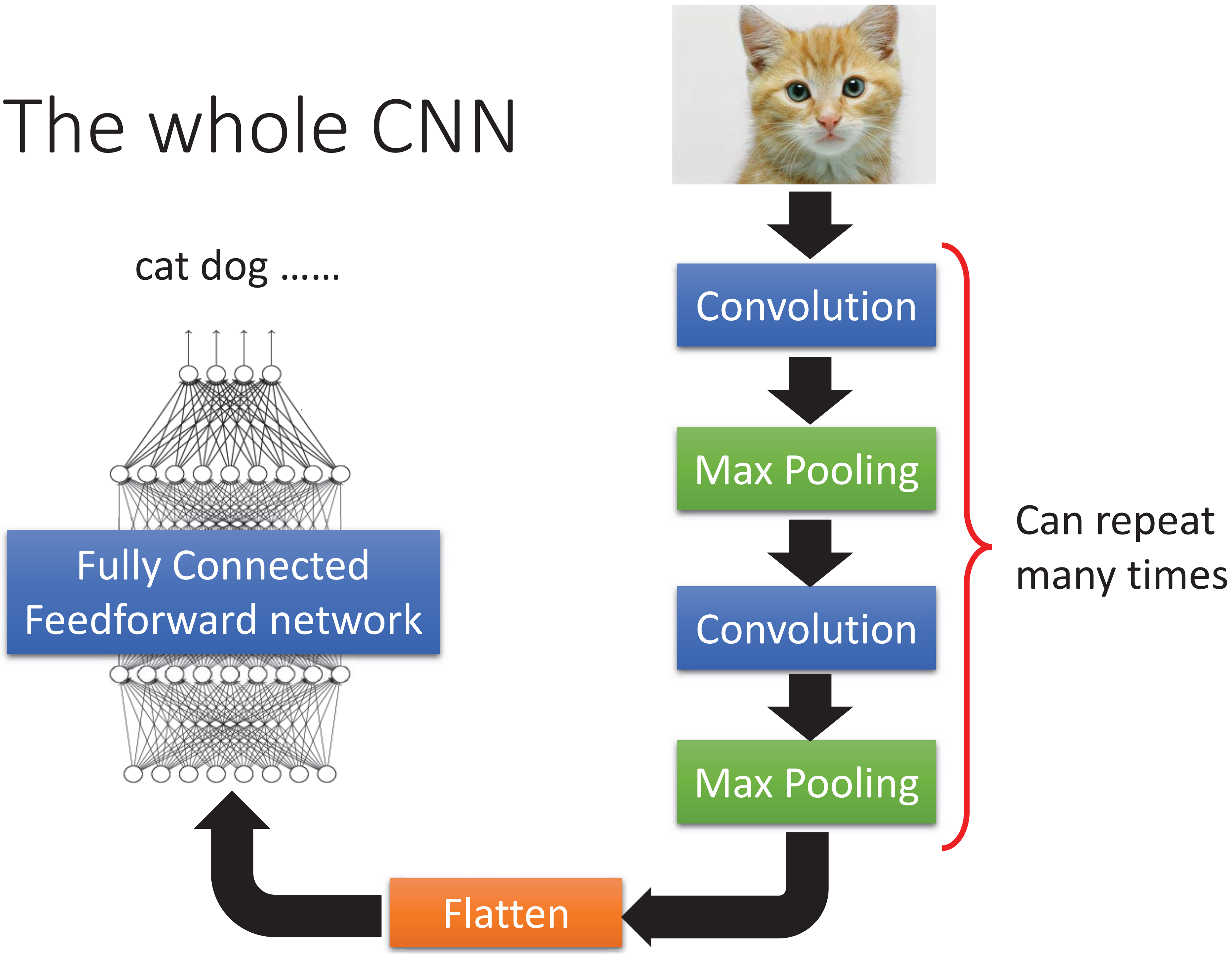
subsampling

bird

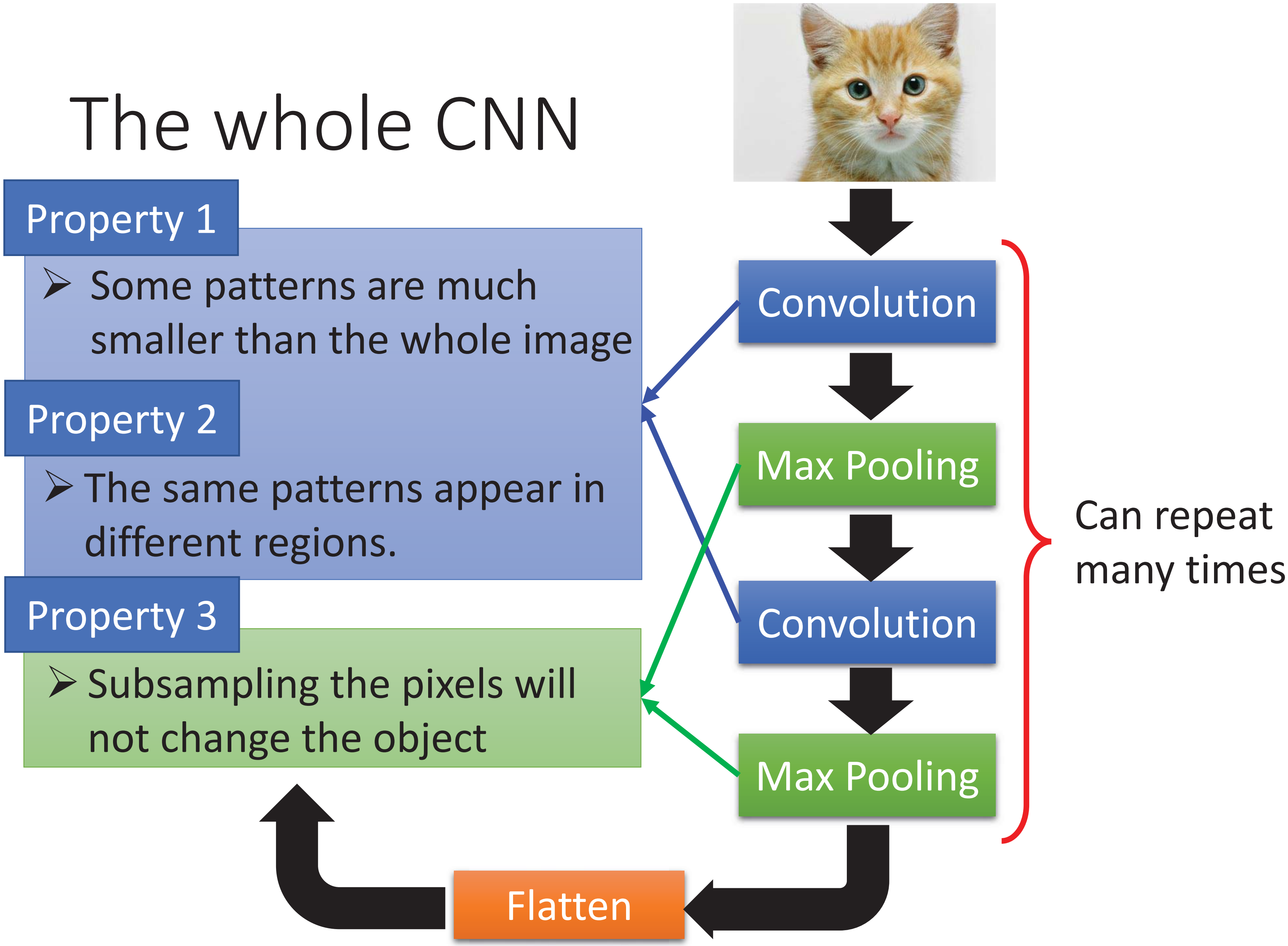
We can subsample the pixels to make image smaller

Less parameters for the network to process the image

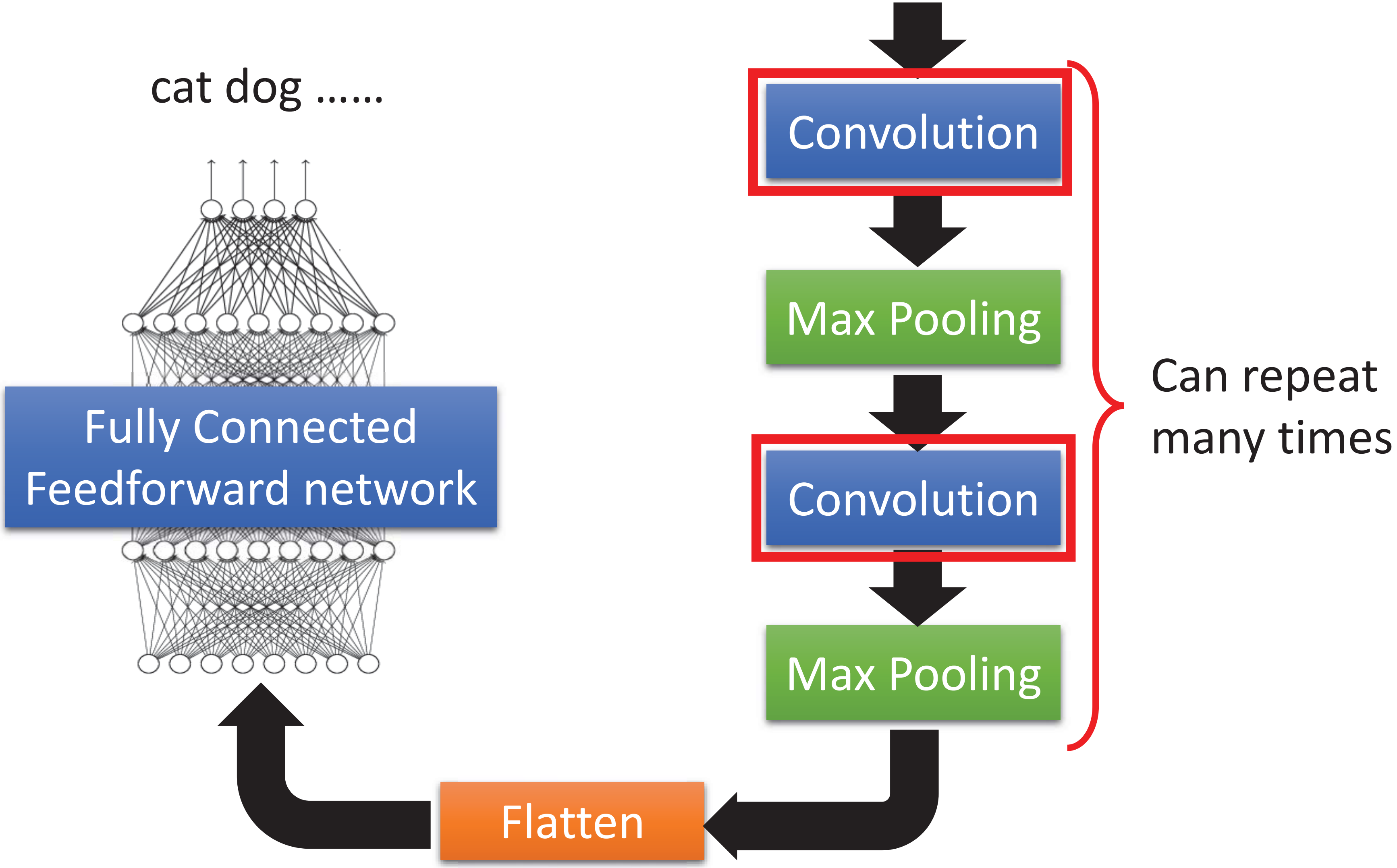
The whole CNN



The whole CNN



The whole CNN



CNN – Convolution

Those are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Matrix

⋮

Property 1 Each filter detects a small pattern (3 x 3).

CNN – Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -1

CNN – Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -3

We set stride=1 below

CNN – Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Property 2

CNN – Convolution

-1	1	-1
-1	1	-1
-1	1	-1

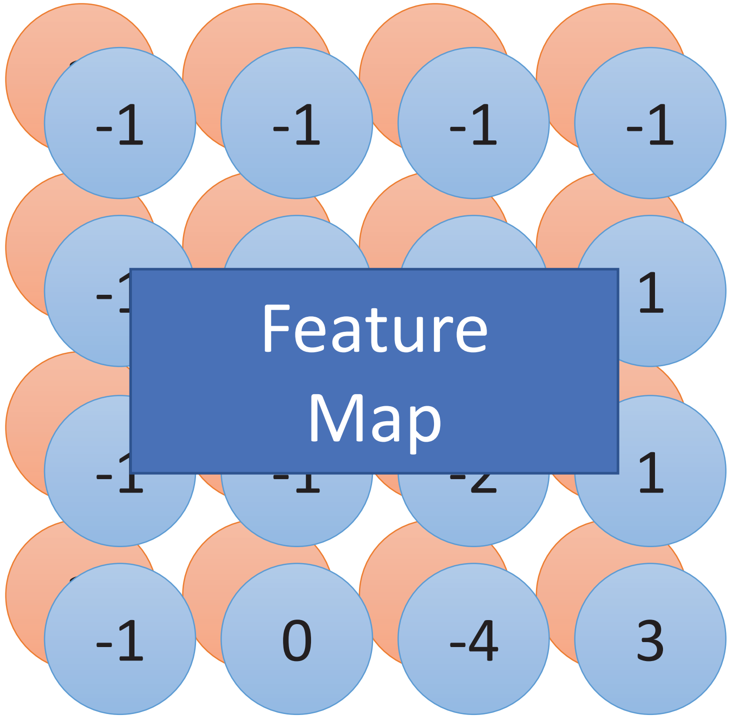
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Do the same process for every filter



4 x 4 image

CNN – Zero Padding

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

0	0	0					
0	1	0	0	0	0	1	
0	0	1	0	0	1	0	
	0	0	1	1	0	0	
	1	0	0	0	1	0	
	0	1	0	0	1	0	0
	0	0	1	0	1	0	0
					0	0	0

6 x 6 image

You will get another 6 x 6 images in this way

→ Zero padding

CNN – Colorful image

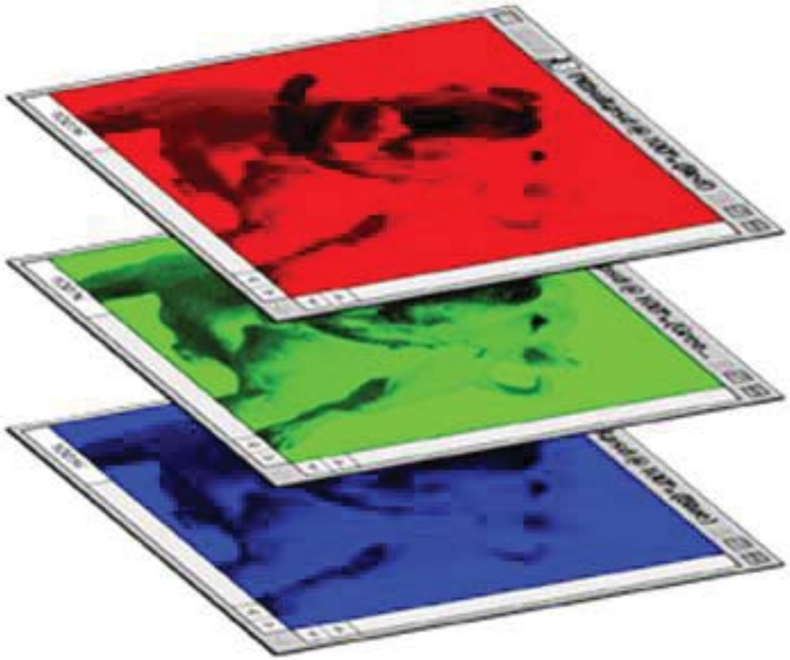
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

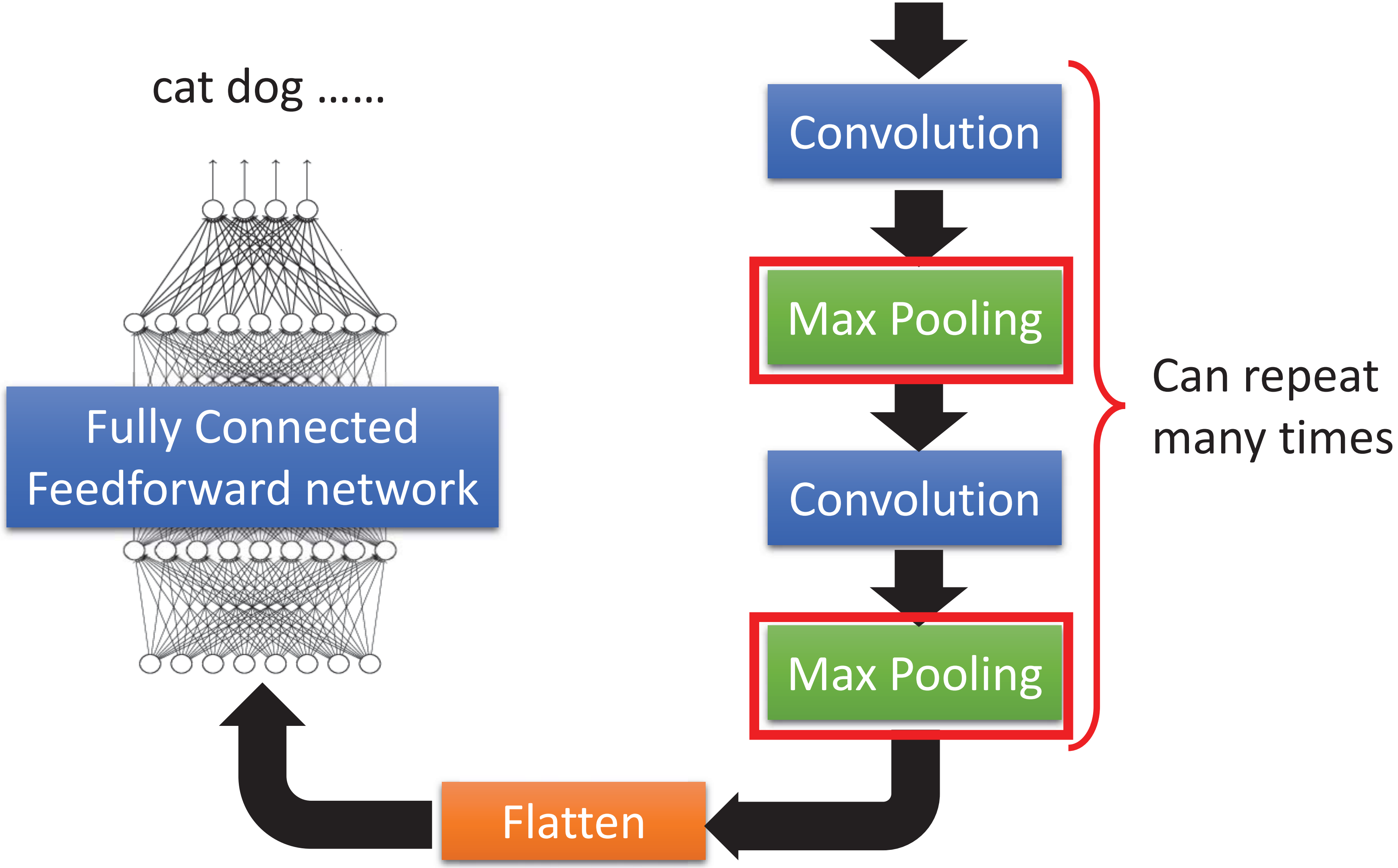
Filter 2

Colorful image



1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

The whole CNN



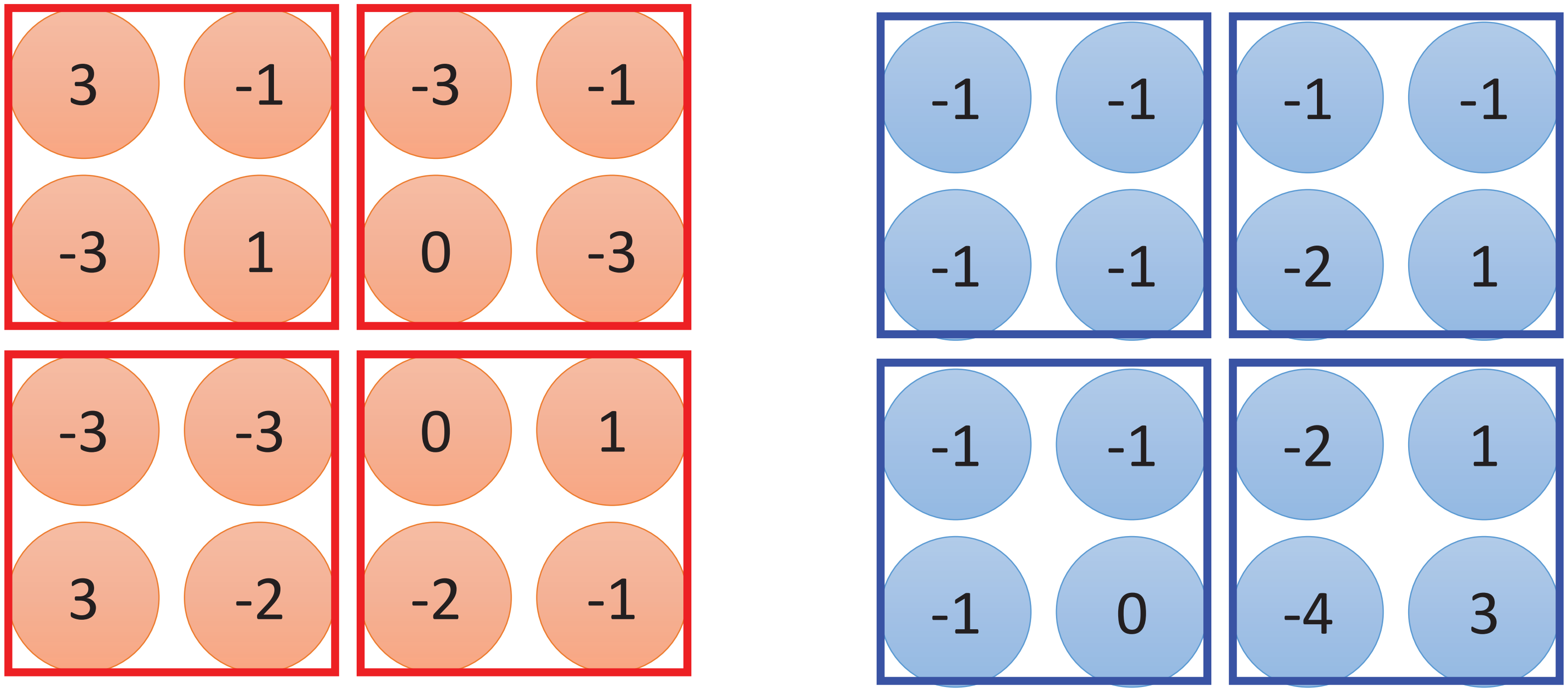
CNN – Max Pooling

Filter 1

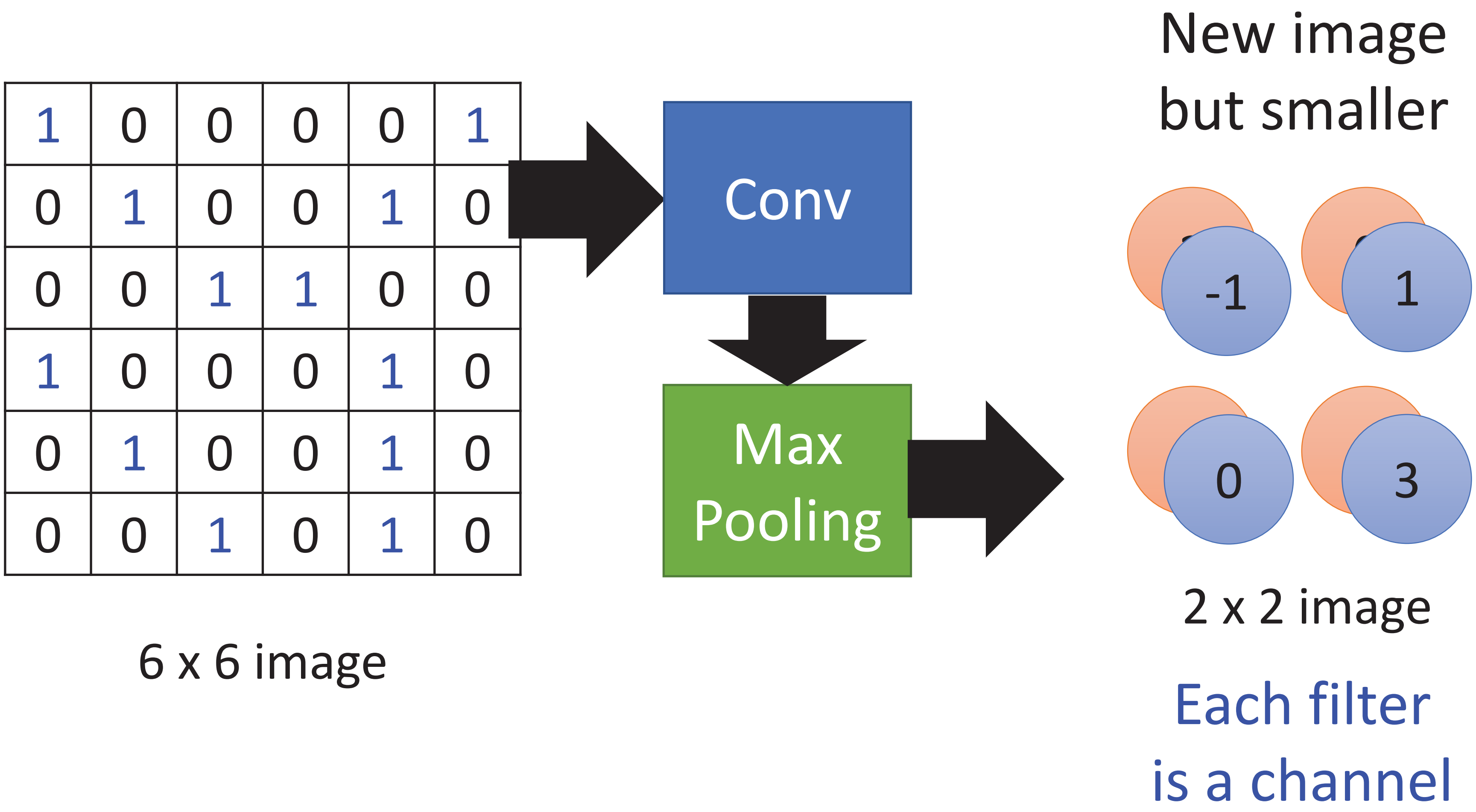
1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

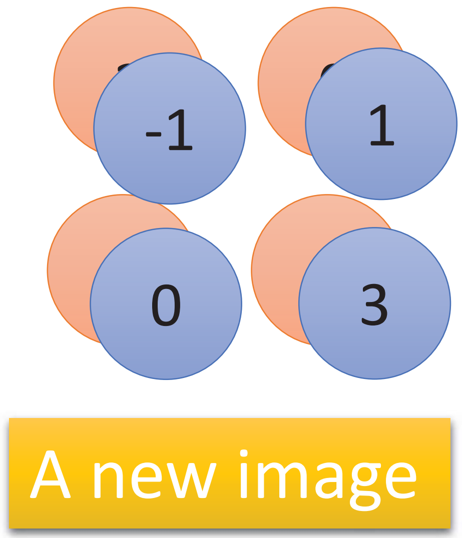
-1	1	-1
-1	1	-1
-1	1	-1

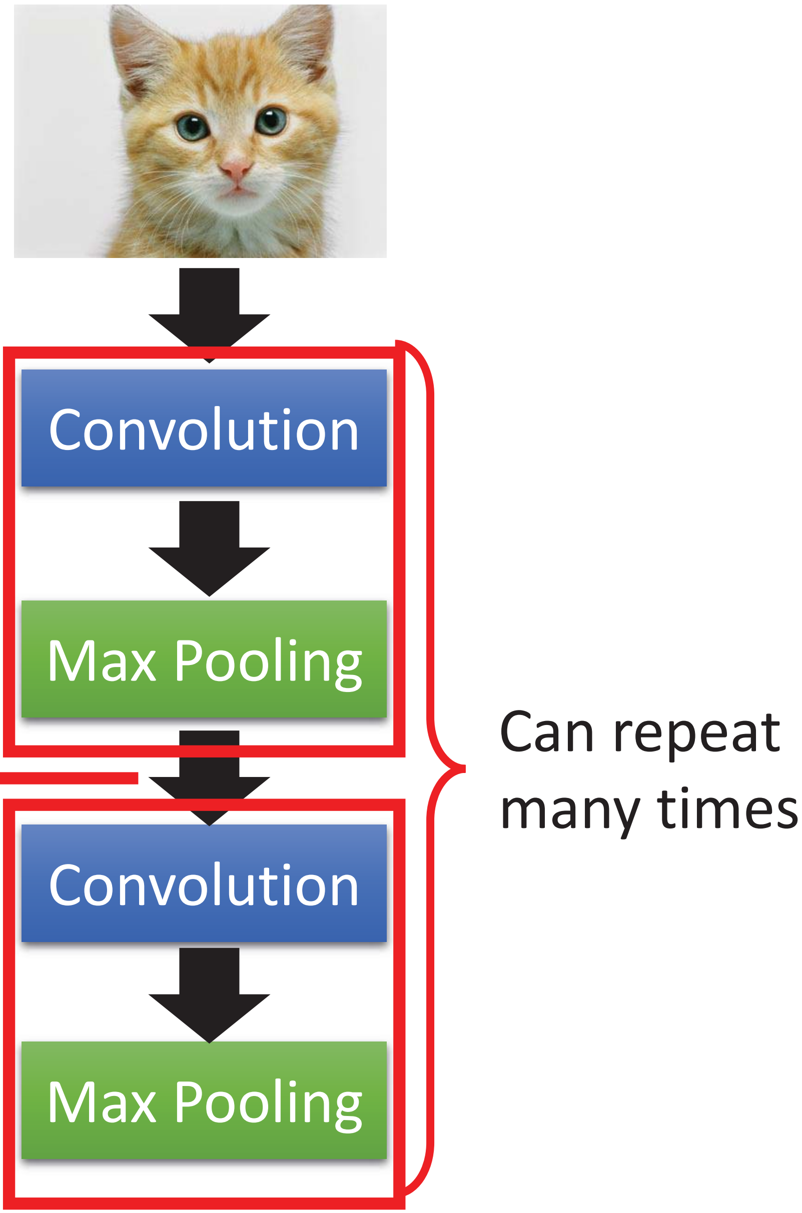


CNN – Max Pooling



The whole CNN

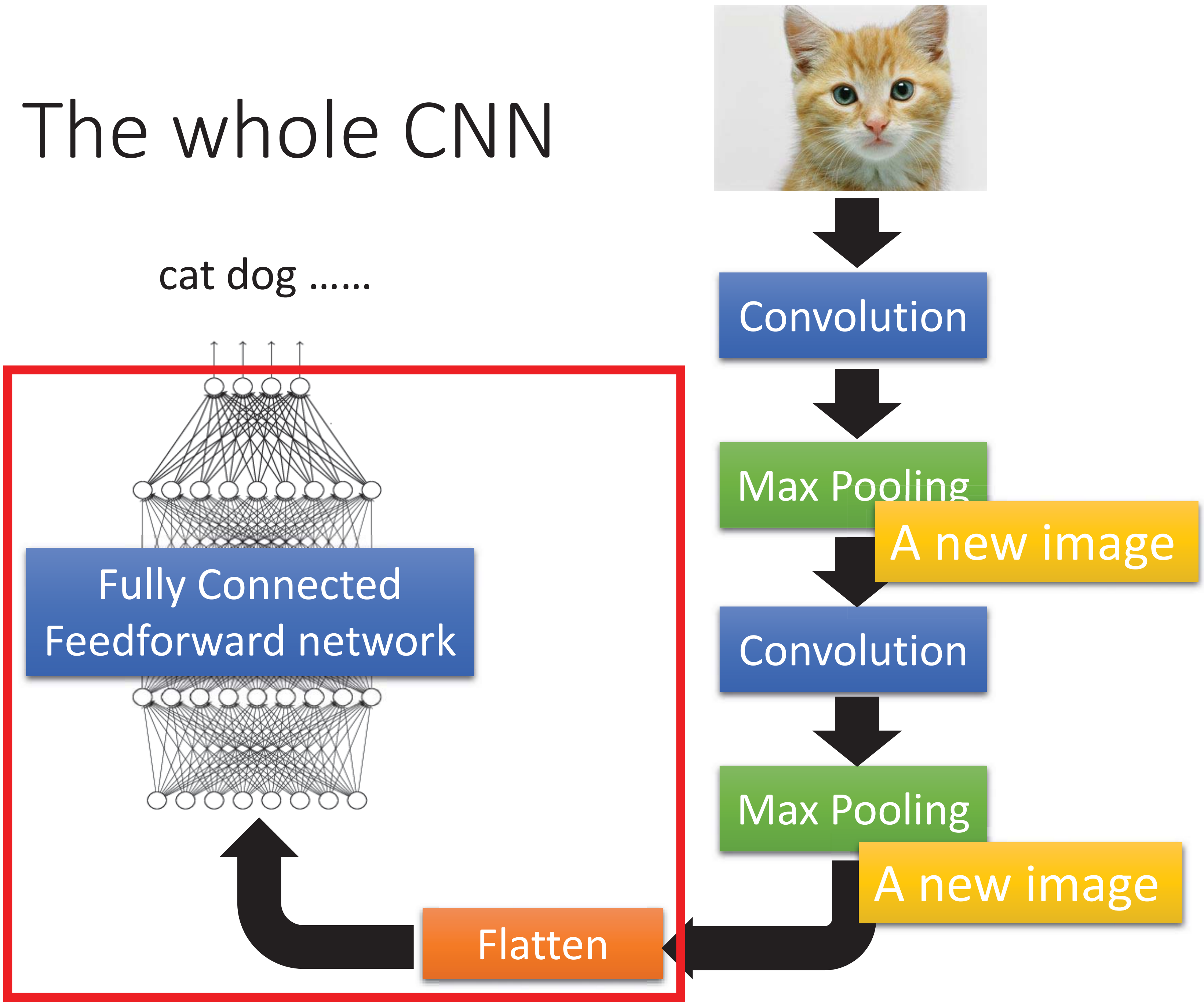




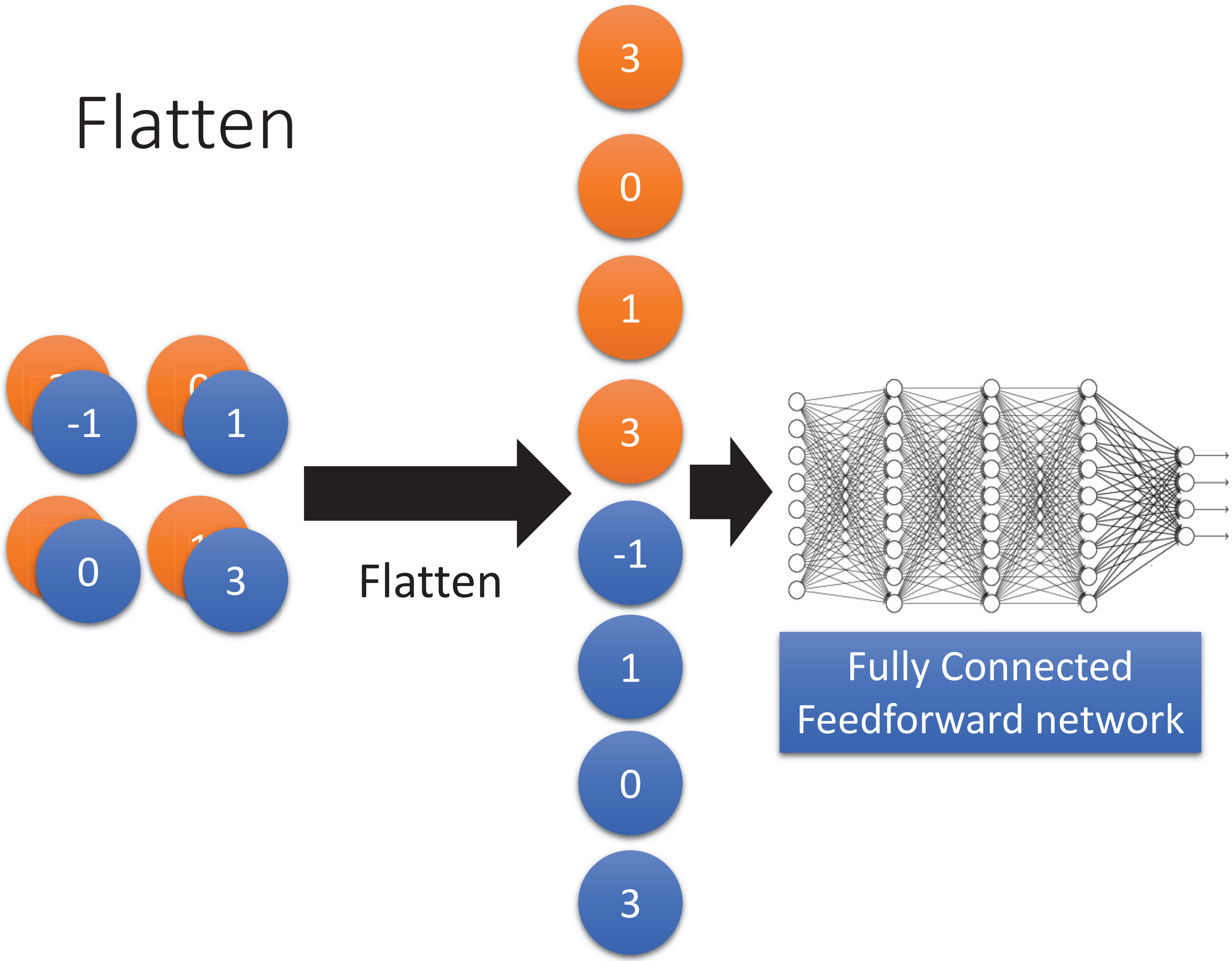
Smaller than the original image

The number of the channel is the number of filters

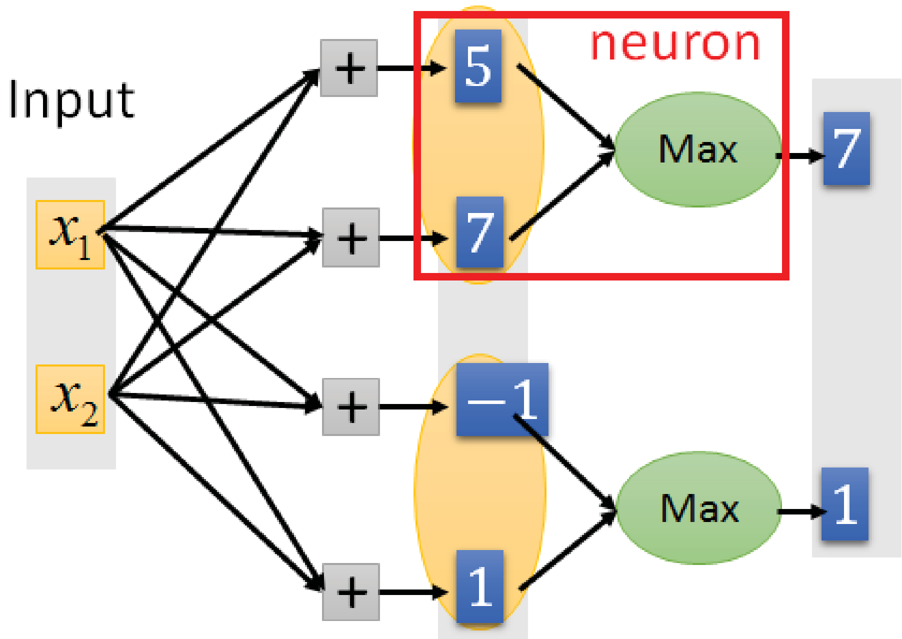
The whole CNN

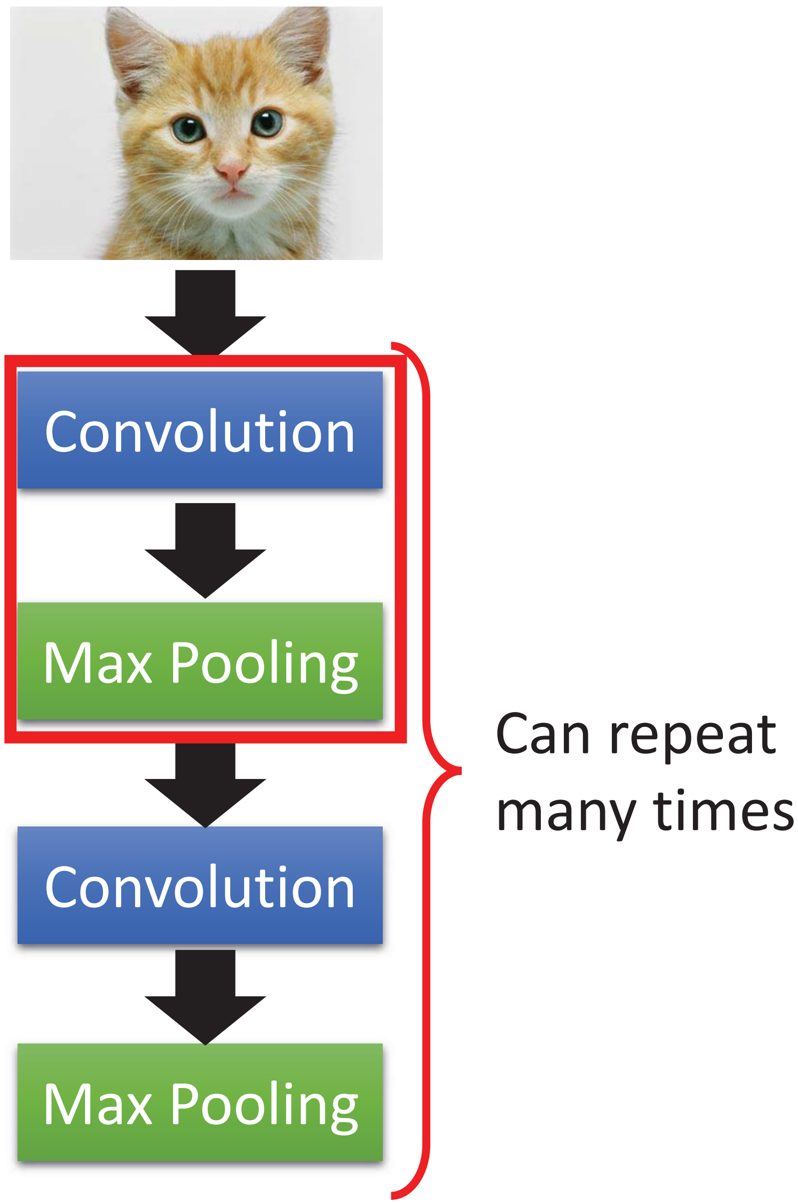


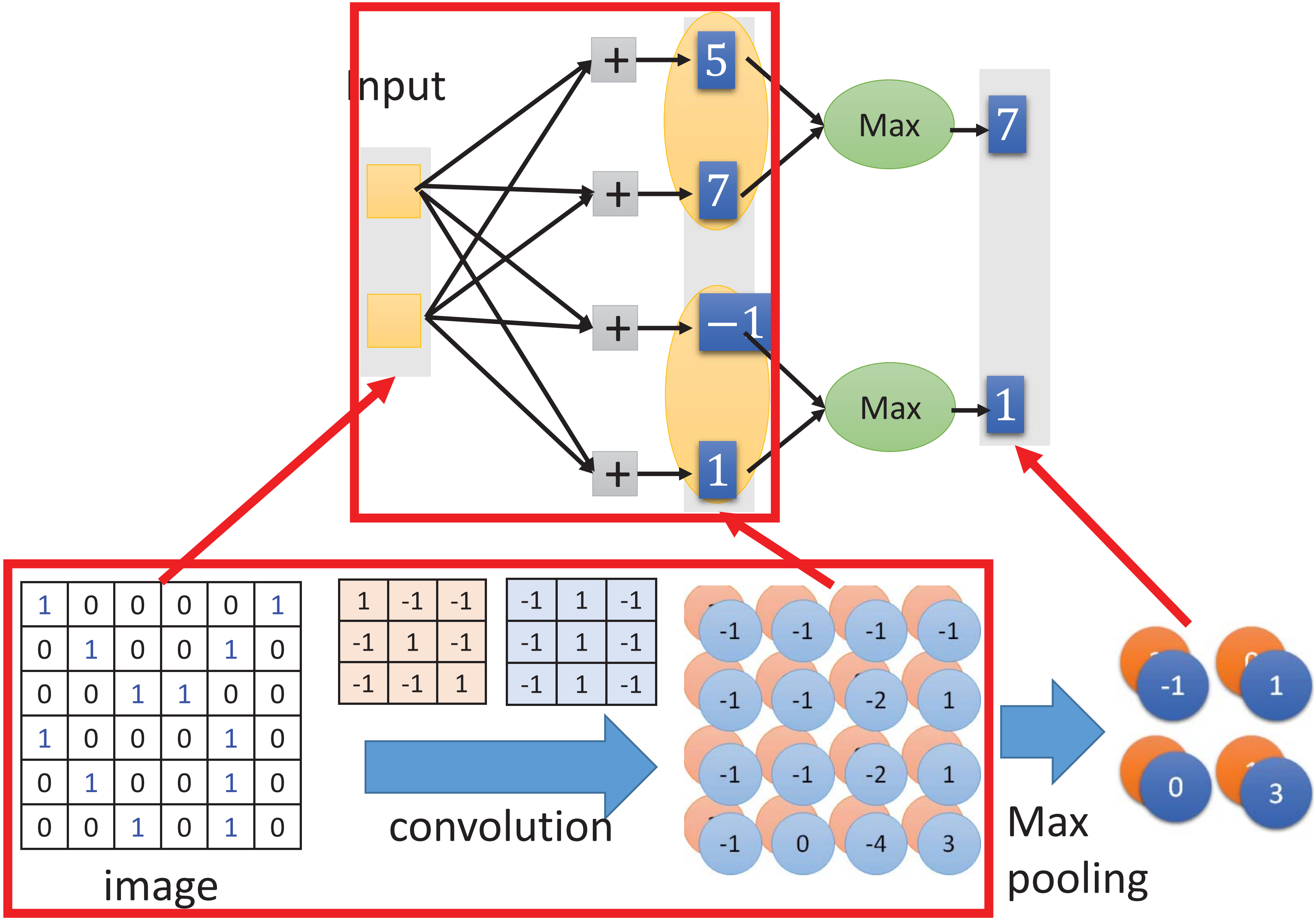
Flatten


-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

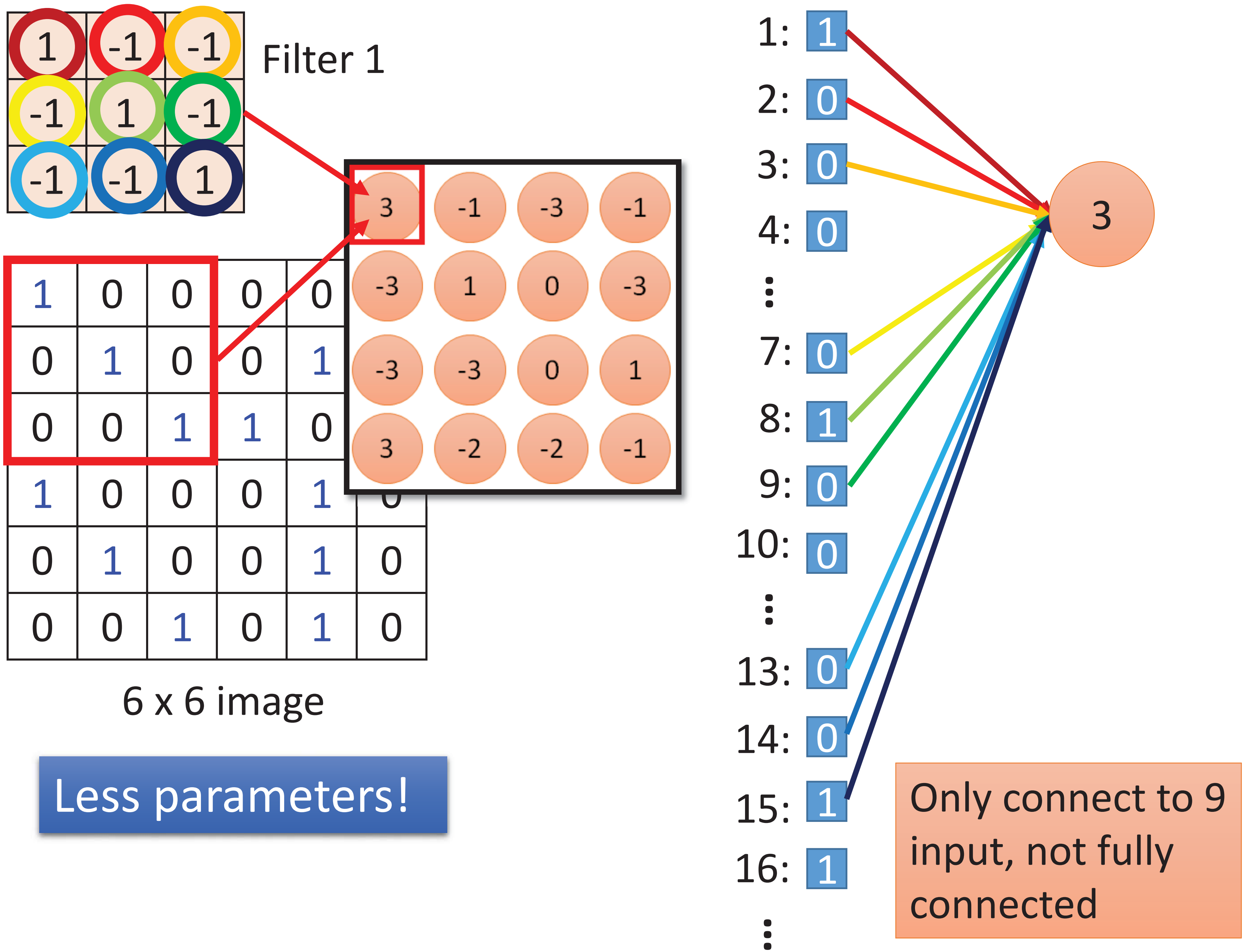
The whole CNN

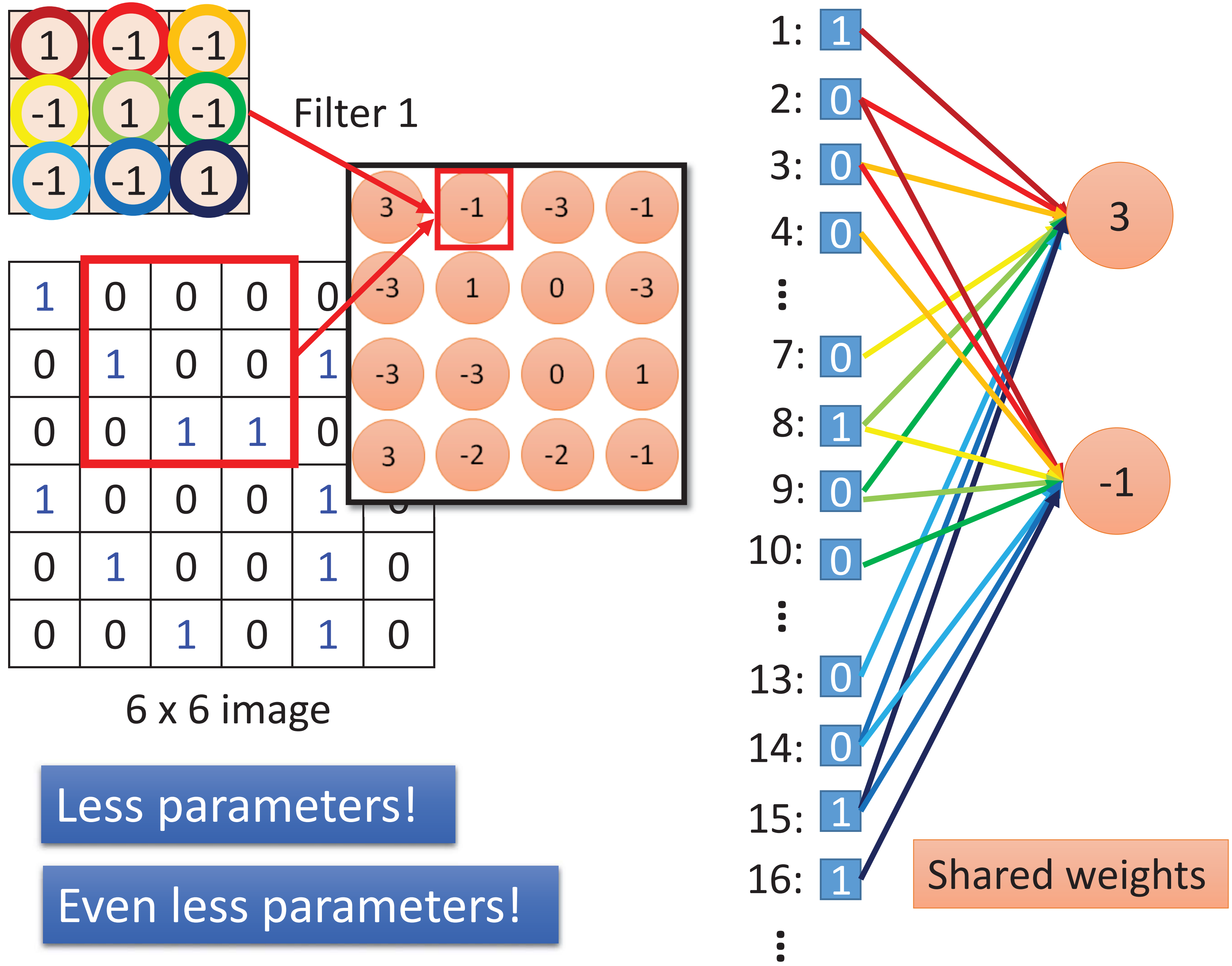




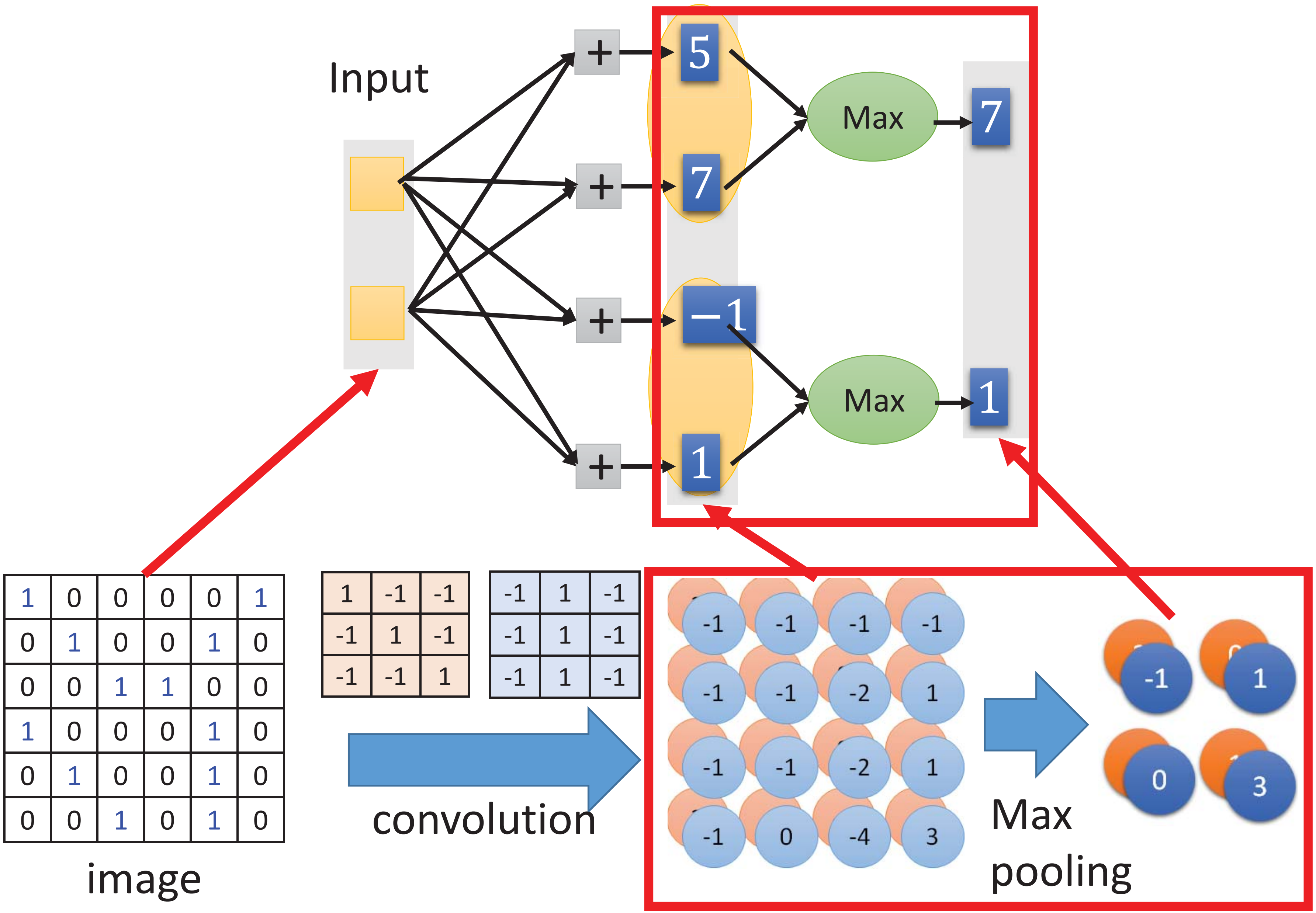


(Ignoring the non-linear activation function after the convolution.)

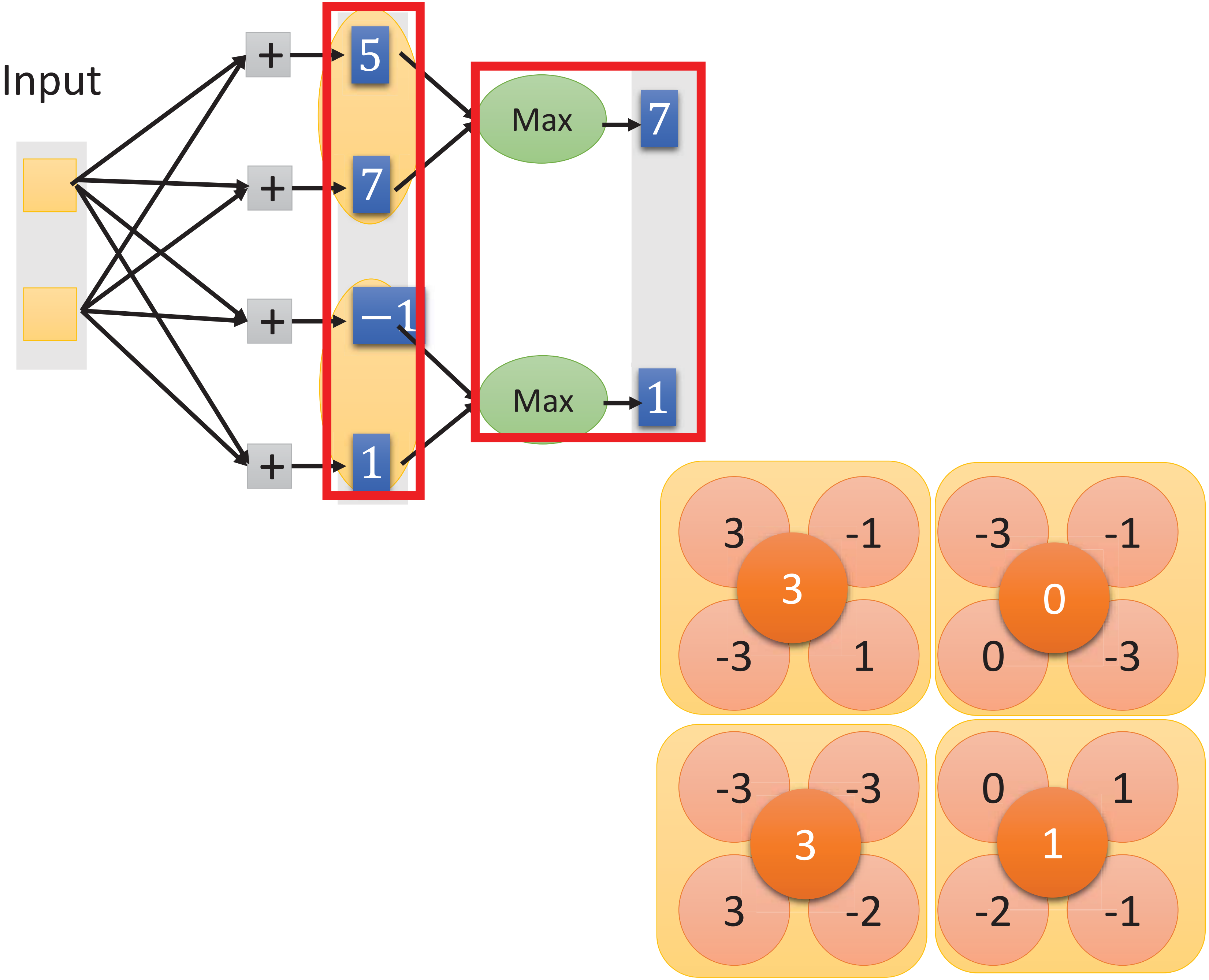

Filter 1
1 -1 -1
-1 1 -1
-1 -1 1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
6 x 6 image
Less parameters!
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
Only connect to 9 input, not fully connected



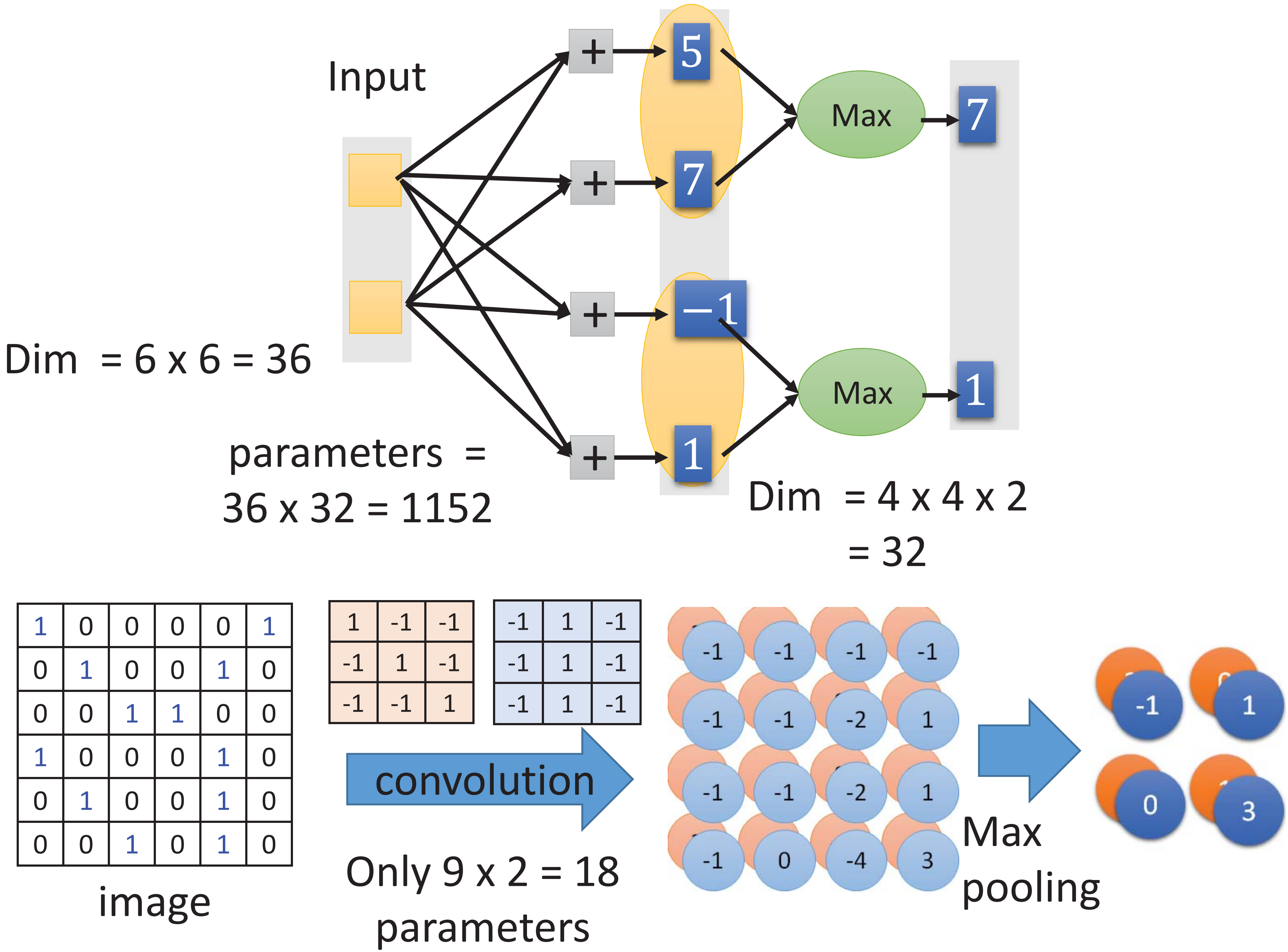
Filter 1
6 x 6 image
Less parameters!
Even less parameters!
Shared weights



(Ignoring the non-linear activation function after the convolution.)



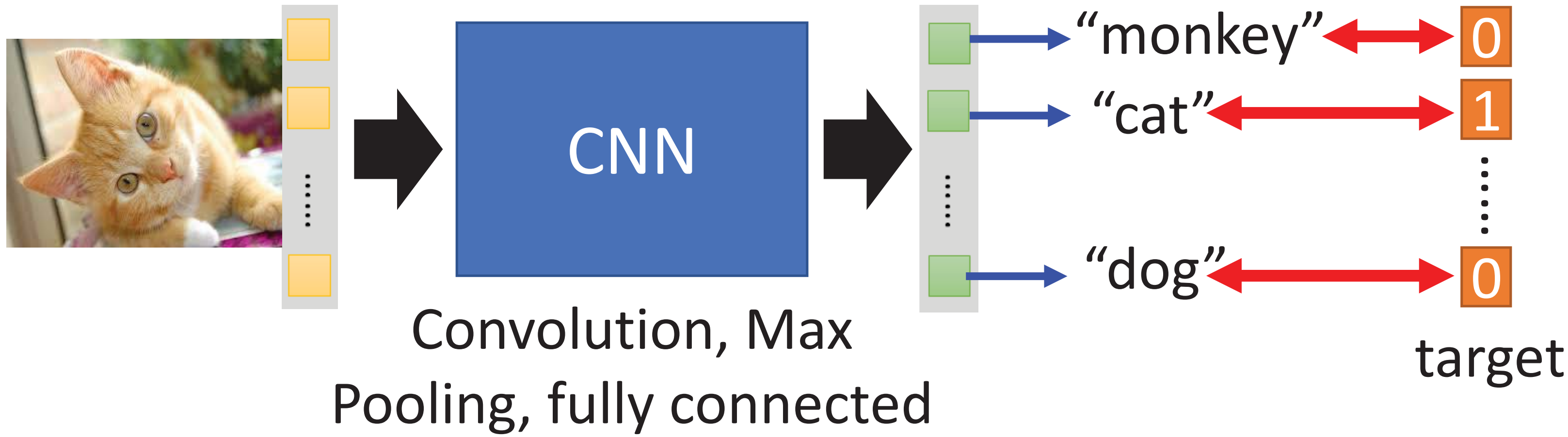
Input
+
+
+
+
5
7
−1
1
Max
Max
7
1
3
-1
3
-3
1
-3
-1
0
0
-3
-3
-3
3
3
-2
0
1
1
-2
-1



Input
Dim = 6 x 6 = 36
parameters =
36 x 32 = 1152
5
7
−1
1
Max
Max
7
1
Dim = 4 x 4 x 2
= 32
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
image
1 -1 -1
-1 1 -1
-1 -1 1
-1 1 -1
-1 1 -1
-1 1 -1
convolution
Only 9 x 2 = 18
parameters
-1 -1 -1 -1
-1 -1 -2 1
-1 -1 -2 1
-1 0 -4 3
Max
pooling
-1
1
0
3

Convolutional Neural Network

Step 1: Convolutional Neural Network → Step 2: goodness of function → Step 3: pick the best function



Learning: Nothing special, just gradient descent ……

Playing Go

19 x 19 matrix (image)

Network

Next move (19 x 19 positions)

19 x 19 vector

Black: 1

white: -1

none: 0

Fully-connected feedword network can be used

But CNN performs much better.

Variants of Neural Networks

Convolutional Neural Network (CNN)

Recurrent Neural Network (RNN)

Neural Network with Memory

Example Application

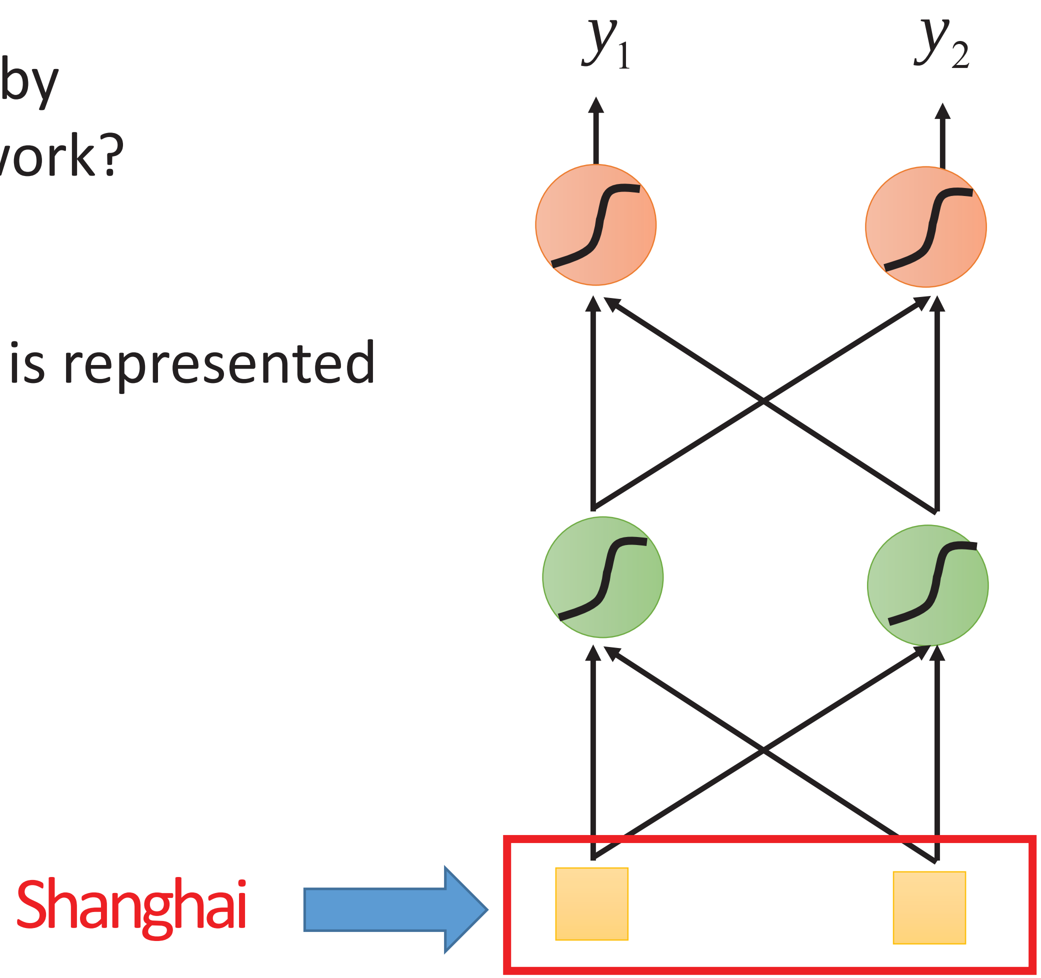
- Slot Filling

Example Application

Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)



1-of-N encoding

How to represent each word as a vector?

1-of-N Encoding lexicon = {apple, bag, cat, dog, elephant}

The vector is lexicon size.

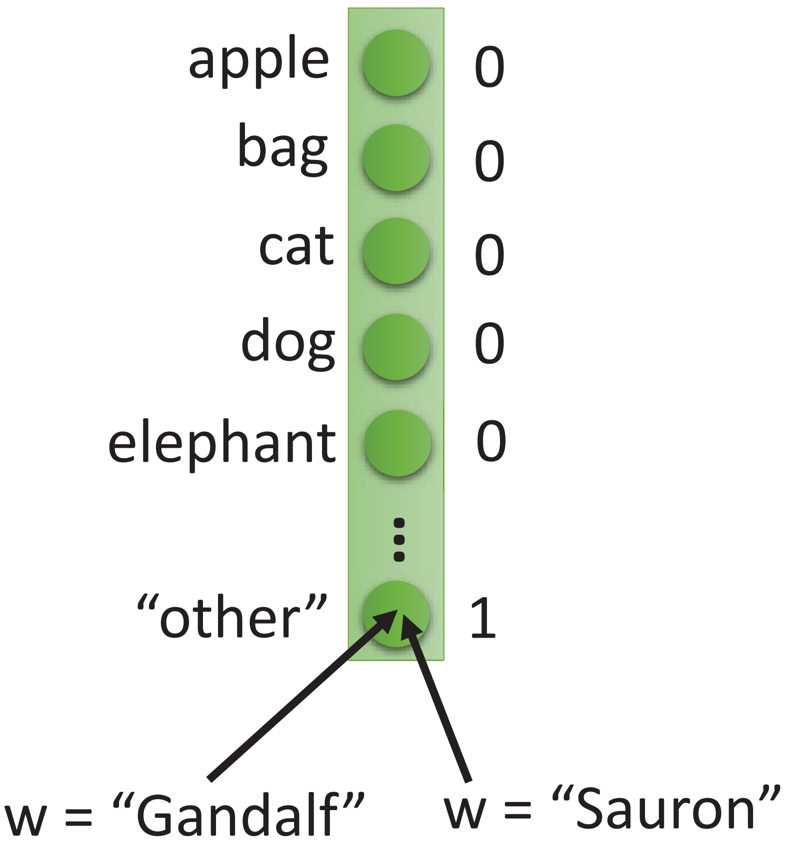
Each dimension corresponds to a word in the lexicon

The dimension for the word is 1, and others are 0

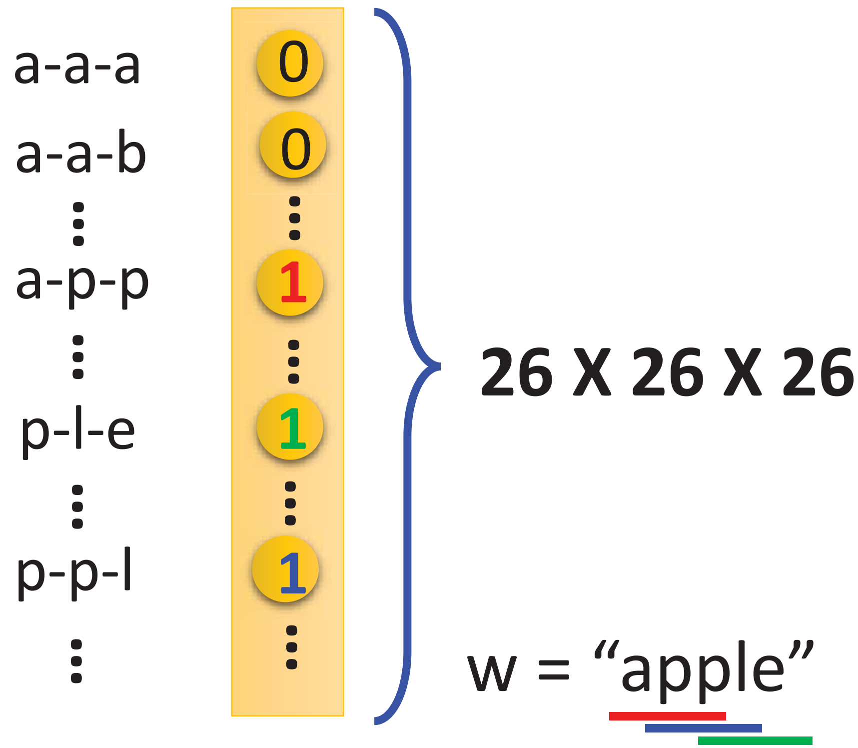
apple = [1 0 0 0 0]
bag = [0 1 0 0 0]
cat = [0 0 1 0 0]
dog = [0 0 0 1 0]
elephant = [0 0 0 0 1]

Beyond 1-of-N encoding

Dimension for "Other"



Word hashing



Example Application

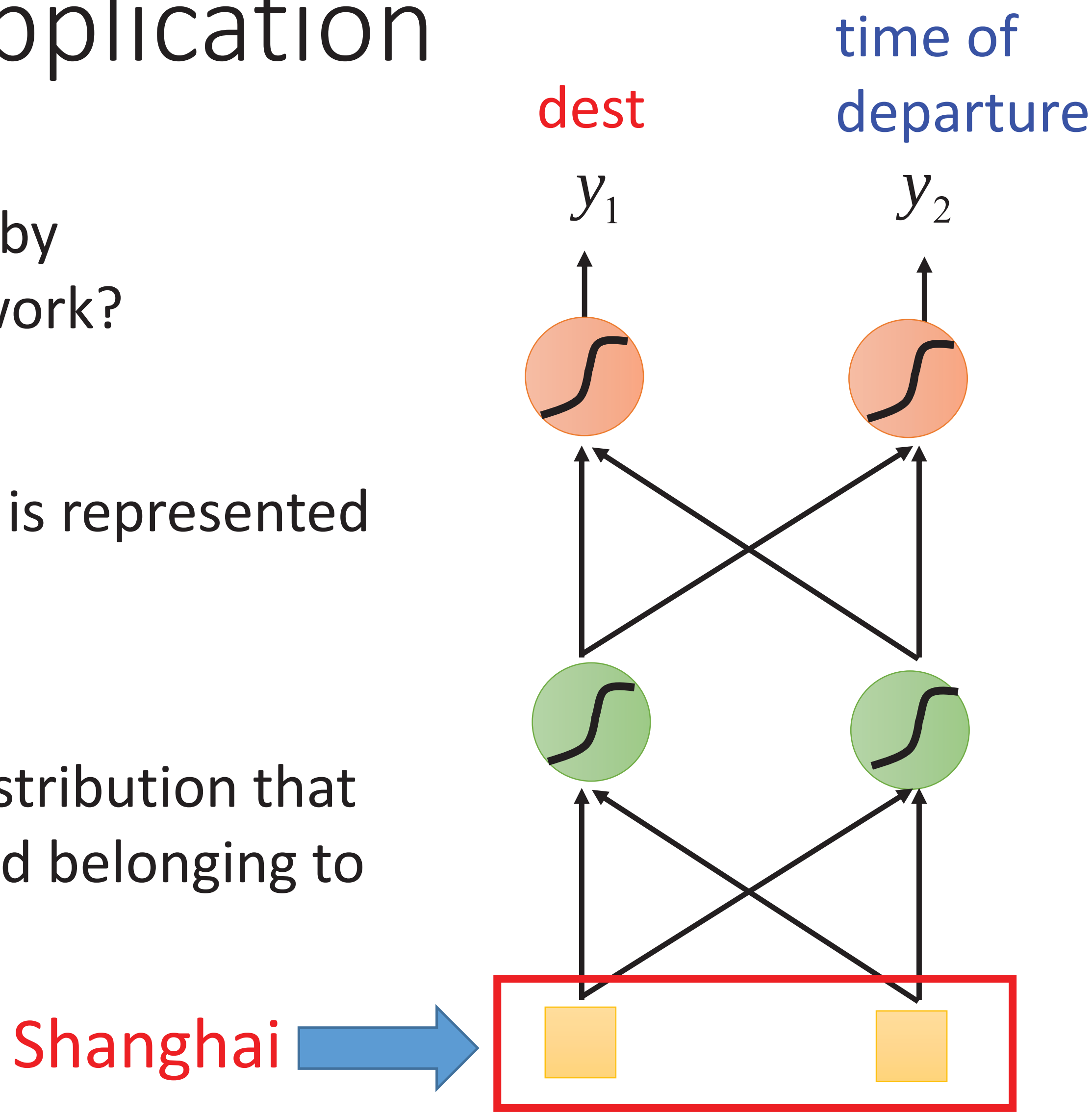
Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)

Output:

Probability distribution that the input word belonging to the slots



Example Application

arrive Shanghai on November 2nd

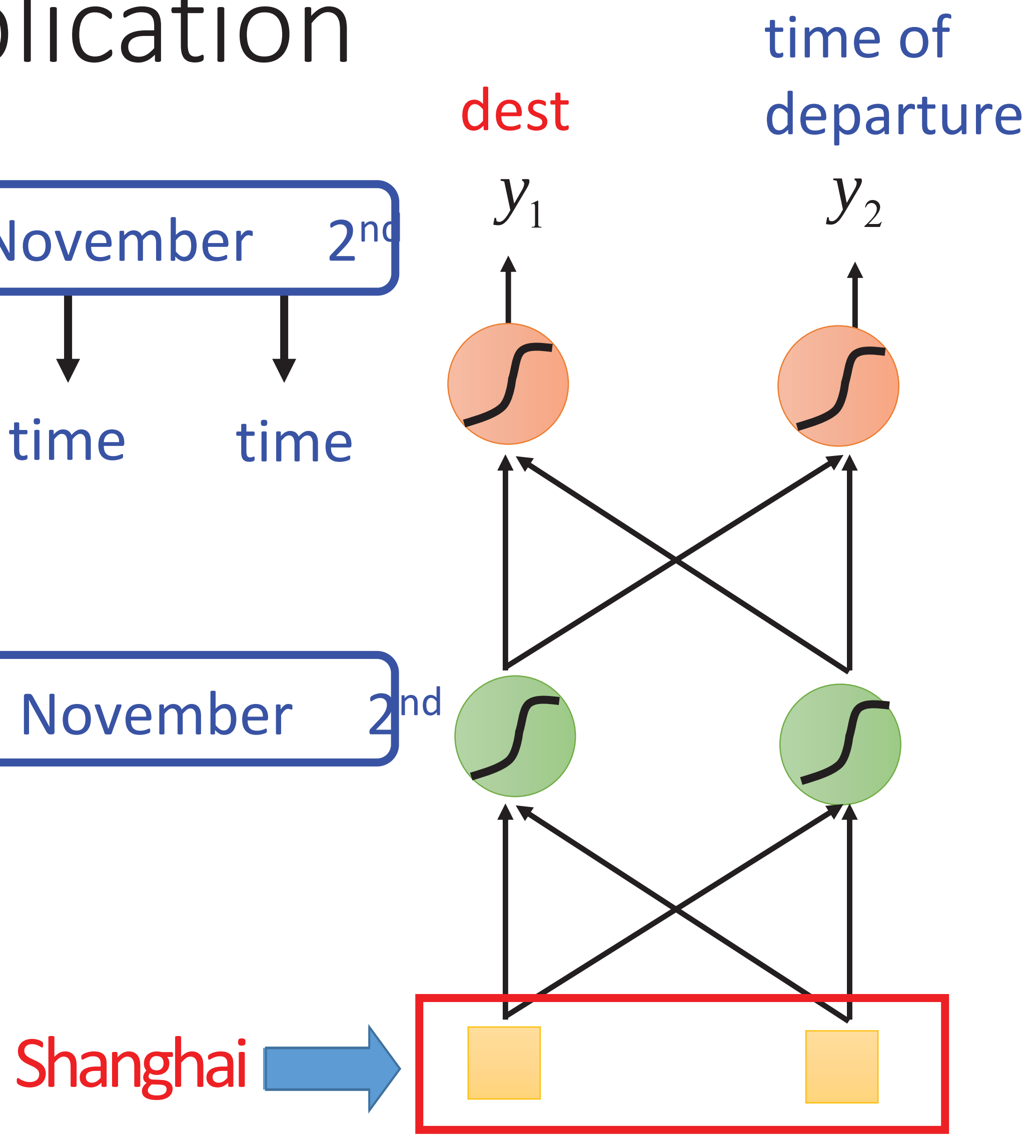
other | dest | other | time | time

Problem?

leave Shanghai on November 2nd

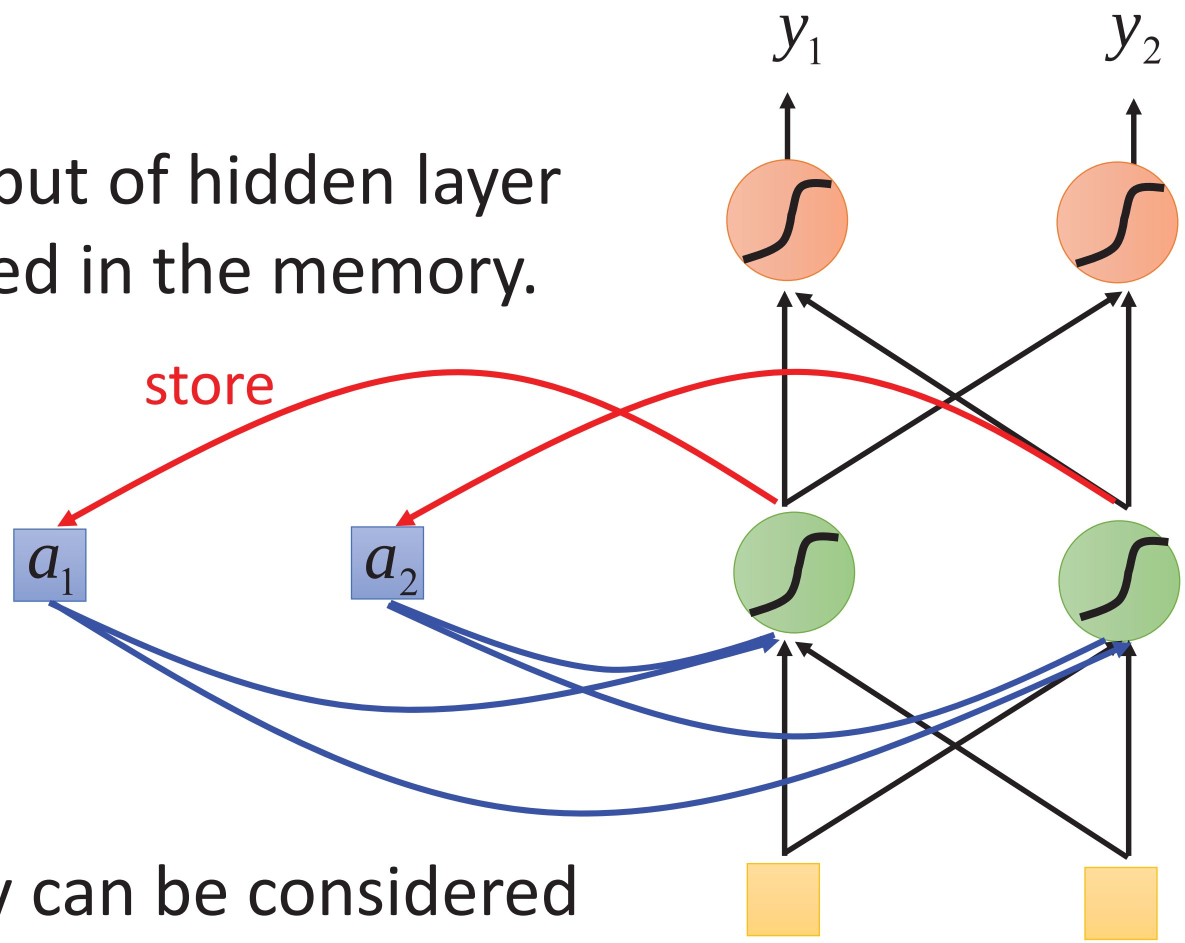
place of departure

Neural network needs memory!



Recurrent Neural Network (RNN)

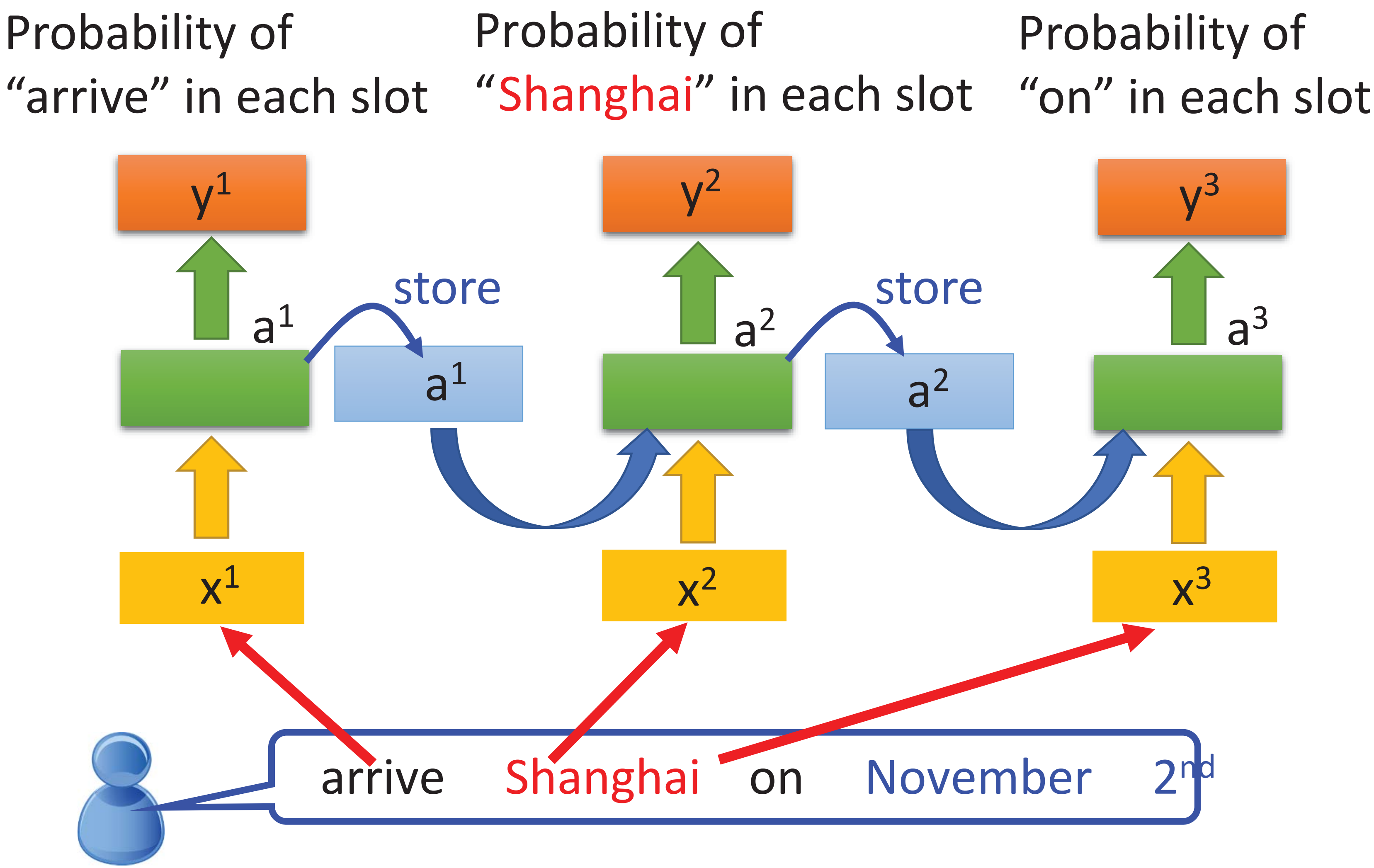
The output of hidden layer are stored in the memory.

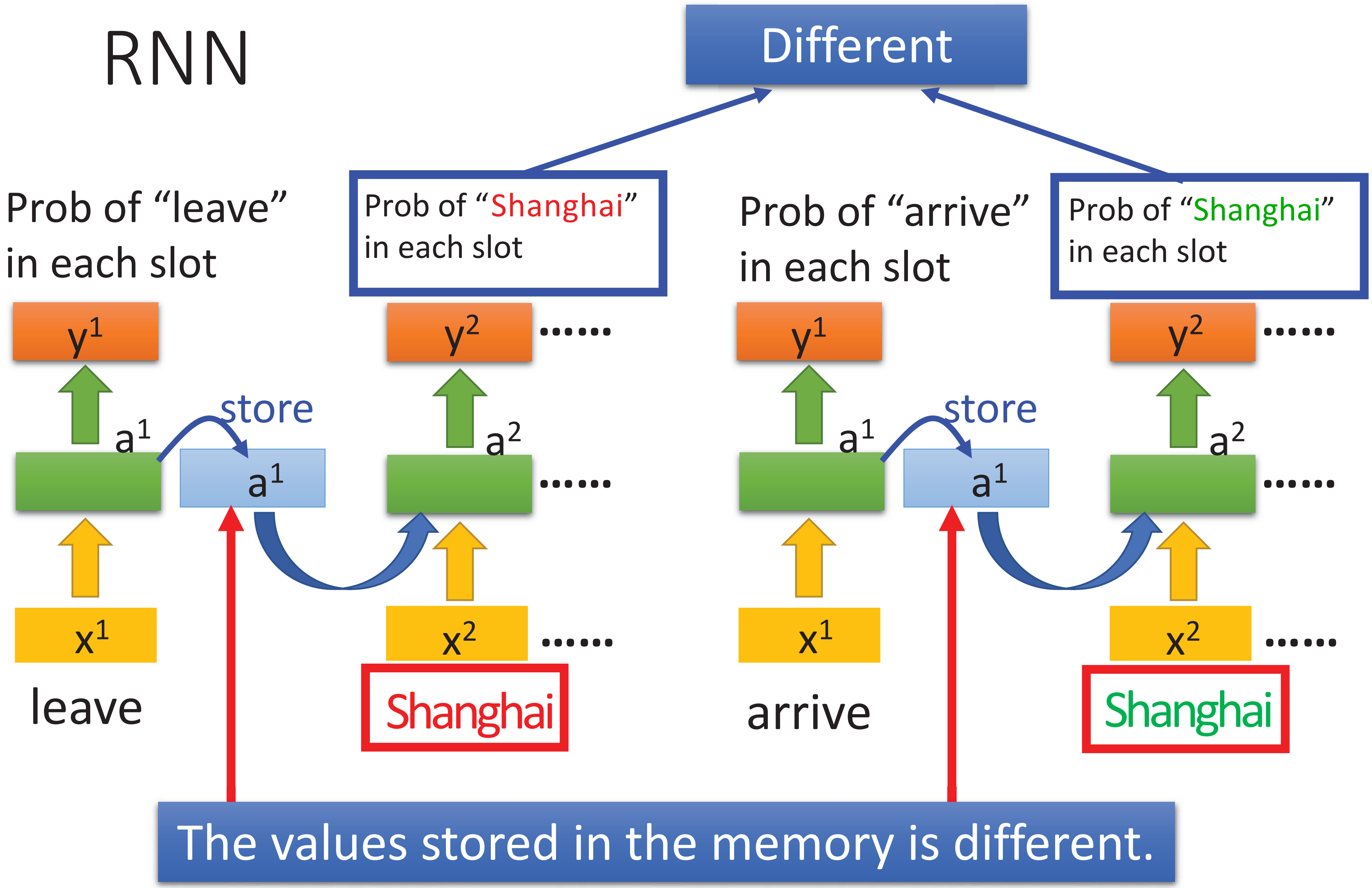


Memory can be considered as another input.

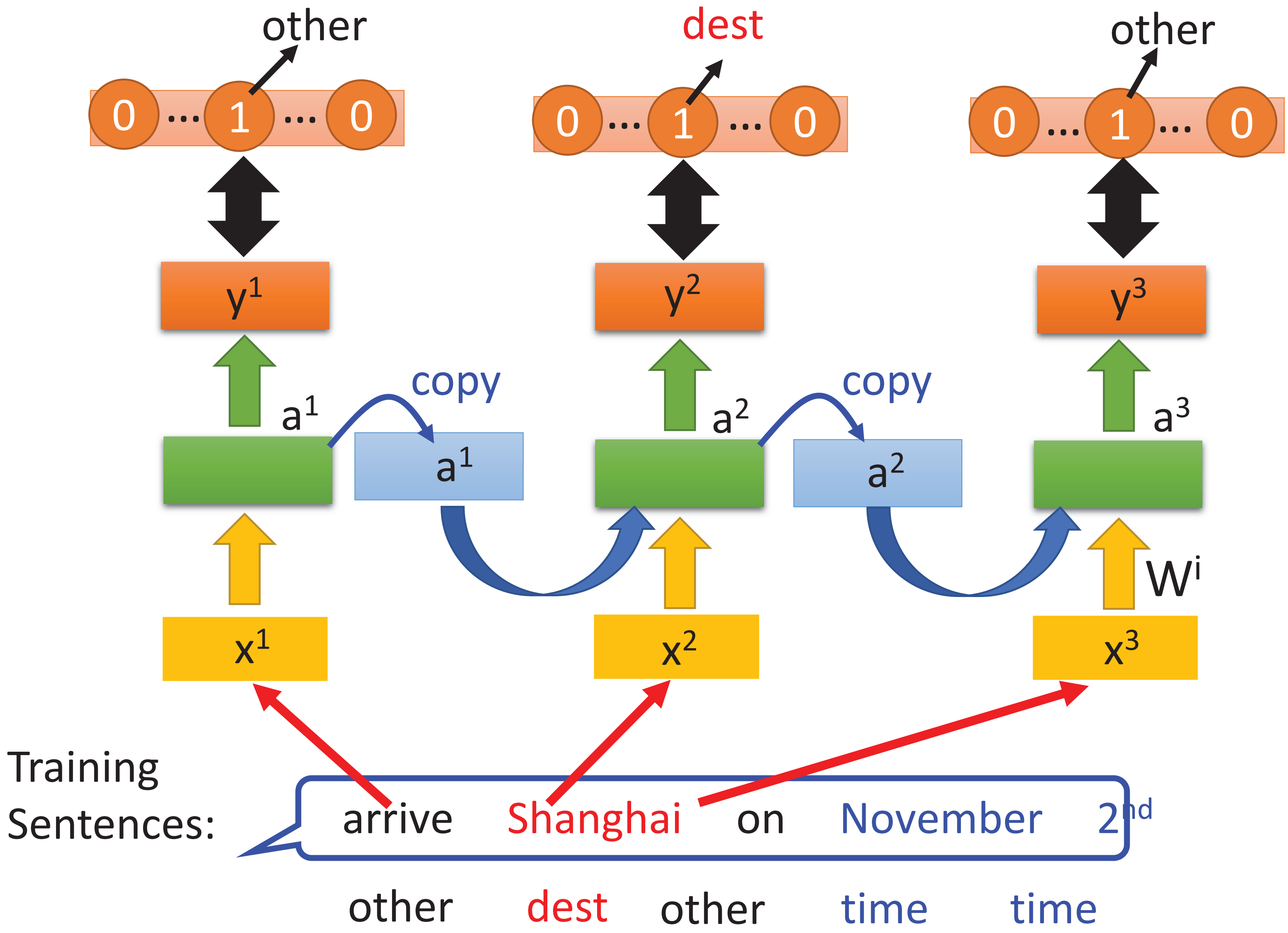
RNN

The same network is used again and again.



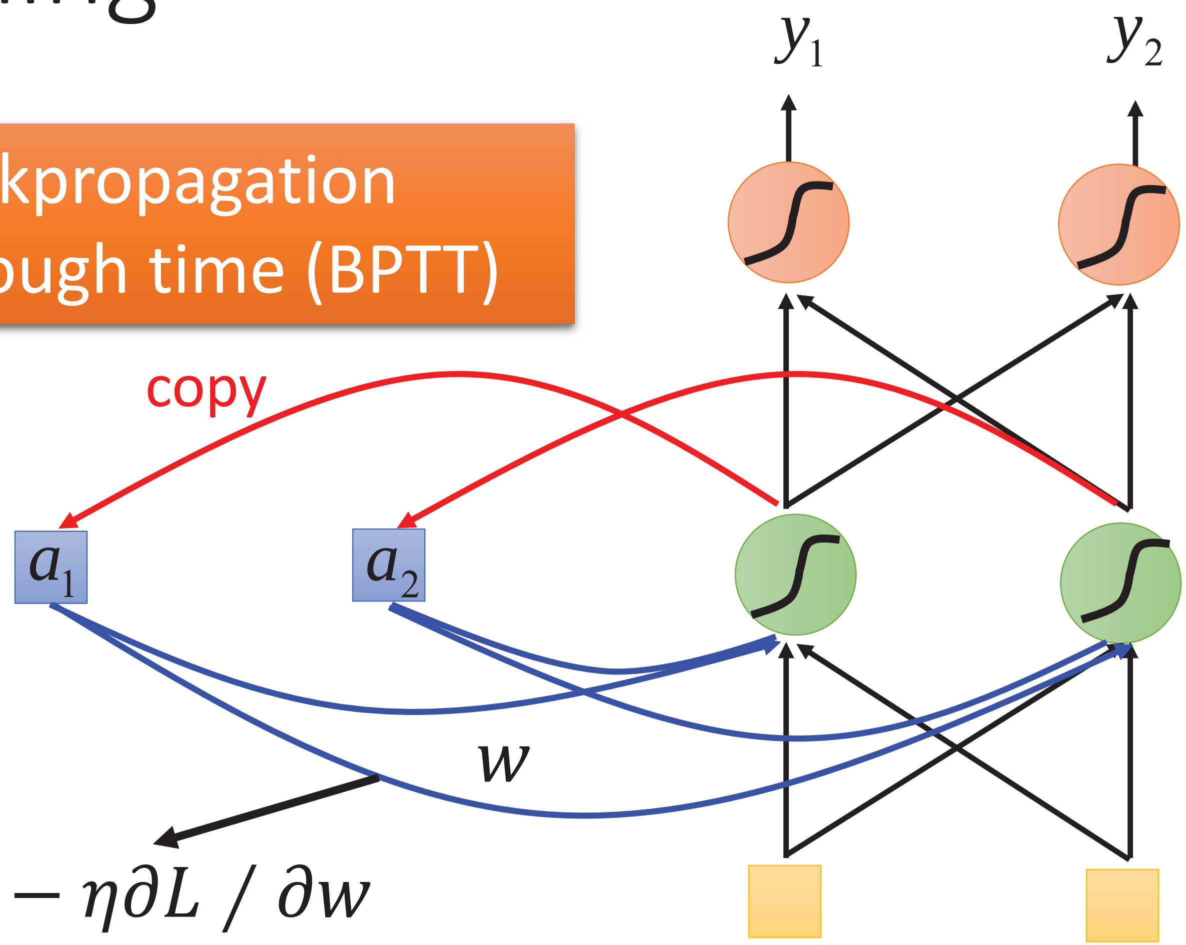

RNN
Different
Prob of "leave" in each slot
Prob of "Shanghai" in each slot
Prob of "arrive" in each slot
Prob of "Shanghai" in each slot
y^1
y^2
......
store
a^1
a^2
x^1
x^2
leave
Shanghai
arrive
Shanghai
The values stored in the memory is different.

Learning Target



Learning

Backpropagation through time (BPTT)



$$w \leftarrow w - \eta \partial L / \partial w$$

RNN Learning is very difficult in practice.

vanishing/exploding gradient problem (1)

- Similar but simpler RNN formulation:

$$\begin{aligned} h_t &= Wf(h_{t-1}) + W^{(hx)} x_{[t]} \\ \hat{y}_t &= W^{(S)} f(h_t) \end{aligned}$$

- Total error is the sum of each error at time steps t

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$

- Hardcore chain rule application:

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

vanishing/exploding gradient problem (2)

- Similar to backprop but less efficient formulation
- Useful for analysis, we'll look at:

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \boxed{\frac{\partial h_t}{\partial h_k}} \frac{\partial h_k}{\partial W}$$

- Remember: $h_t \quad = \quad W f(h_{t-1}) + W^{(hx)} x_{[t]}$
- More chain rule, remember:

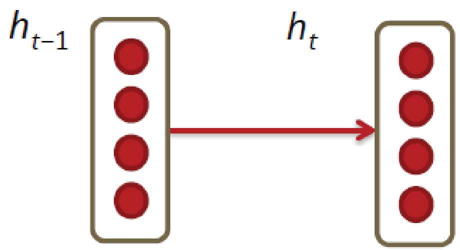
$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$$

- Each partial is a Jacobian:

$$\frac{d\mathbf{f}}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

vanishing/exploding gradient problem (3)

- From previous slide: $\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$



- Remember: $h_t \quad = \quad Wf(h_{t-1}) + W^{(hx)} x_{[t]}$

- To compute Jacobian, derive each element of matrix: $\frac{\partial h_{j,m}}{\partial h_{j-1,n}}$

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k+1}^{t} W^T \mathrm{diag}[f'(h_{j-1})]$$

- Where: $\mathrm{diag}(z) = \begin{pmatrix} z_1 & & & 0 \\ & z_2 & & \\ & & \ddots & \\ 0 & & z_{n-1} & \\ & & & z_n \end{pmatrix}$

Check at home
that you understand
the diag matrix
formulation

vanishing/exploding gradient problem (4)

- Analyzing the norms of the Jacobians, yields:

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \|W^T\| \|\text{diag}[f'(h_{j-1})]\| \leq \beta_W \beta_h$$

- Where we defined β's as upper bounds of the norms
- The gradient is a product of Jacobian matrices, each associated with a step in the forward computation.

$$\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq (\beta_W \beta_h)^{t-k}$$

- This can become very small or very large quickly [Bengio et al 1994], and the locality assumption of gradient descent breaks down. → **Vanishing or exploding gradient**

Solve vanishing/exploding gradient

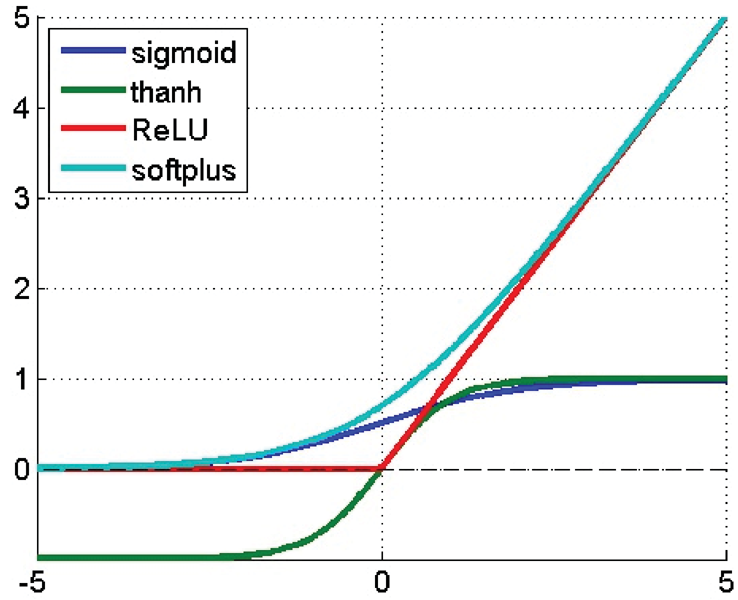
- Clip gradients to a maximum value

$$
\begin{aligned}
&\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta} \\
&\textbf{if} \quad \|\hat{\mathbf{g}}\| \geq threshold \textbf{ then} \\
&\quad \hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}} \\
&\textbf{end if}
\end{aligned}
$$

- Truncated gradient
  - Only use recent information
- Relu or Softplus

$Softplus(x) = log(1 + e^x)$ $f(x) = max(0, x)$

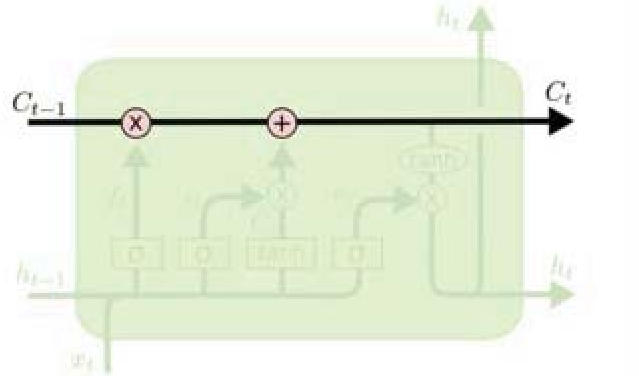
- Gate: GRU or LSTM

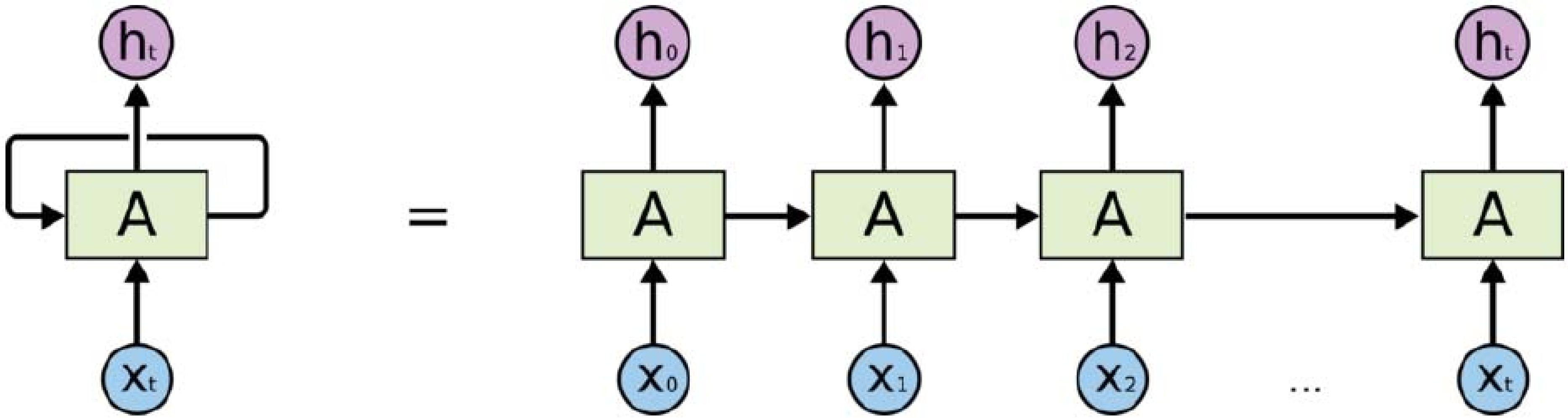


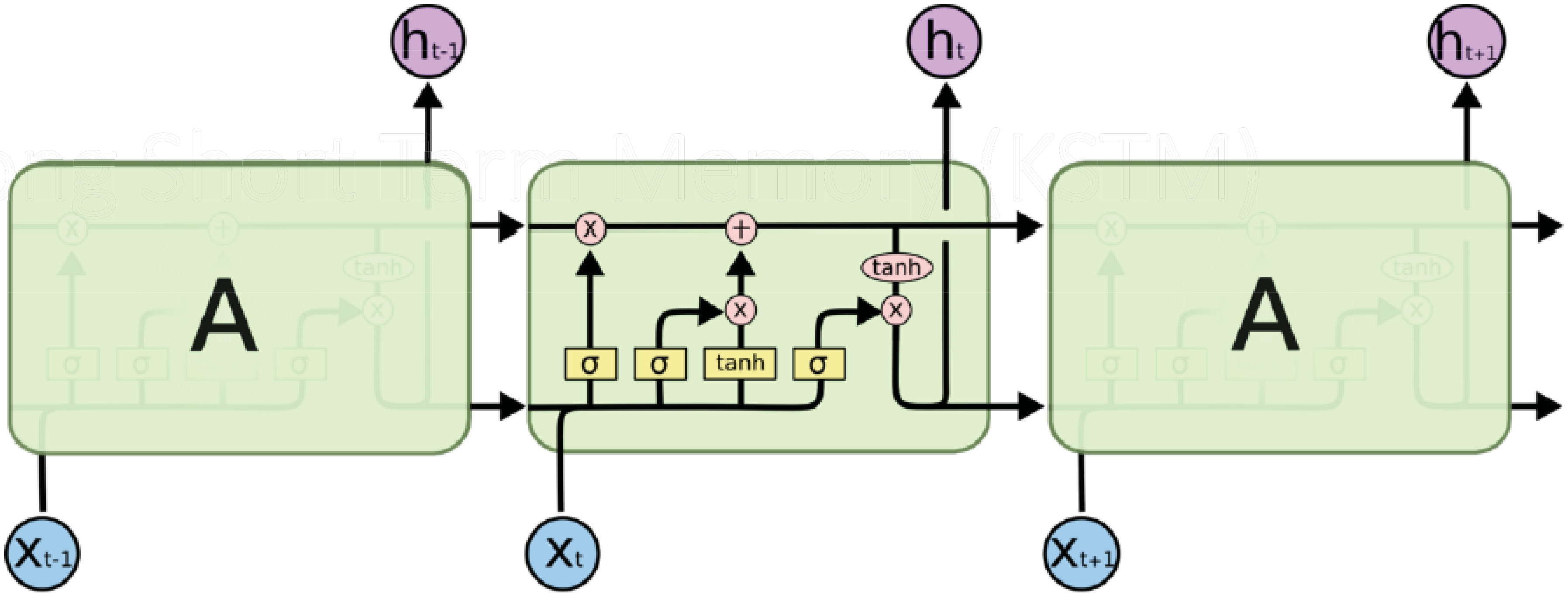
Long Short Term Memory (LSTM)

- Add hidden cell as memory C
- Inertia $\mathbf{c}_t = \mathbf{i}_t \odot \mathbf{u}_t + \mathbf{f}_t \odot \mathbf{c}_{t-1}$
- Avoid gradient vanishing

(gradient derivation)







$$\mathbf{i}_t = \sigma(\mathbf{W}^{(i)}\mathbf{x}_t + \mathbf{U}^{(i)}\mathbf{h}_{t-1} + \mathbf{b}^{(i)})$$

$$\mathbf{o}_t = \sigma(\mathbf{W}^{(o)}\mathbf{x}_t + \mathbf{U}^{(o)}\mathbf{h}_{t-1} + \mathbf{b}^{(o)})$$

$$\frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} = \sigma(\mathbf{W}^{(f)}\mathbf{x}_t + \mathbf{U}^{(f)}\mathbf{h}_{t-1} + \mathbf{b}^{(f)})$$

$$\mathbf{f}_t = \sigma(\mathbf{W}^{(f)}\mathbf{x}_t + \mathbf{U}^{(f)}\mathbf{h}_{t-1} + \mathbf{b}^{(f)})$$

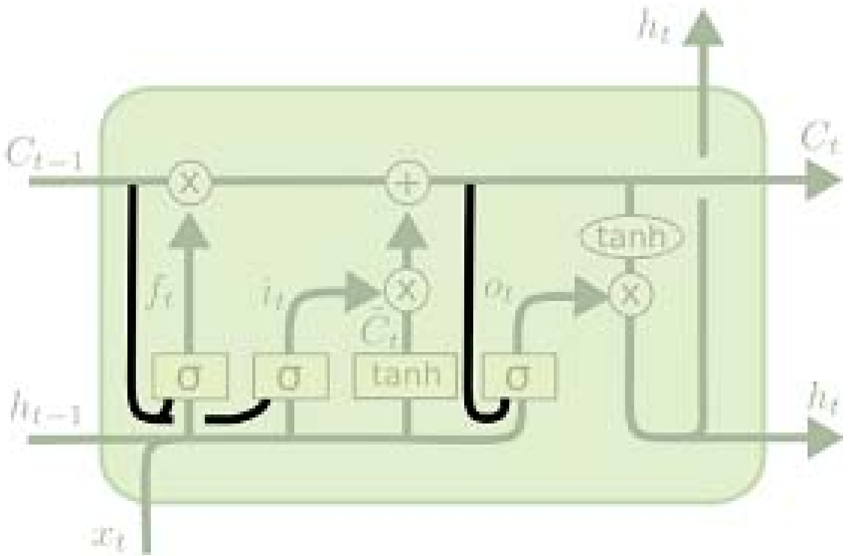
$$\mathbf{u}_t = tanh(\mathbf{W}^{(u)}\mathbf{x}_t + \mathbf{U}^{(u)}\mathbf{h}_{t-1} + \mathbf{b}^{(u)})$$

$$\mathbf{c}_t = \mathbf{i}_t \odot \mathbf{u}_t + \mathbf{f}_t \odot \mathbf{c}_{t-1}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c})$$

peephole connections

- Gates are related to memories



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$

$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

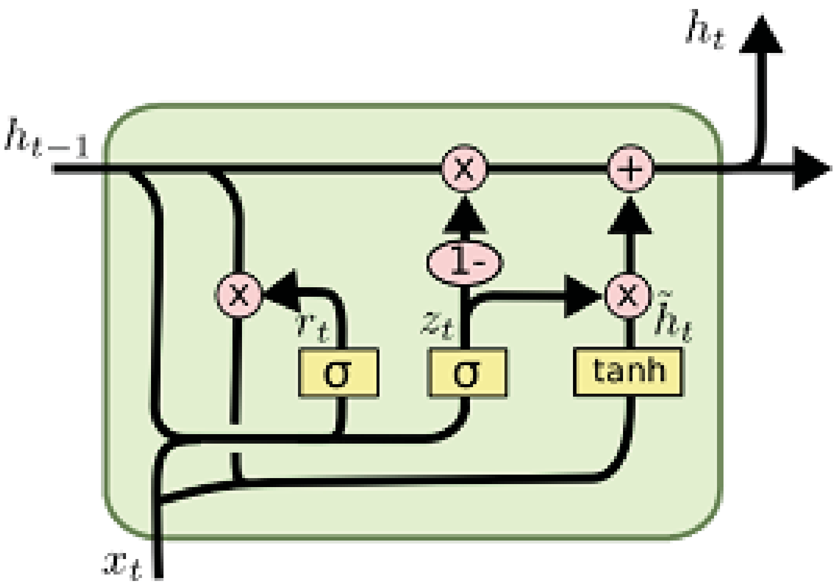
$$\mathbf{u}_t = tanh(\mathbf{W}^{(u)}\mathbf{x}_t + \mathbf{U}^{(u)}\mathbf{h}_{t-1} + \mathbf{b}^{(u)})$$

$$\mathbf{c}_t = \mathbf{i}_t \odot \mathbf{u}_t + \mathbf{f}_t \odot \mathbf{c}_{t-1}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c})$$

Gated Recurrent Unit (GRU)

- Combine forget gate and input gate as update gate z_t



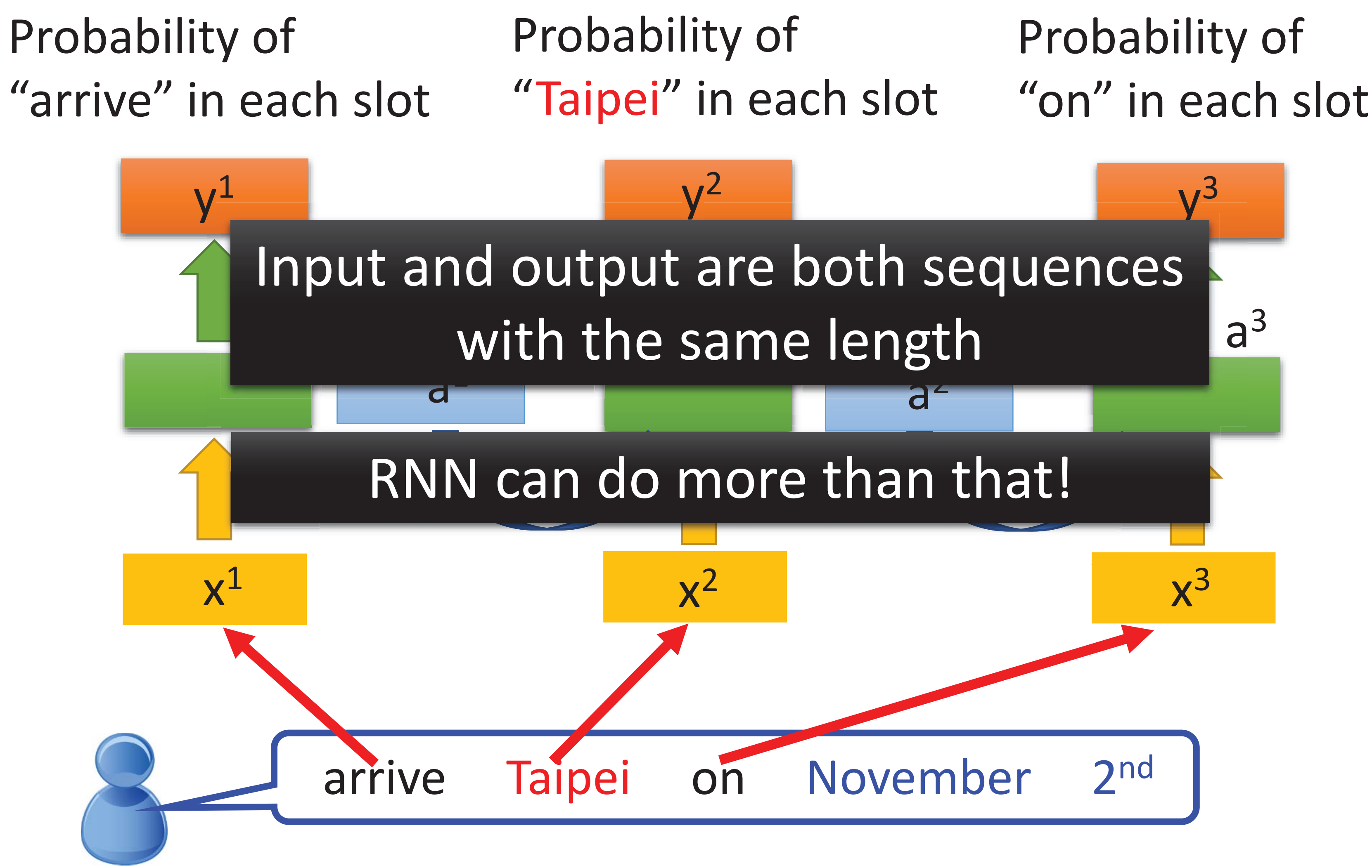
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

More Applications ……

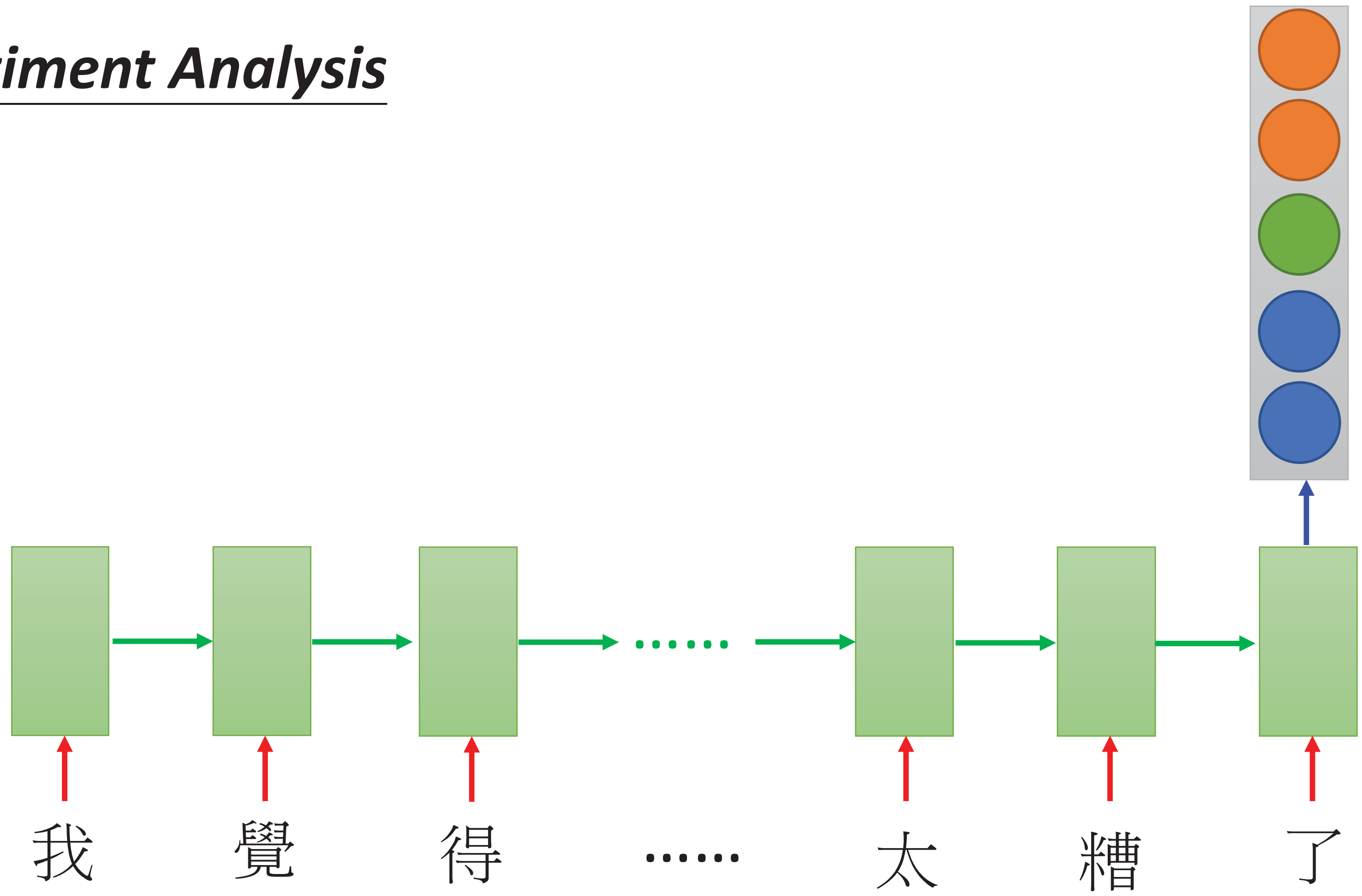


Many to one

Keras Example:
https://github.com/fchollet/keras/blob/master/examples/imdb_lstm.py

- Input is a vector sequence, but output is only one vector

Sentiment Analysis

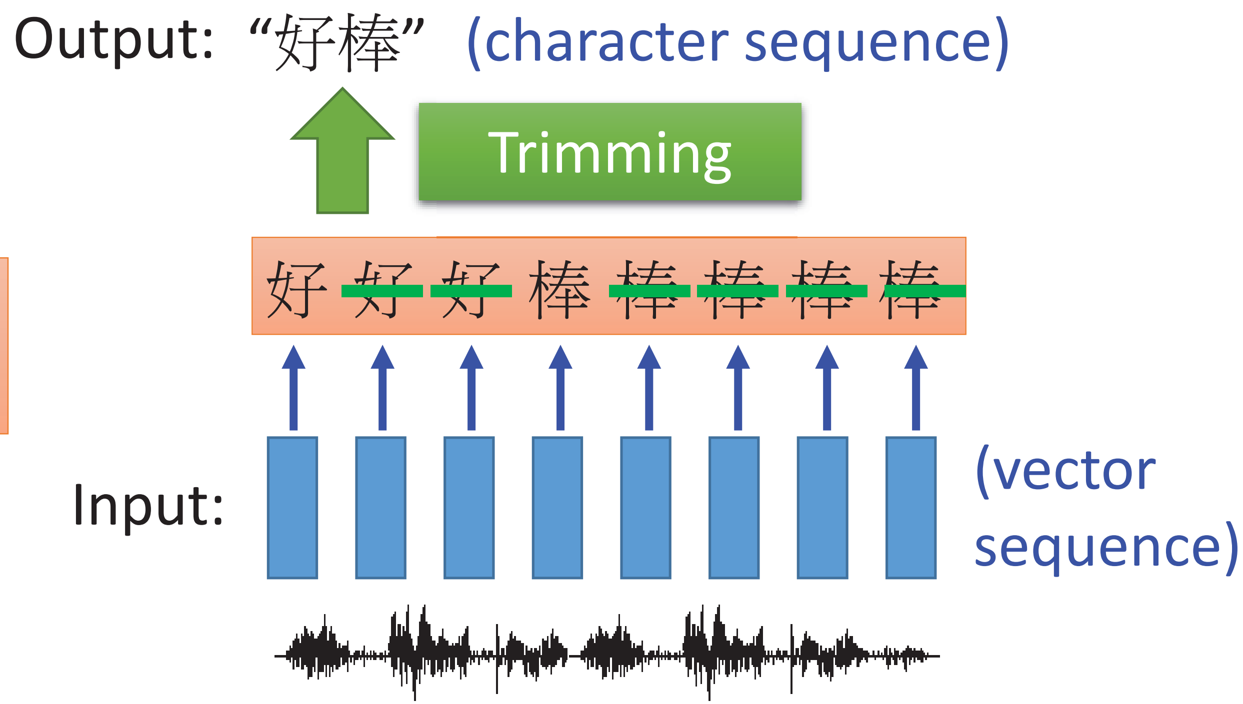


Many to Many (Output is shorter)

- Both input and output are both sequences, ***but the output is shorter.***
  - E.g. ***Speech Recognition***

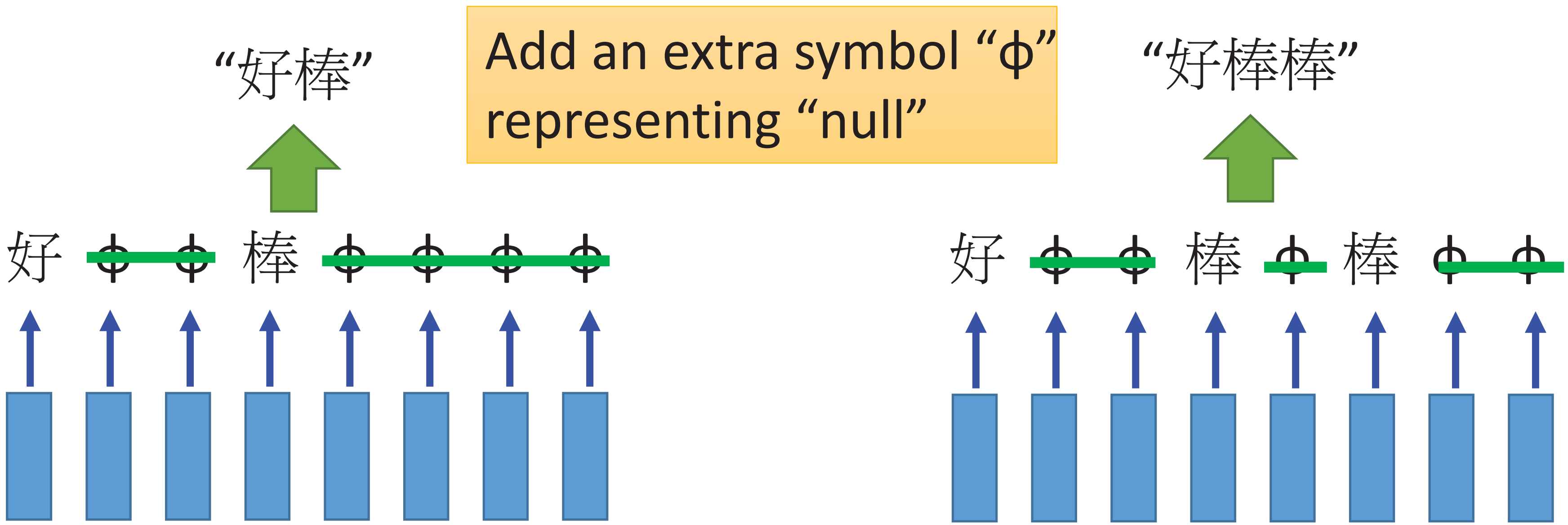
Problem?

Why can’t it be “好棒棒”



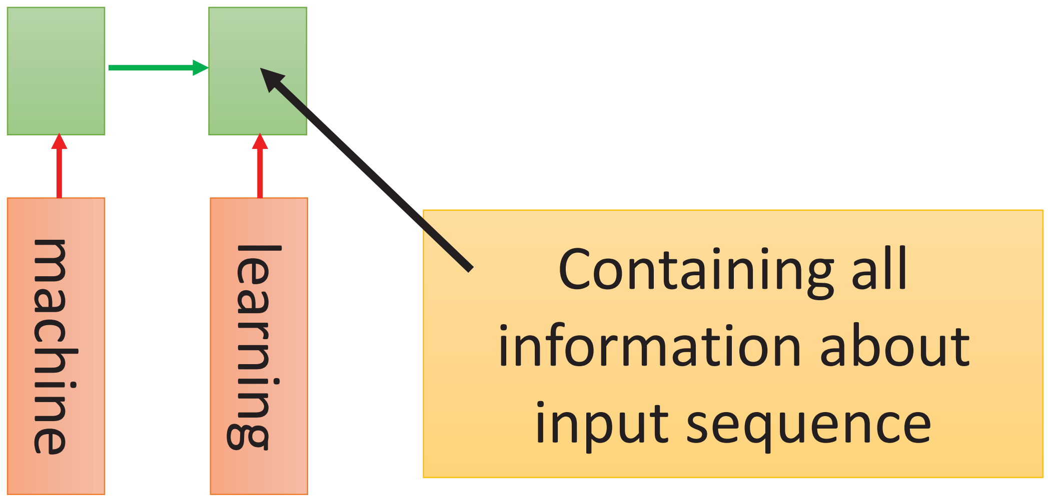
Many to Many (Output is shorter)

- Both input and output are both sequences, ***but the output is shorter.***
- Connectionist Temporal Classification (CTC) [Alex Graves, ICML'06][Alex Graves, ICML'14][Haşim Sak, Interspeech'15][Jie Li, Interspeech'15][Andrew Senior, ASRU'15]



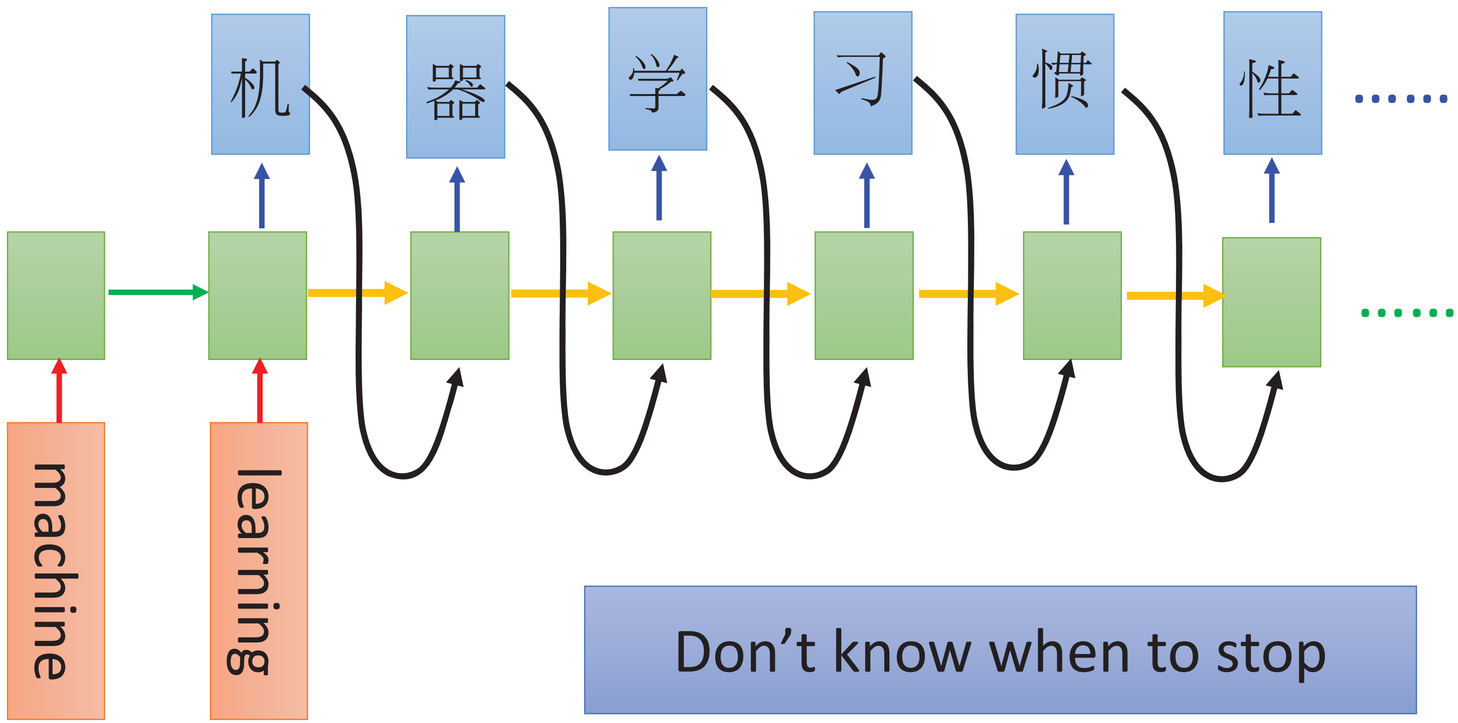
Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→機器學習)



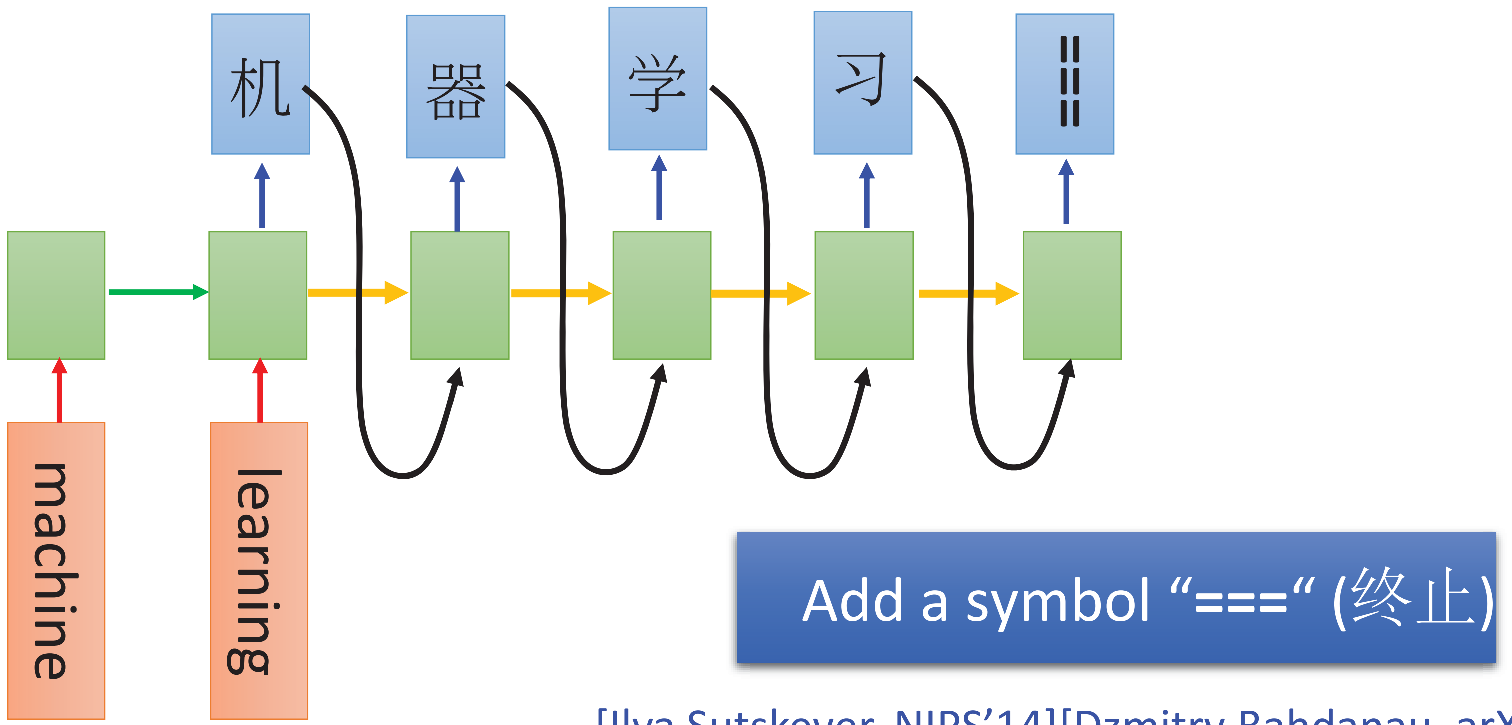
Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→机器学习)



Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→机器学习)

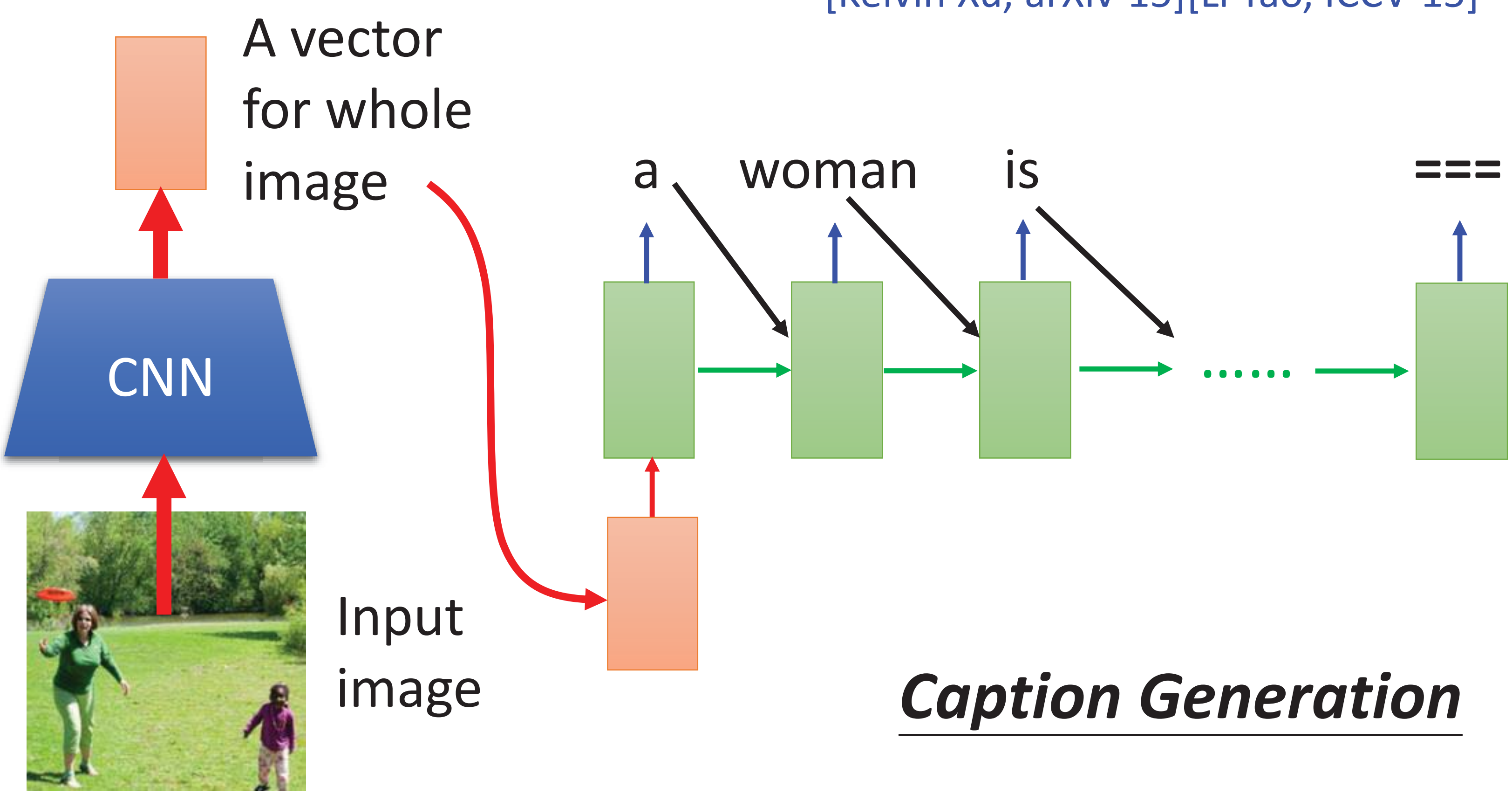


[Ilya Sutskever, NIPS'14][Dzmitry Bahdanau, arXiv'15]

One to Many

- Input an image, but output a sequence of words

[Kelvin Xu, arXiv'15][Li Yao, ICCV'15]



Caption Generation