# DEEP LEARNING CNN & RNN

Lecture-6 , Day-3

STTP on “Deep Learning, Computer Vision and Speech Processing”

By: Suprava, Patnaik, Professor, ExTC, XIE, Mumbai

# PART I:
# DEEPLEARNING :

---

## WEIGHT SHARING(CNN)

# Why Convolution?

- Translation invariance

- Object recognition:
  - Feed-forward NN has different weights everywhere and for every input location.
  - If we train it on left image, it wouldn’t recognize the right image

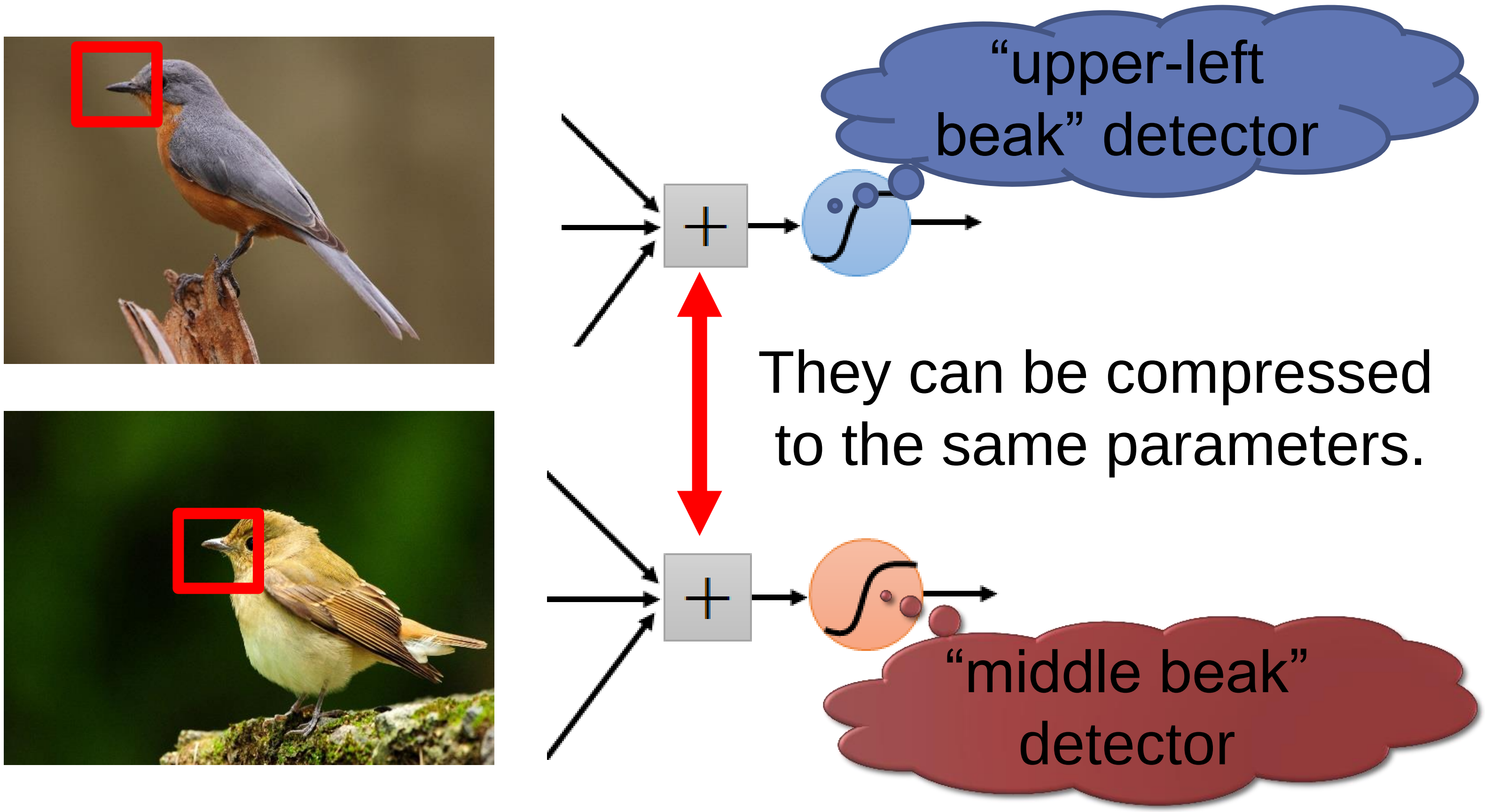
# Consider learning an image:

- Some patterns are much smaller than the whole image

Can represent a small region with fewer parameters

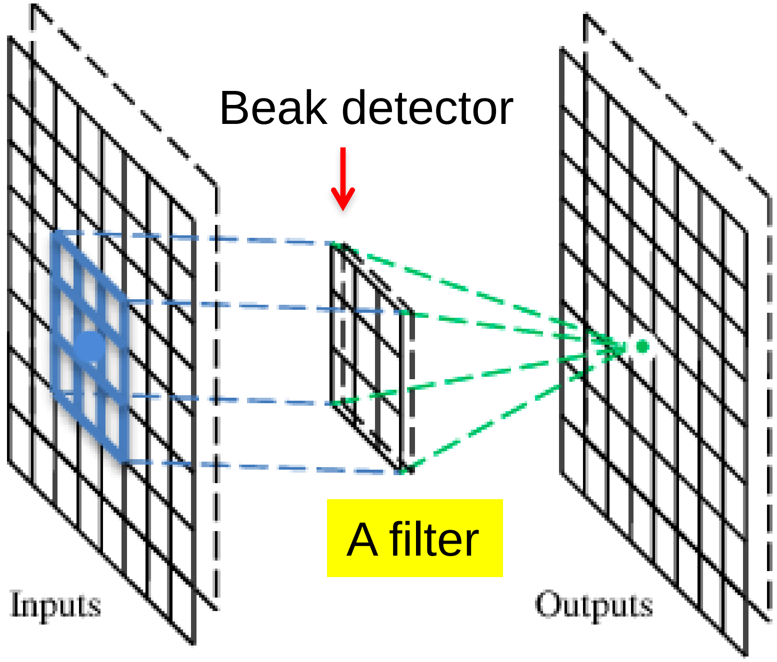
“beak” detector

What about training a lot of such “small” detectors and each detector must “move around”.



# A convolutional layer

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



# Convolution

**These 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

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

Filter 2

⋮ ⋮

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

# 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 |

Dot product

3 -1

6 x 6 image

# 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

# 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 |

# Convolution

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

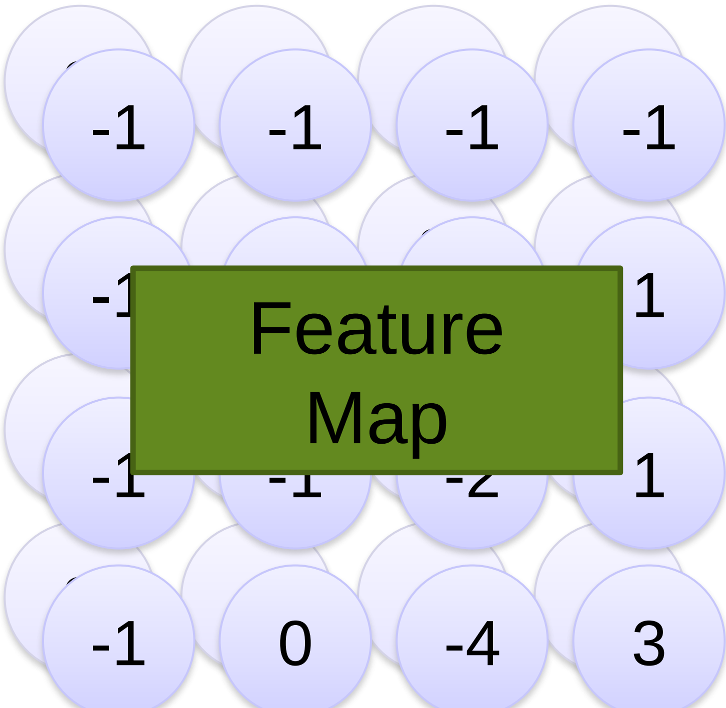
Filter 2

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

Repeat this for each filter




Two 4 x 4 images
Forming 2 x 4 x 4 matrix

# Color image: RGB 3 channels

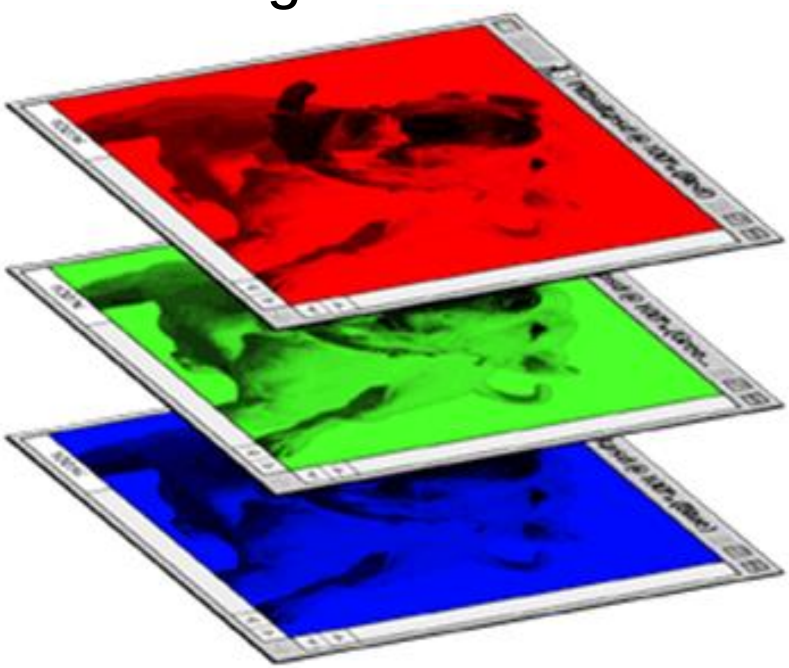
| | | |
|---|---|---|
| 1 | -1 | -1 |
| -1 | 1 | -1 |
| -1 | -1 | 1 |

Filter 1

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

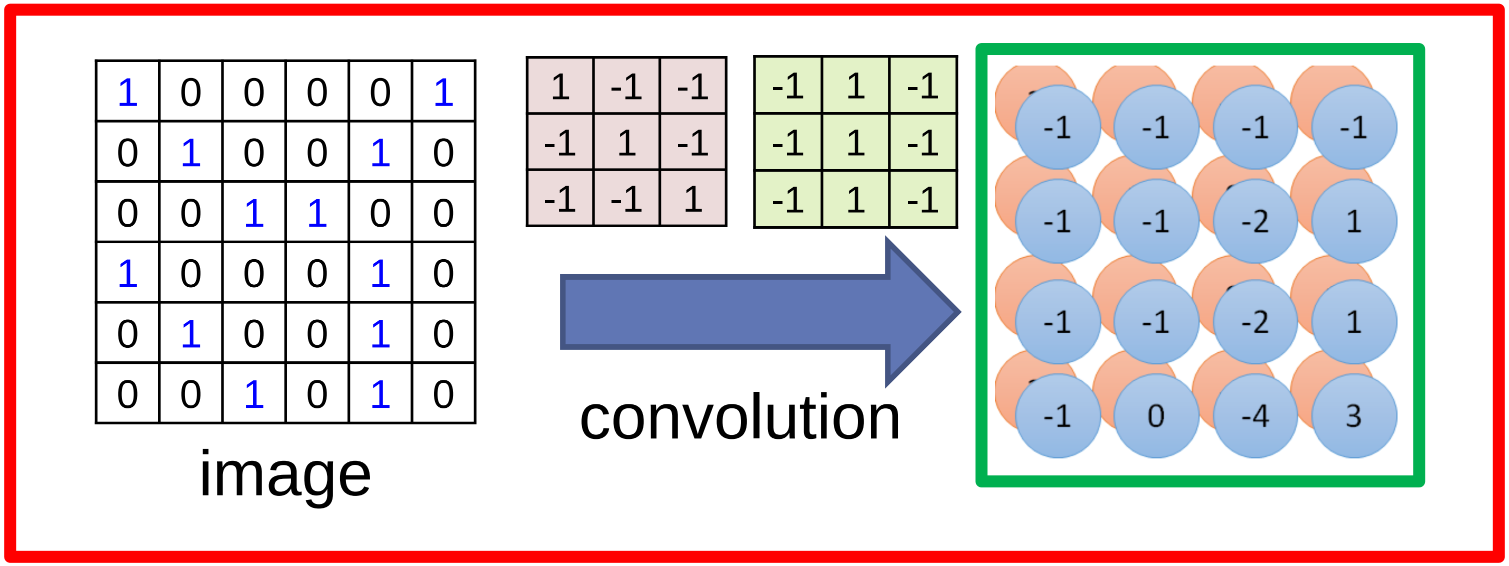
Filter 2

Color image

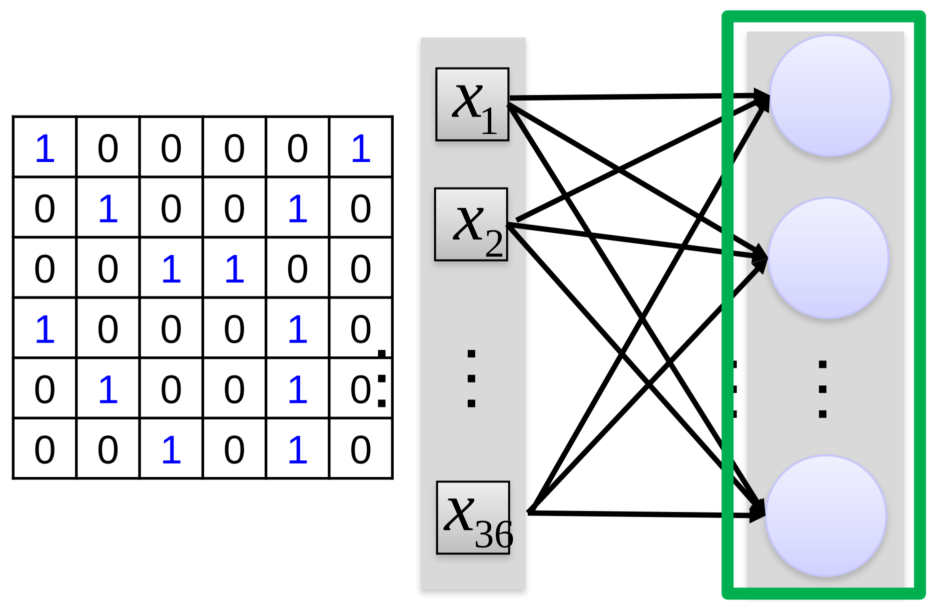


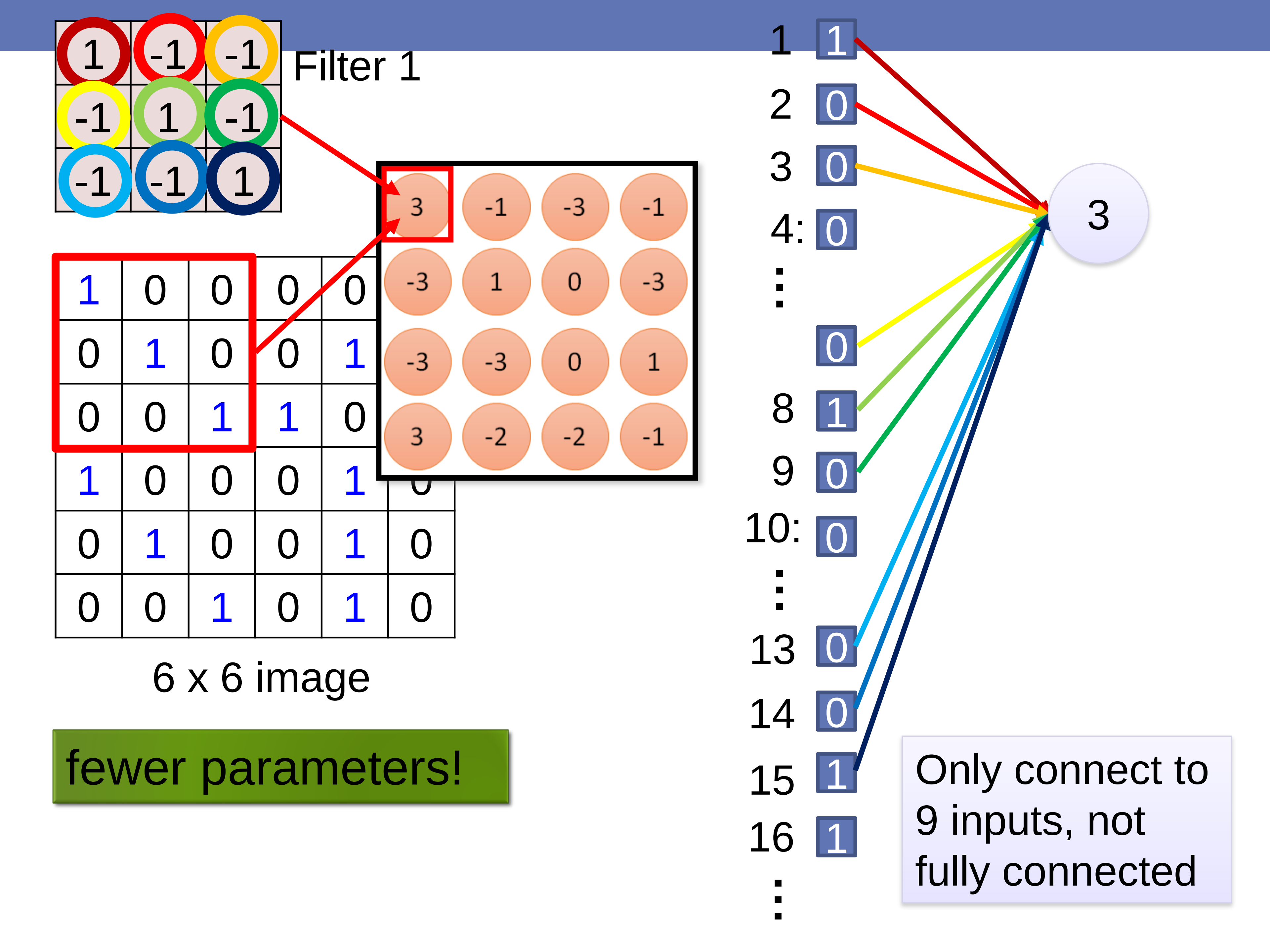
| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |

# Convolution v.s. Fully Connected

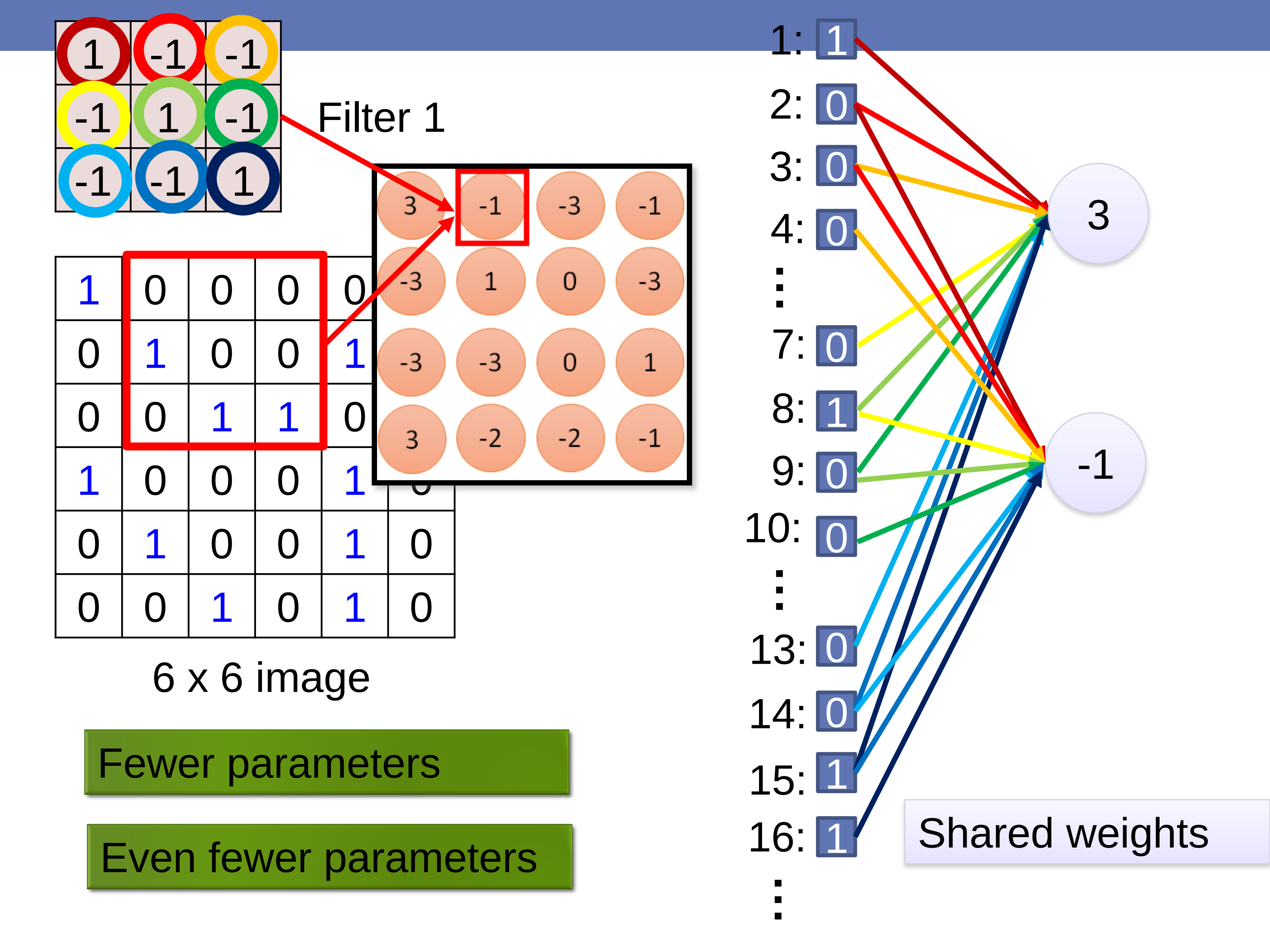



Fully-connected



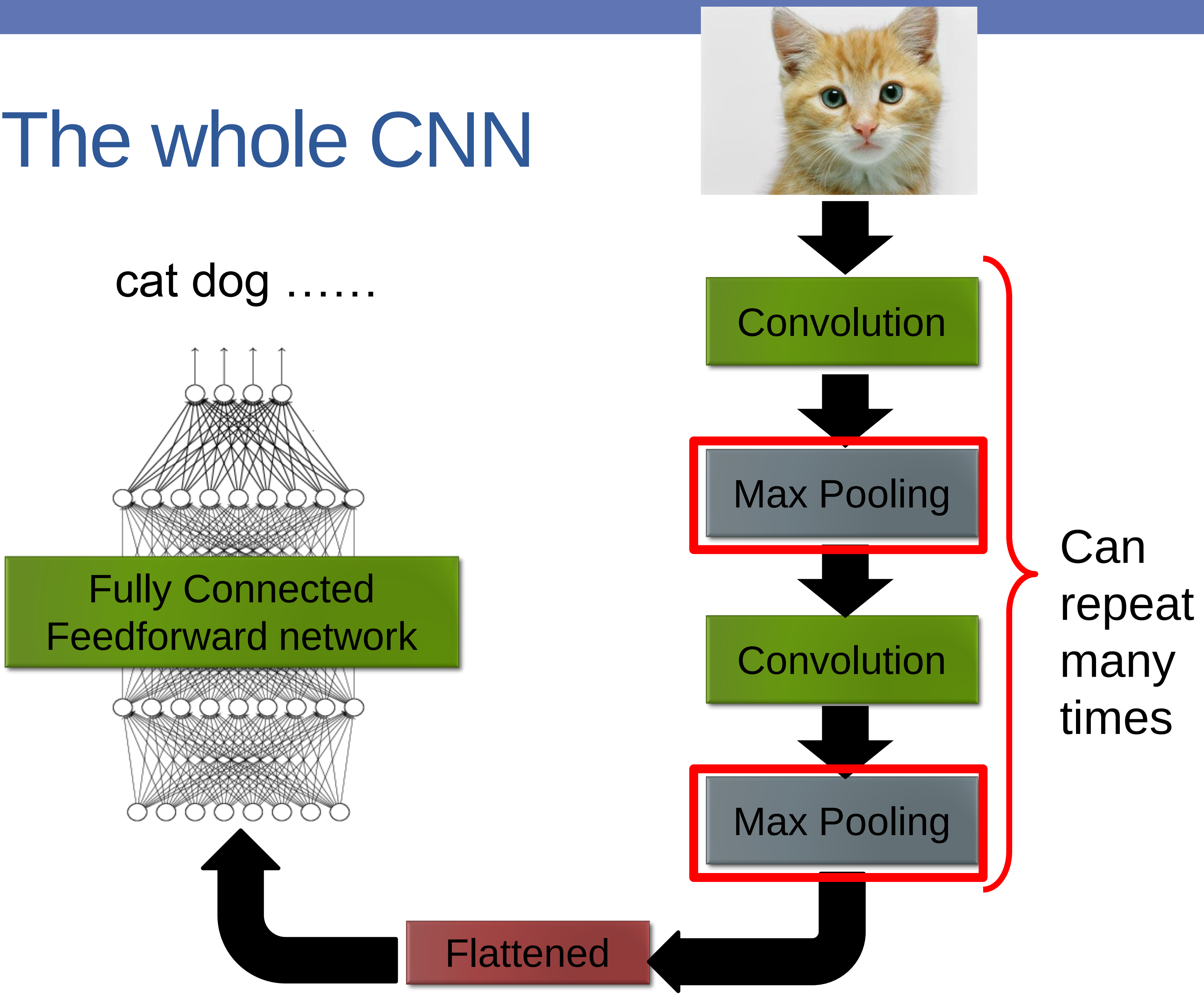


Filter 1
6 x 6 image
fewer parameters!
Only connect to 9 inputs, not fully connected



1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -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
6 x 6 image
Fewer parameters
Even fewer 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
-1
Shared weights

# The whole CNN



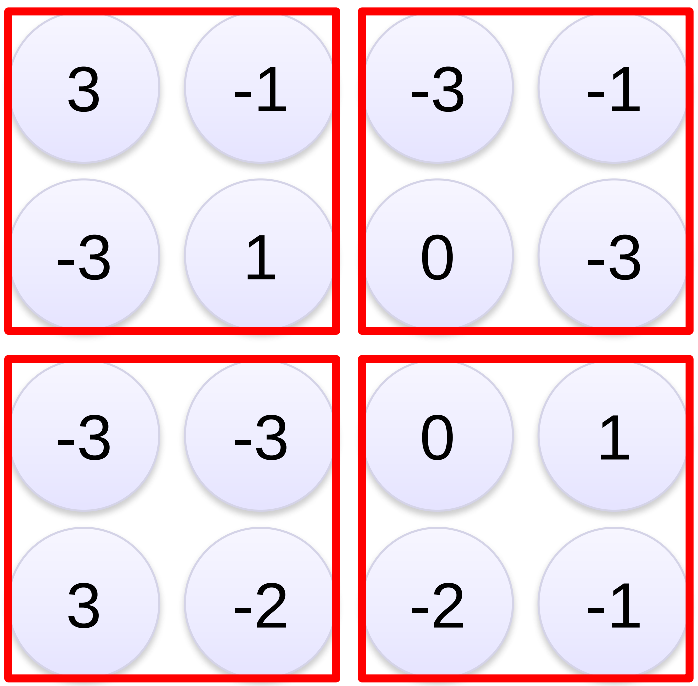
# Max Pooling

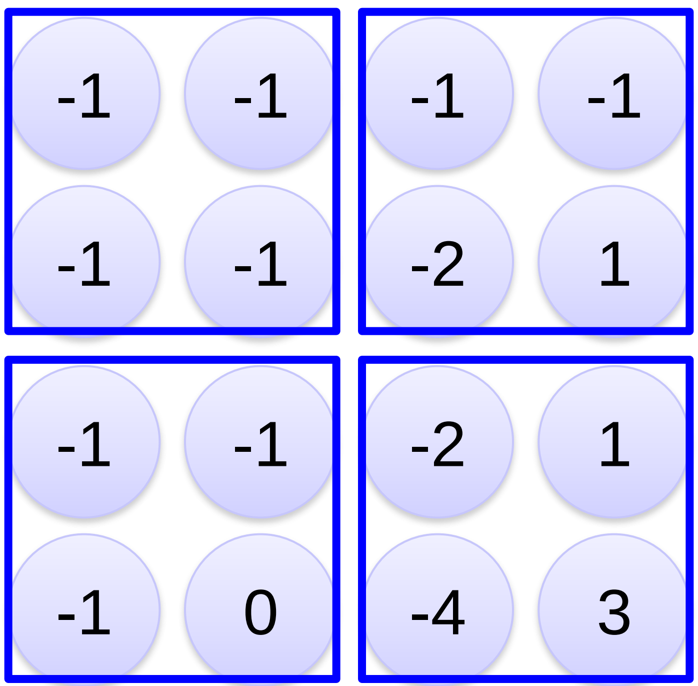
| | | |
|---|---|---|
| 1 | -1 | -1 |
| -1 | 1 | -1 |
| -1 | -1 | 1 |

Filter 1

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

Filter 2



# Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

We can subsample the pixels to make image smaller

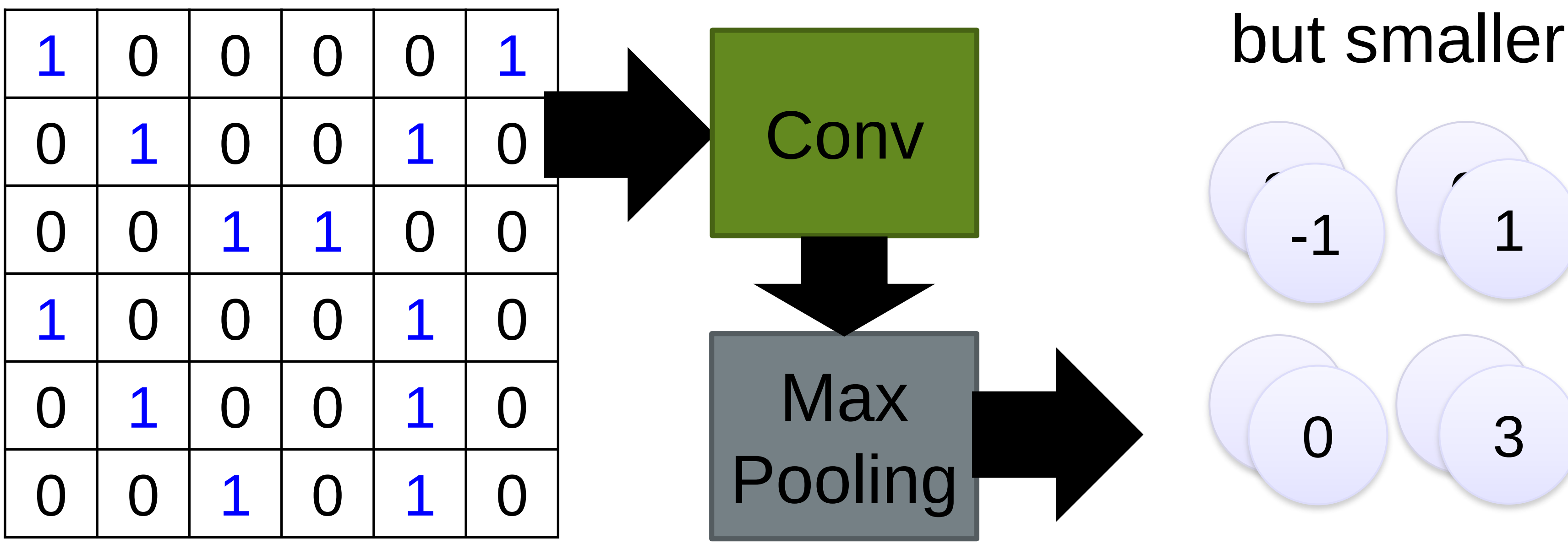
fewer parameters to characterize the image

# A CNN compresses a fully connected network in two ways:

- Reducing number of connections
- Shared weights on the edges to recognize objects independent of location
- Max pooling further reduces the complexity

# Max Pooling

New image
but smaller

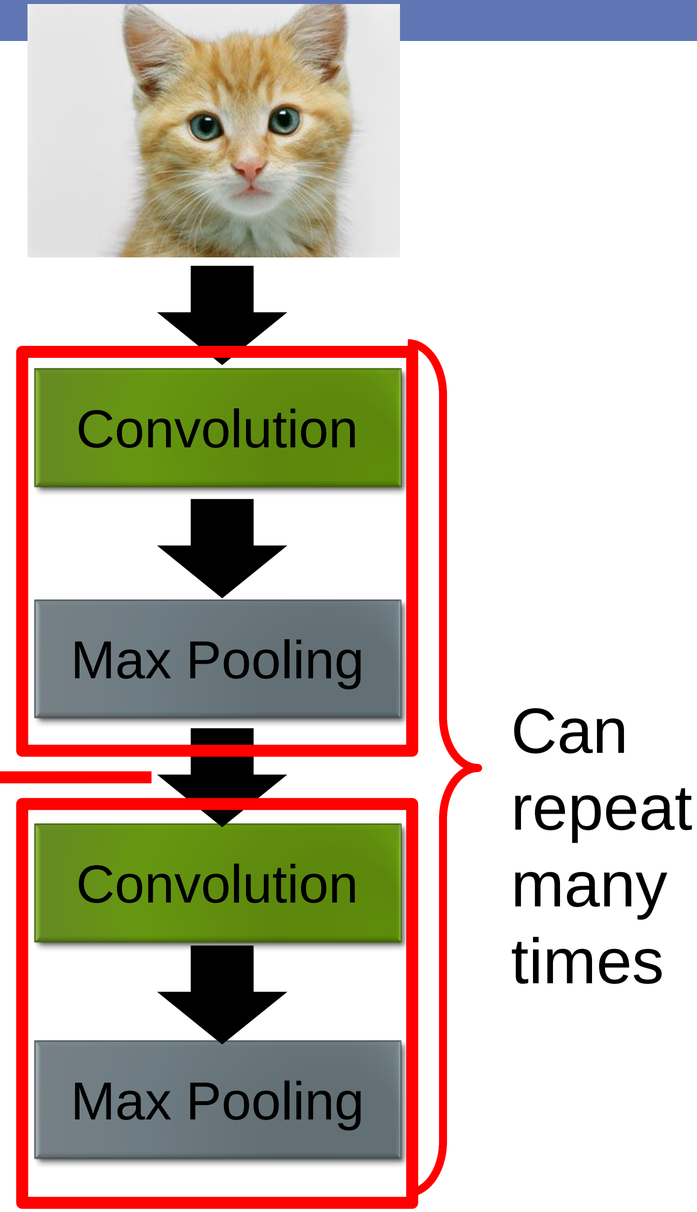


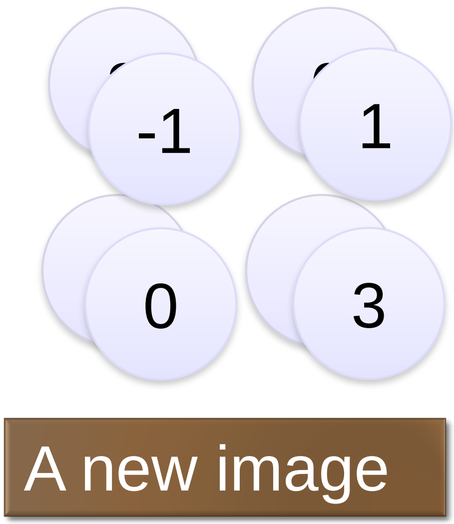

6 x 6 image

2 x 2 image

Each filter
is a channel

# The whole CNN

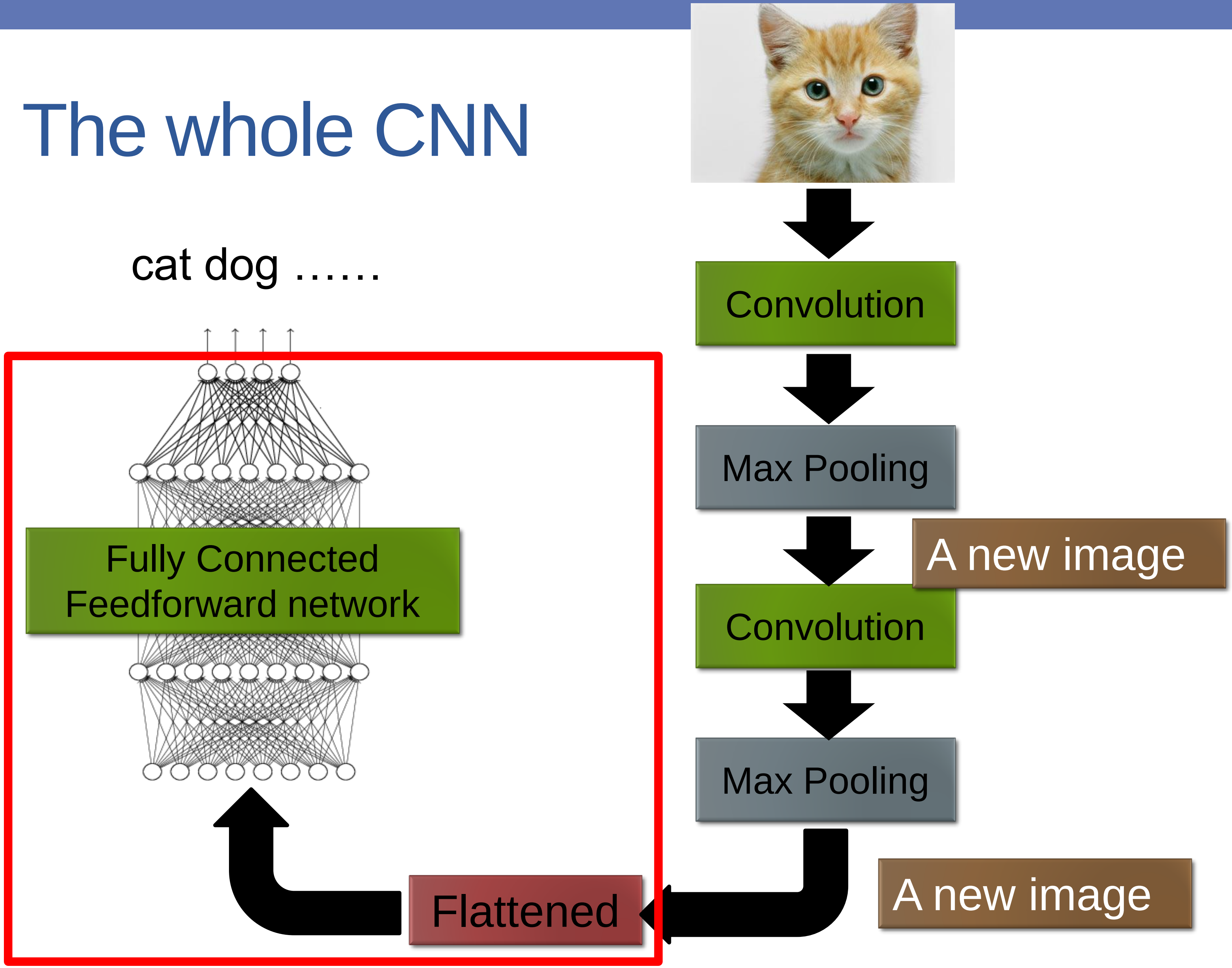


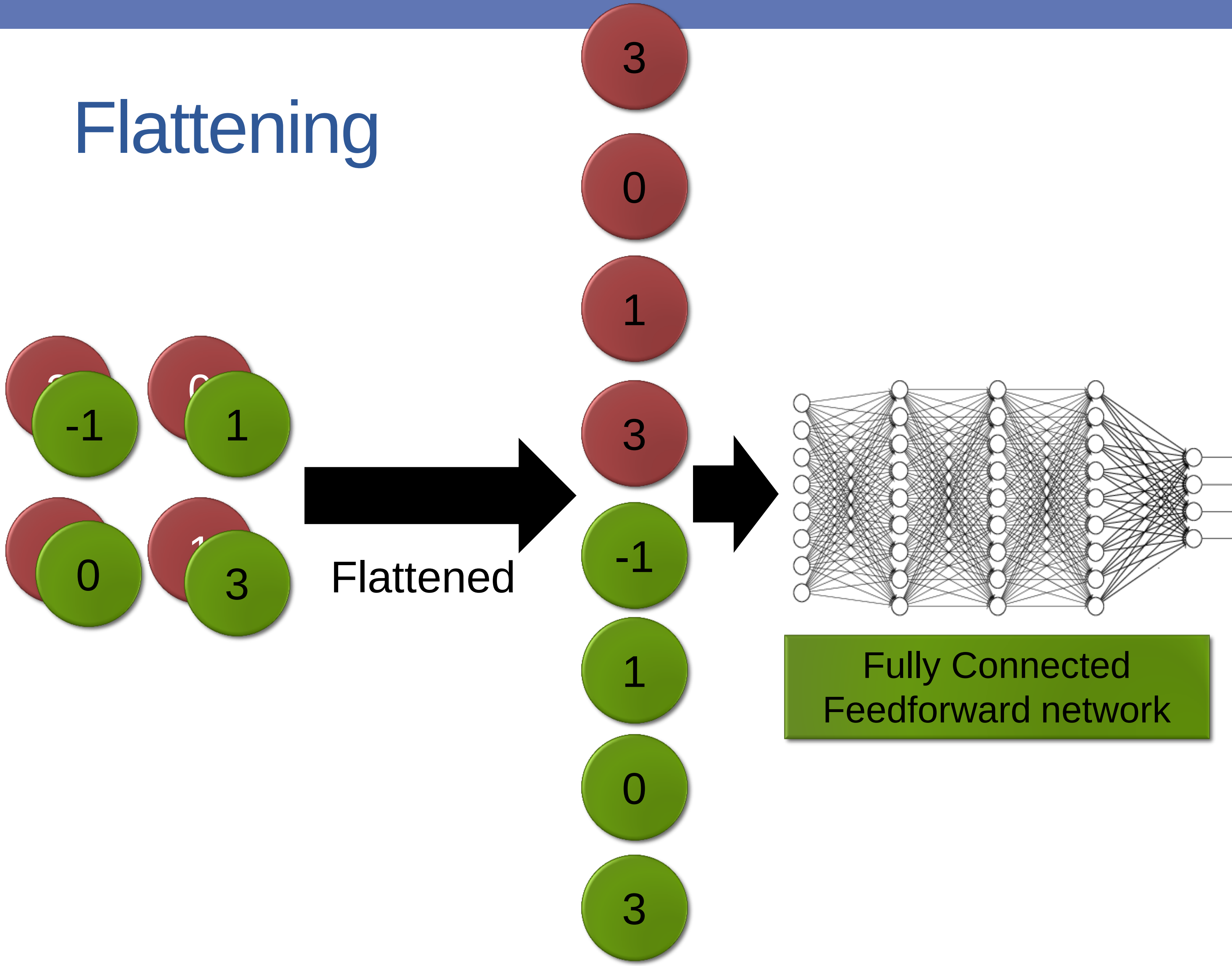

Smaller than the original image

The number of channels is the number of filters

# The whole CNN



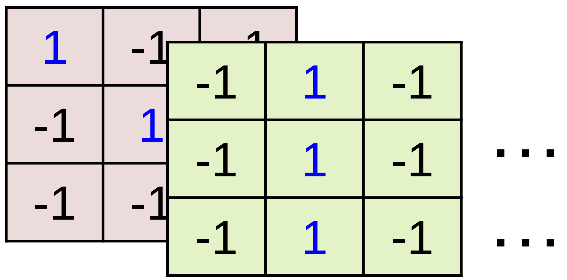
# Flattening


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

# *CNN in Keras*

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

```
model2.add( Convolution2D( 25, 3, 3,
                input_shape=(28,28,1)) )
```



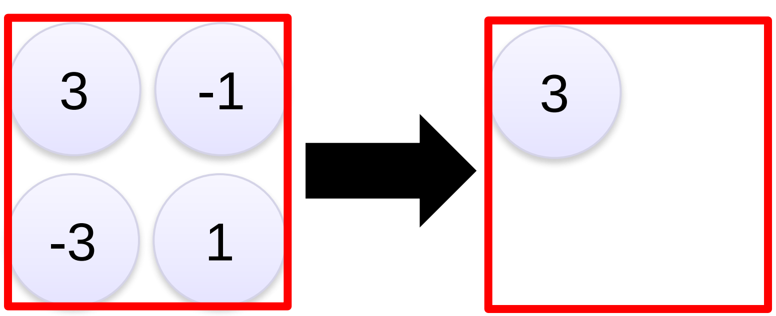

There are 25 3x3 filters.

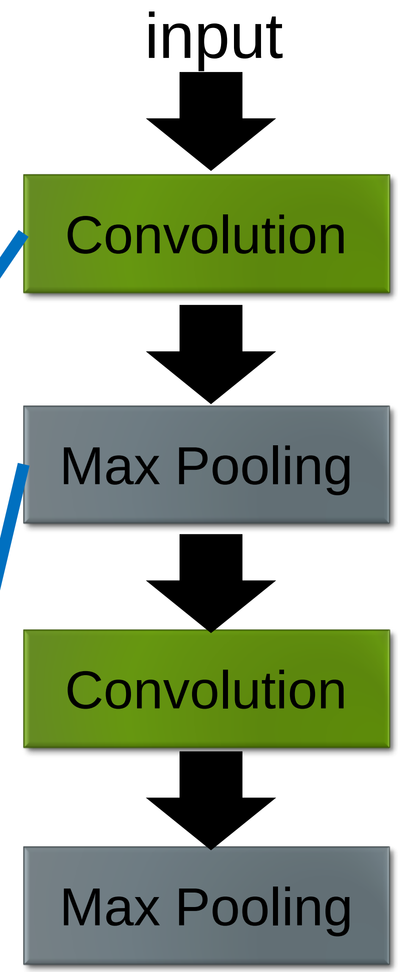
Input_shape = ( 28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

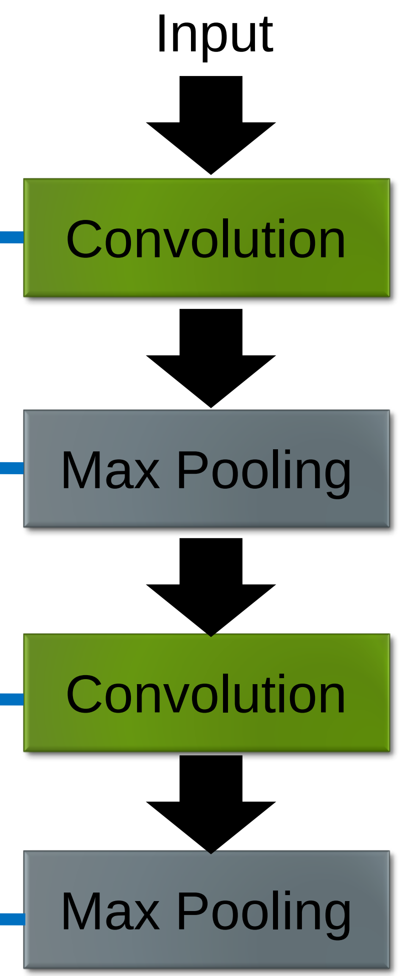
```
model2.add(MaxPooling2D((2,2)))
```



# CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***




1 x 28 x 28

```
model2.add( Convolution2D( 25,3,3,
               input_shape=(28,28,1)) )
```

How many parameters for each filter?

9

25 x 26 x 26

```
model2.add(MaxPooling2D((2,2)))
```

25 x 13 x 13

```
model2.add(Convolution2D(50,3,3))
```

How many parameters for each filter?

225=
25x9

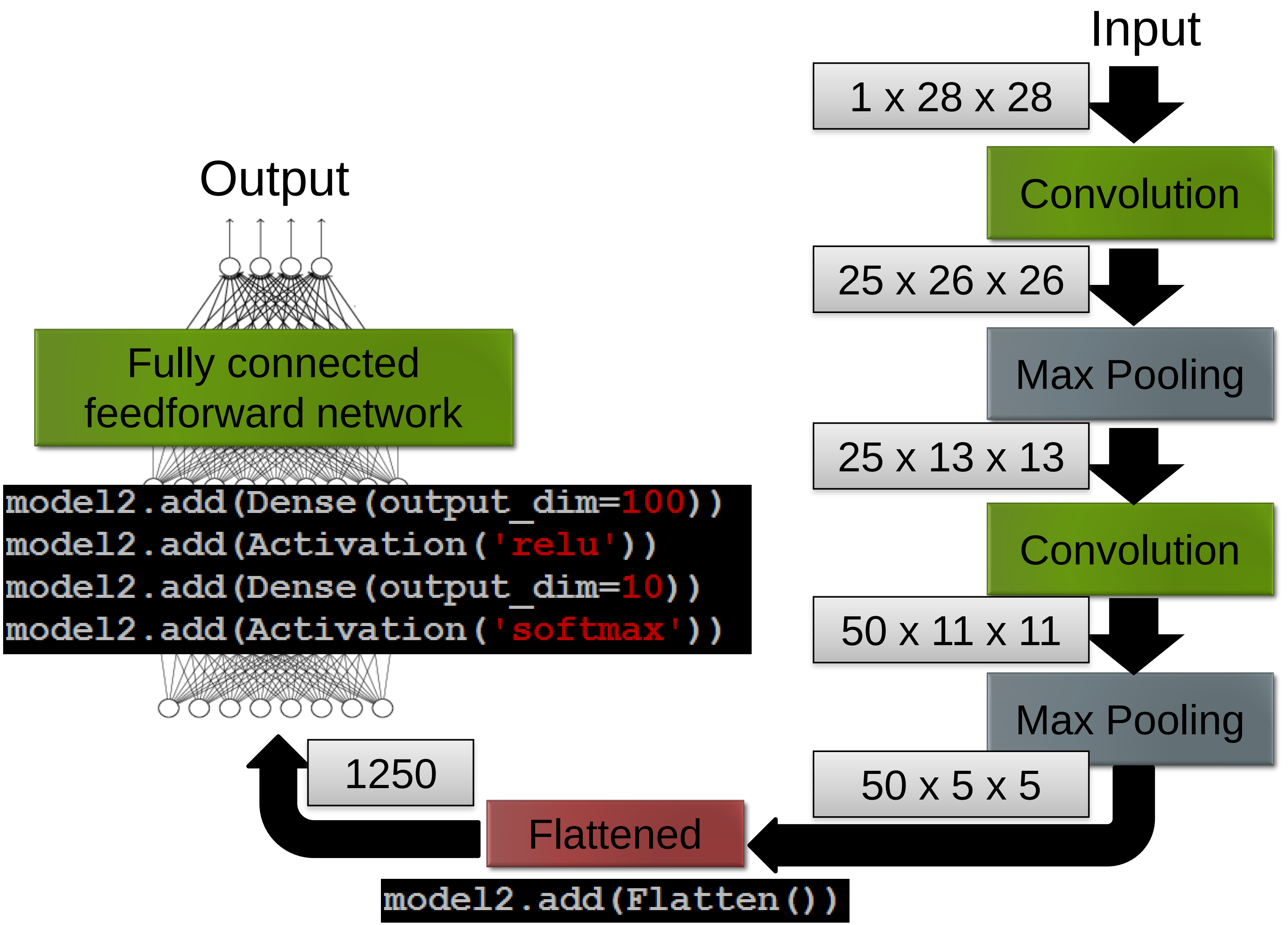
50 x 11 x 11

```
model2.add(MaxPooling2D((2,2)))
```

50 x 5 x 5

# CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***




```
model2.add(Dense(output_dim=100))
model2.add(Activation('relu'))
model2.add(Dense(output_dim=10))
model2.add(Activation('softmax'))
```

```
model2.add(Flatten())
```

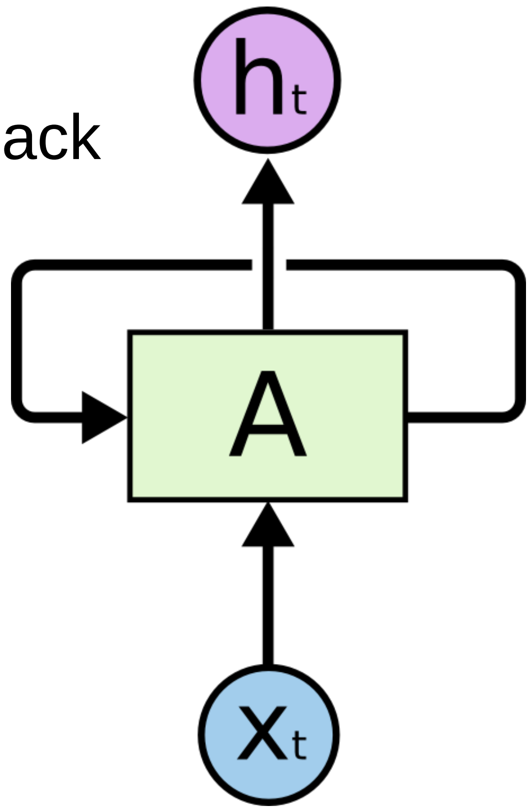
# Advantage of convolution

- Multiplies the same weight everywhere on the image
- Suppose the filter is a feature finder that identifies the car, ie. outputs a comparatively high +ve number, when input is a car. It will find a car no matter where it is on the image
- Layers of convolution behave the same. First layer will find small, simple features anywhere on the image(eg. A line or edge). Next layer will find more complex features, and so on.
- Weight sharing leads to better generalization

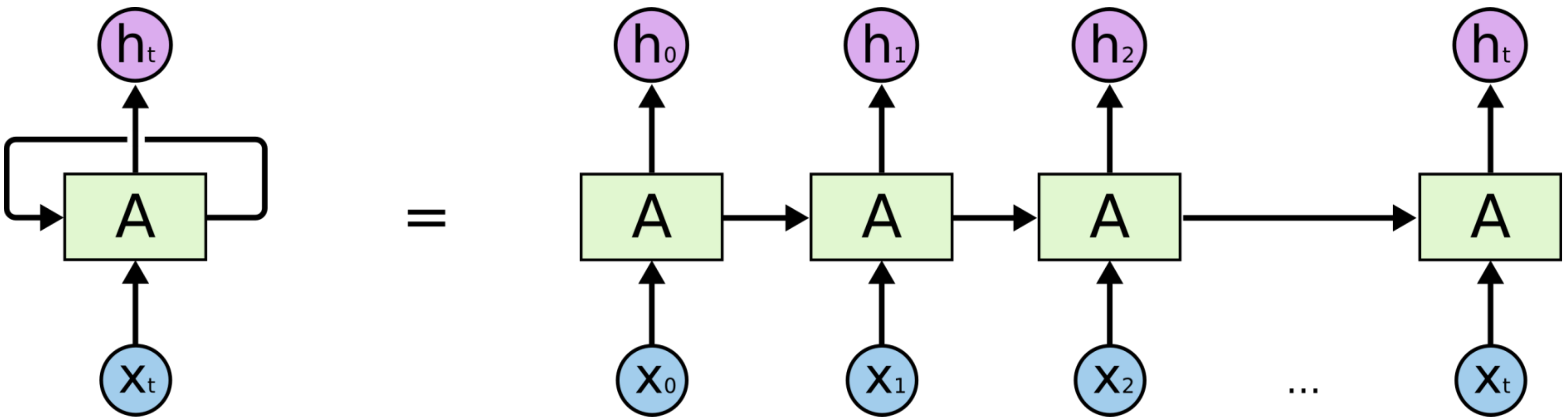
# PART II:
# NEURAL NETWORK WITH MEMORY (RNN)

# Construing from Memory

- RNN are the NN architectures with feedback



- **An unrolled recurrent neural network**

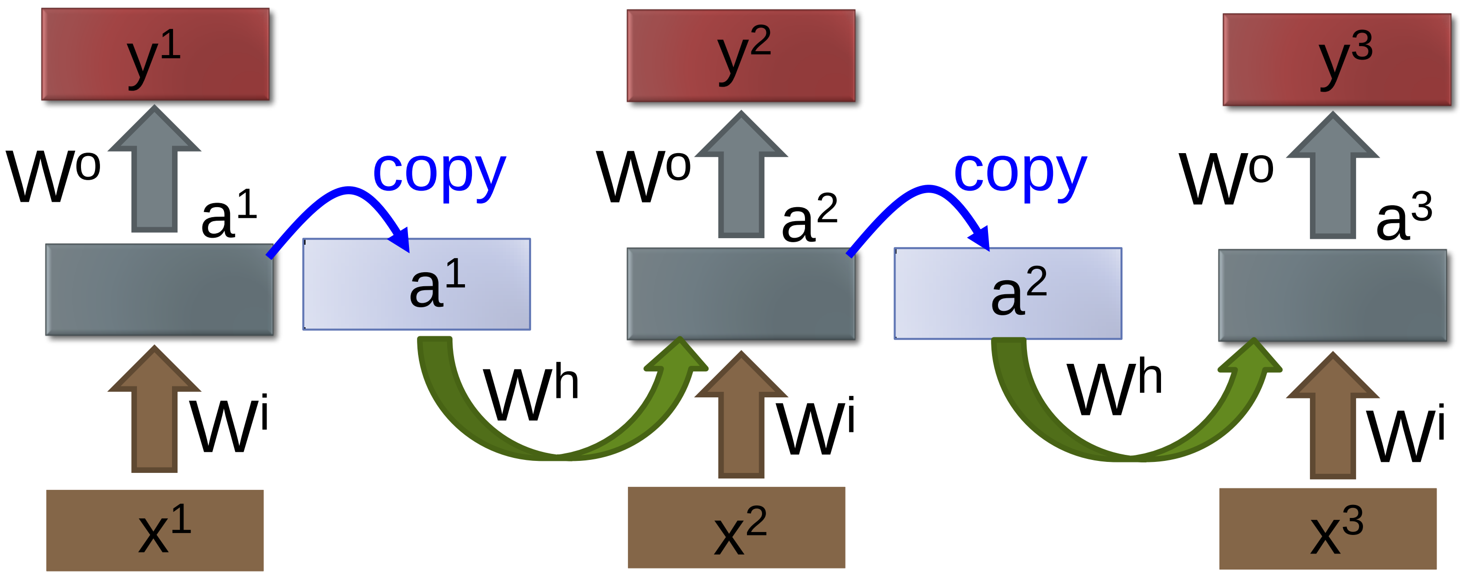



- Long memory vs Short memory
- Examples:

  Clouds are in sky.

  She grew up in **France**…..She speaks fluent **French**.

# RNN

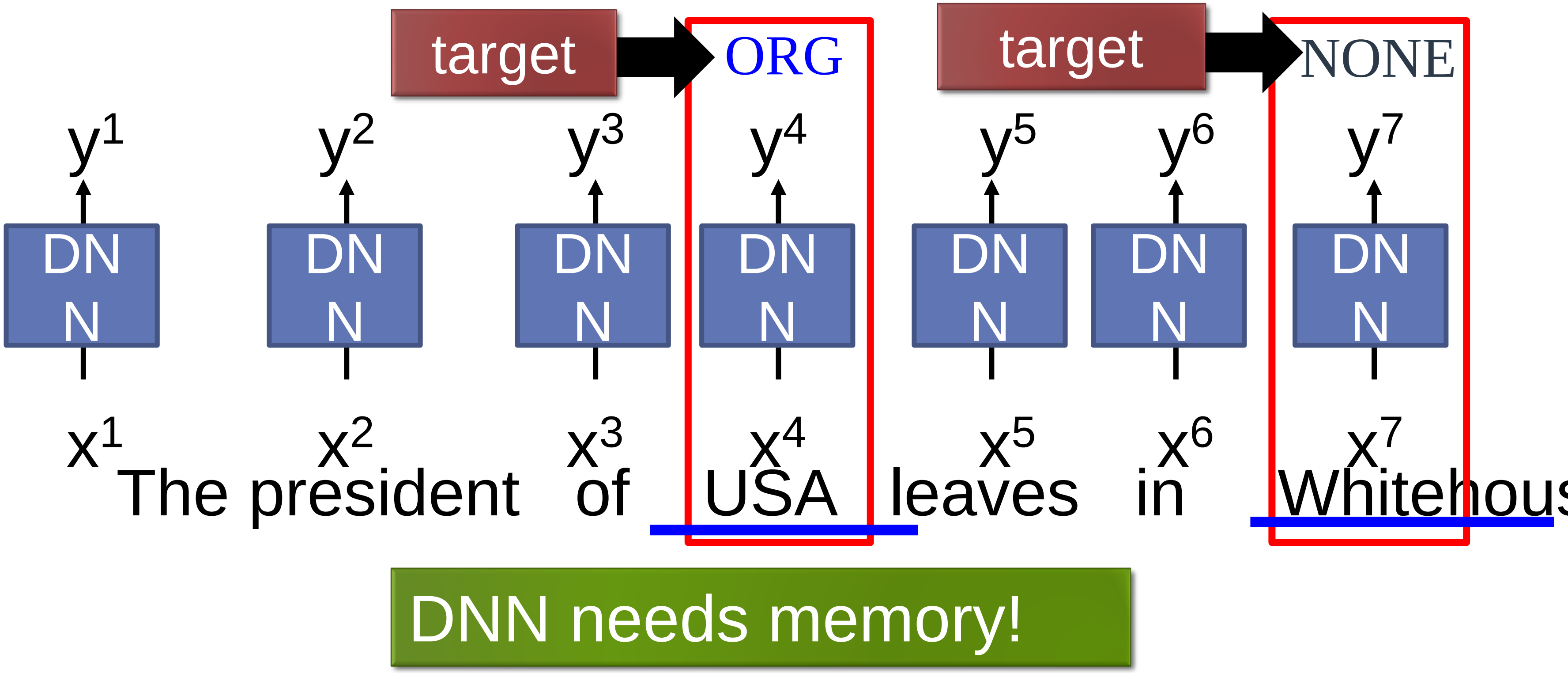



The same network is used again and again.

Output $y^i$ depends on $x^1$, $x^2$, …… $x^i$

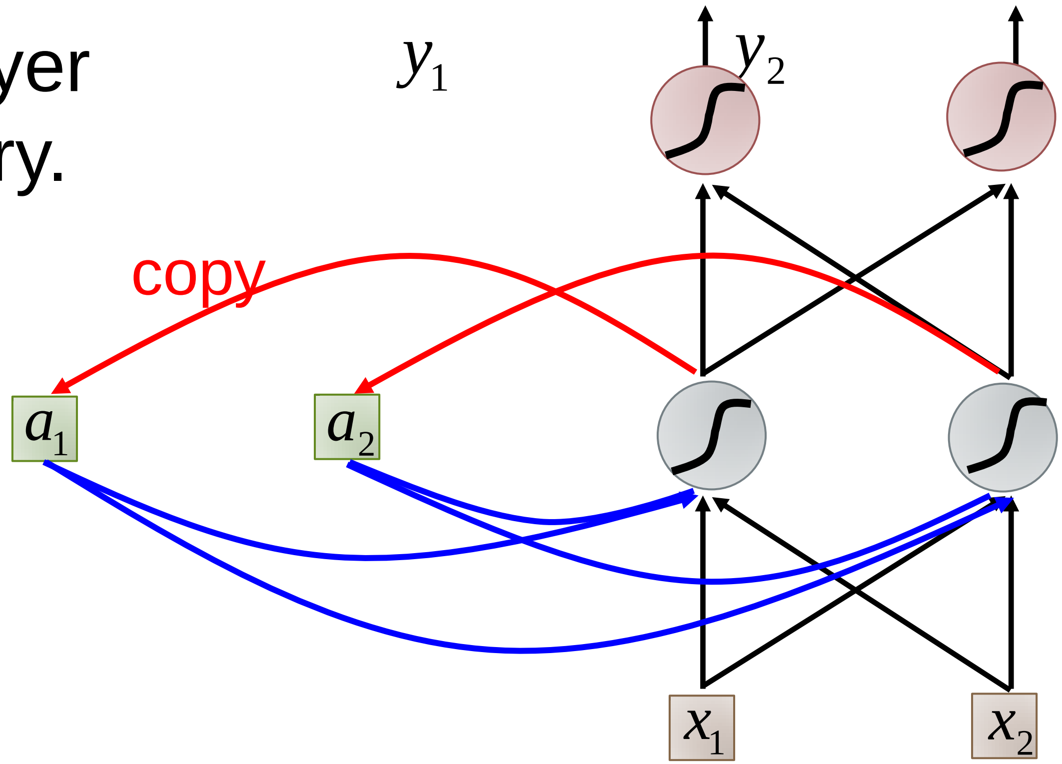
# Neural Network needs Memory

- Name Entity Recognition
  - Detecting entities like name of people, locations, organization, etc. in a sentence.



# Recurrent Neural Network (RNN)

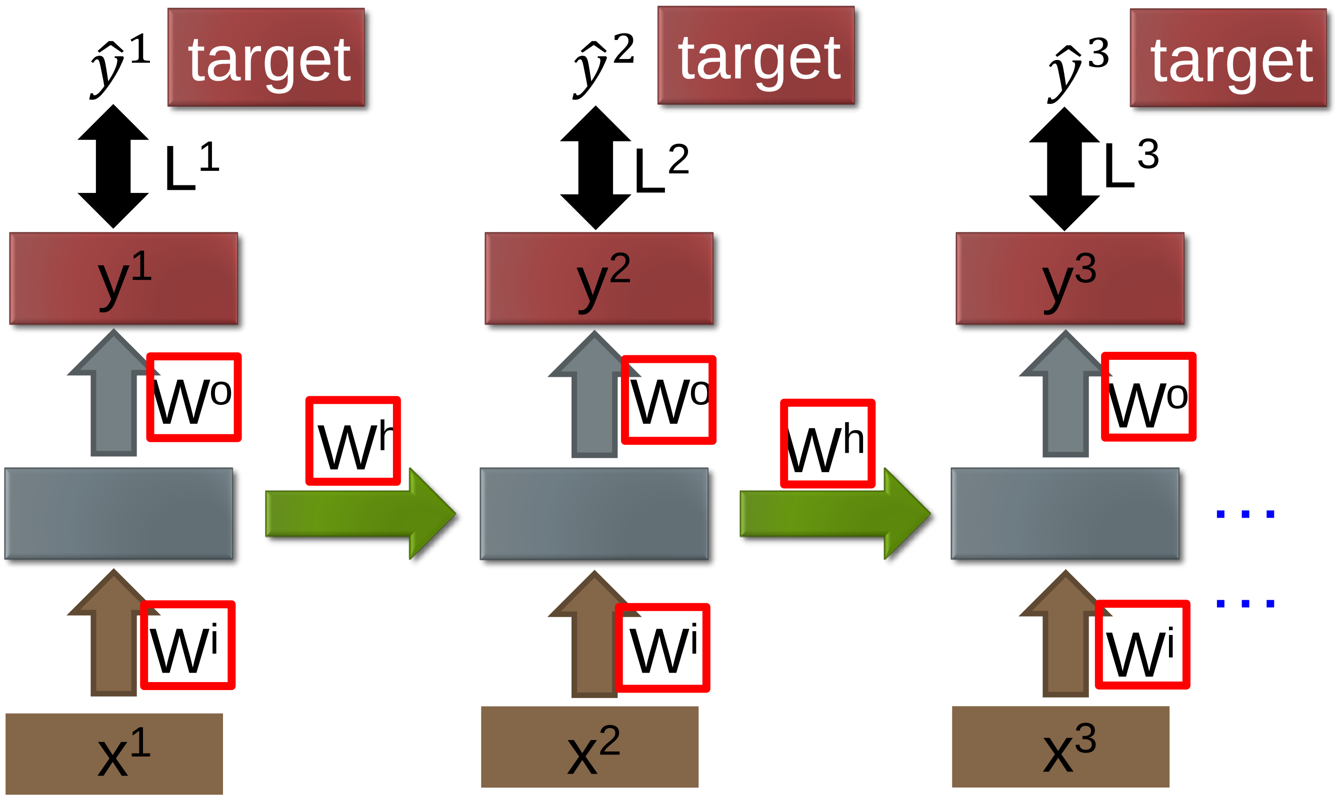
The output of hidden layer are stored in the memory.




- Memory can be considered as another input.
- Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.

# RNN

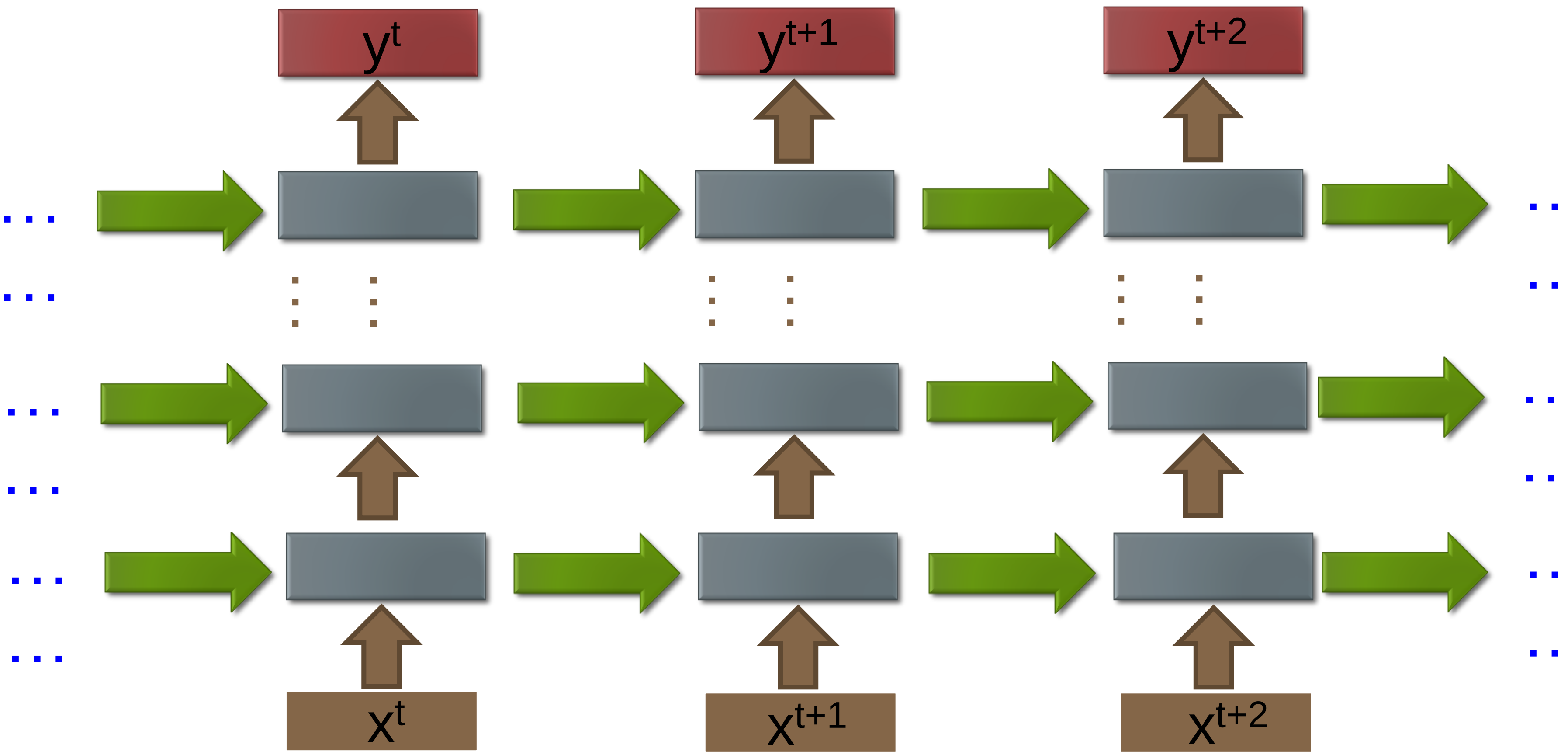
How to train?



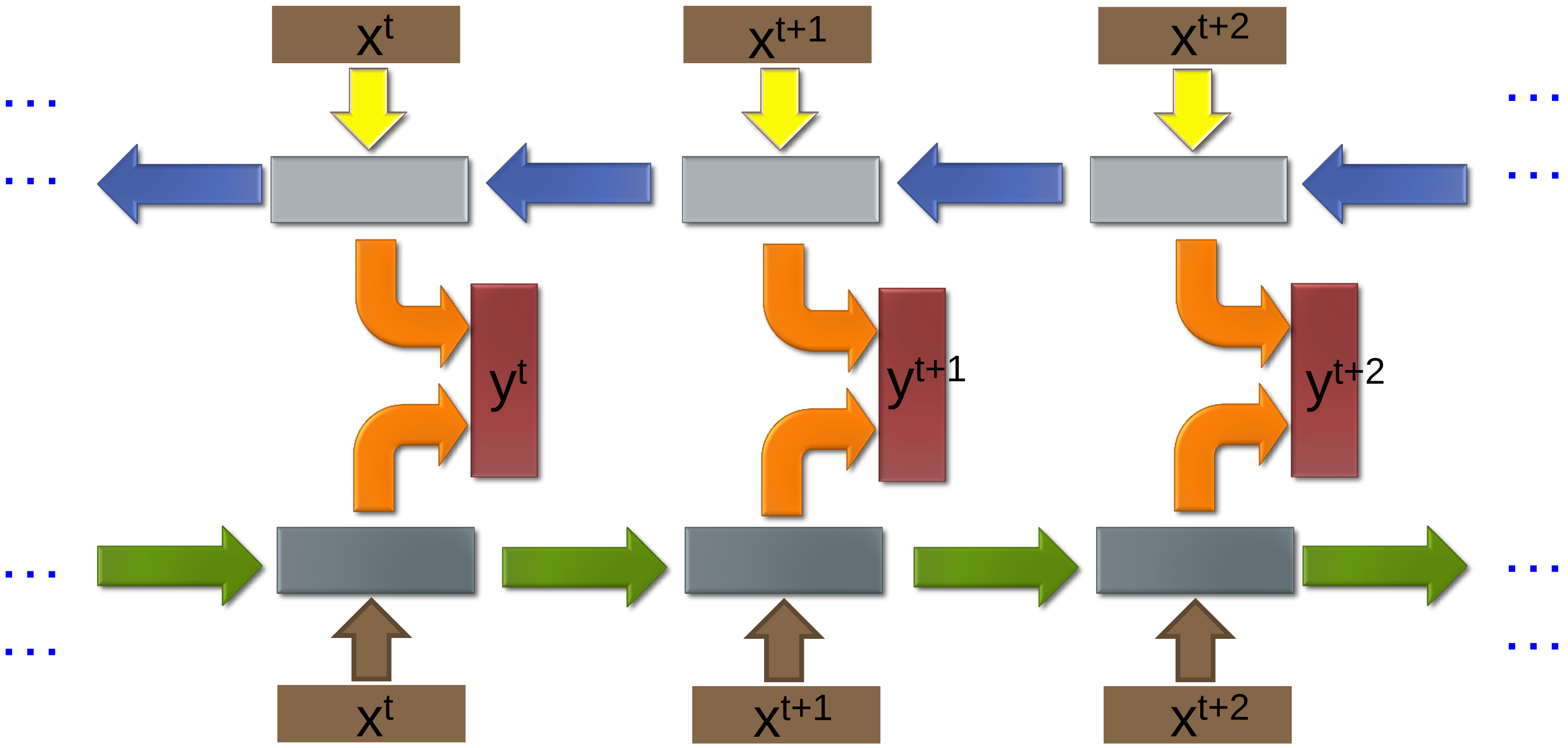

Find the network parameters to minimize the total cost:

Backpropagation through time (BPTT)

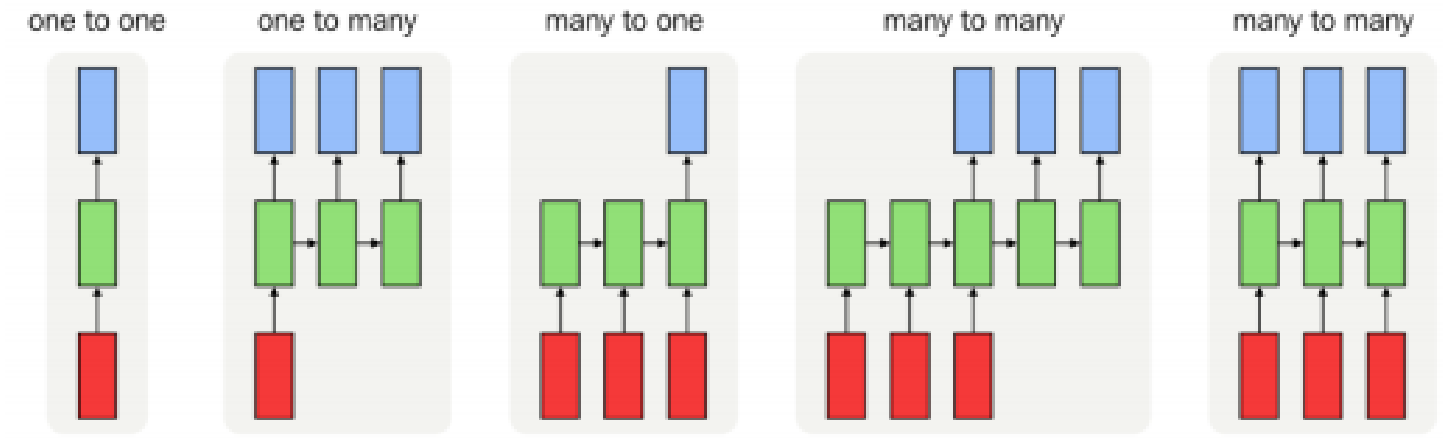
# Of course it can be deep …



# Bidirectional RNN



# No size limitation



# Vanishing or exploding gradient

$w = 1 \Rightarrow y^{1000} = 1$

$w = 1.01 \Rightarrow y^{1000} \approx 20000$

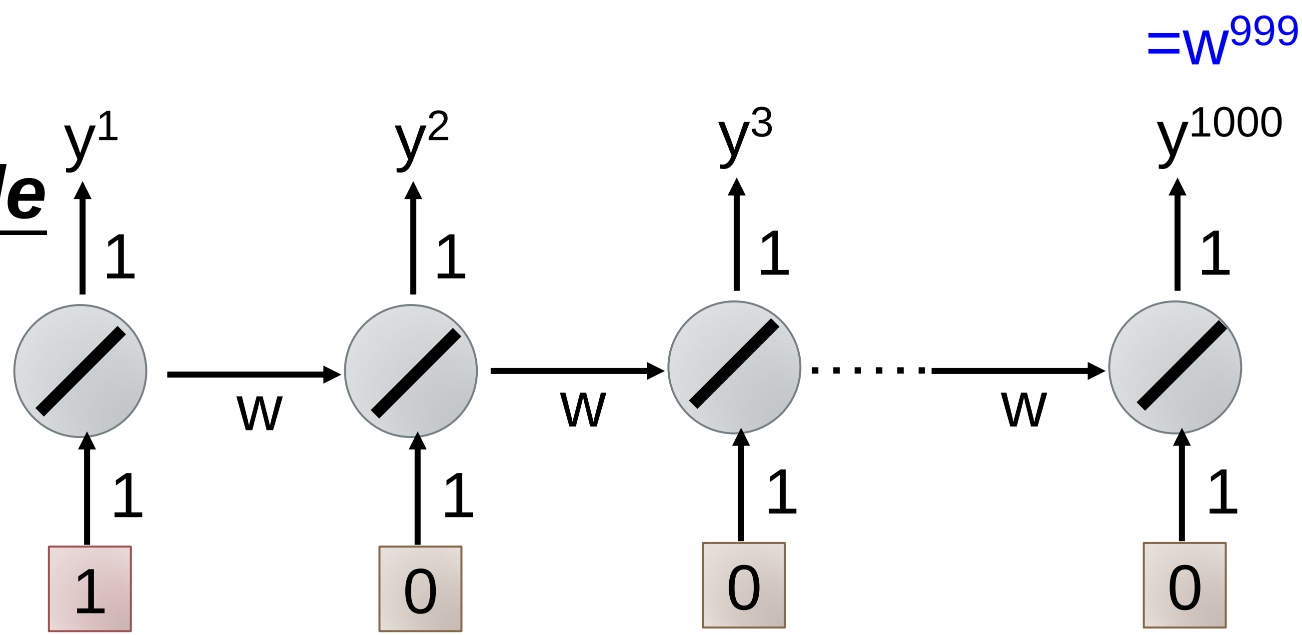
Large gradient → Small Learning rate?

$w = 0.99 \Rightarrow y^{1000} \approx 0$

$w = 0.01 \Rightarrow y^{1000} \approx 0$

small gradient → Large Learning rate?

***Toy Example***

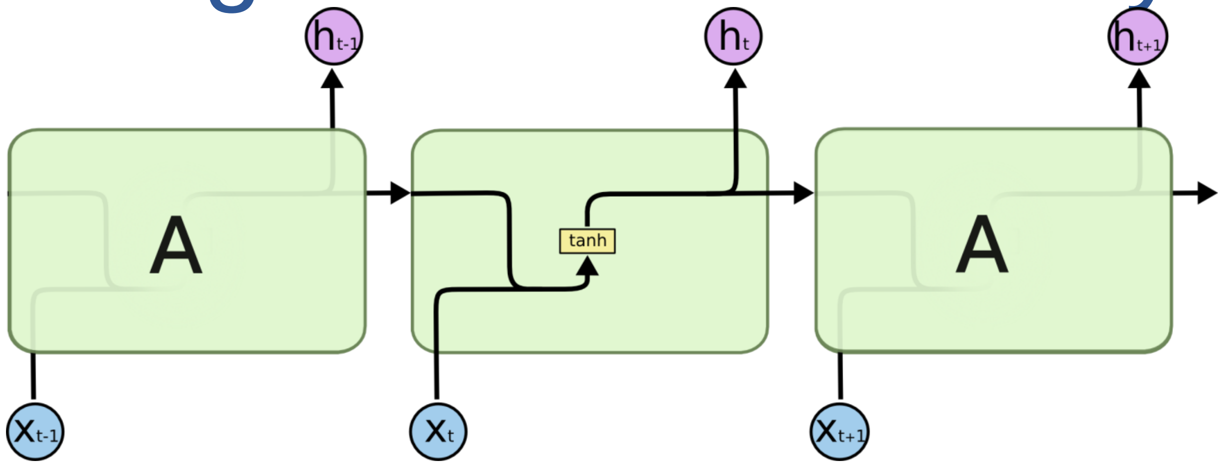


# Helpful Techniques

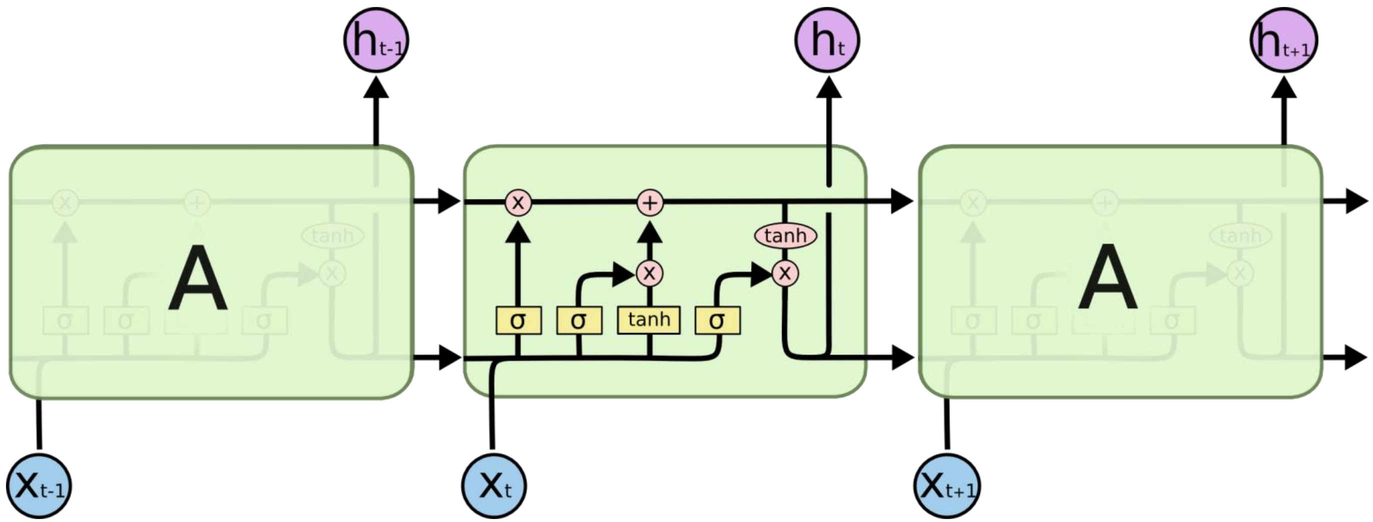
- Nesterov's Accelerated Gradient (NAG):
  - Advance momentum method
- RMS Prop
  - Advanced approach to give each parameter different learning rates
  - Considering the change of Second derivatives
- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)

# LSTM: Long Short Term Memory

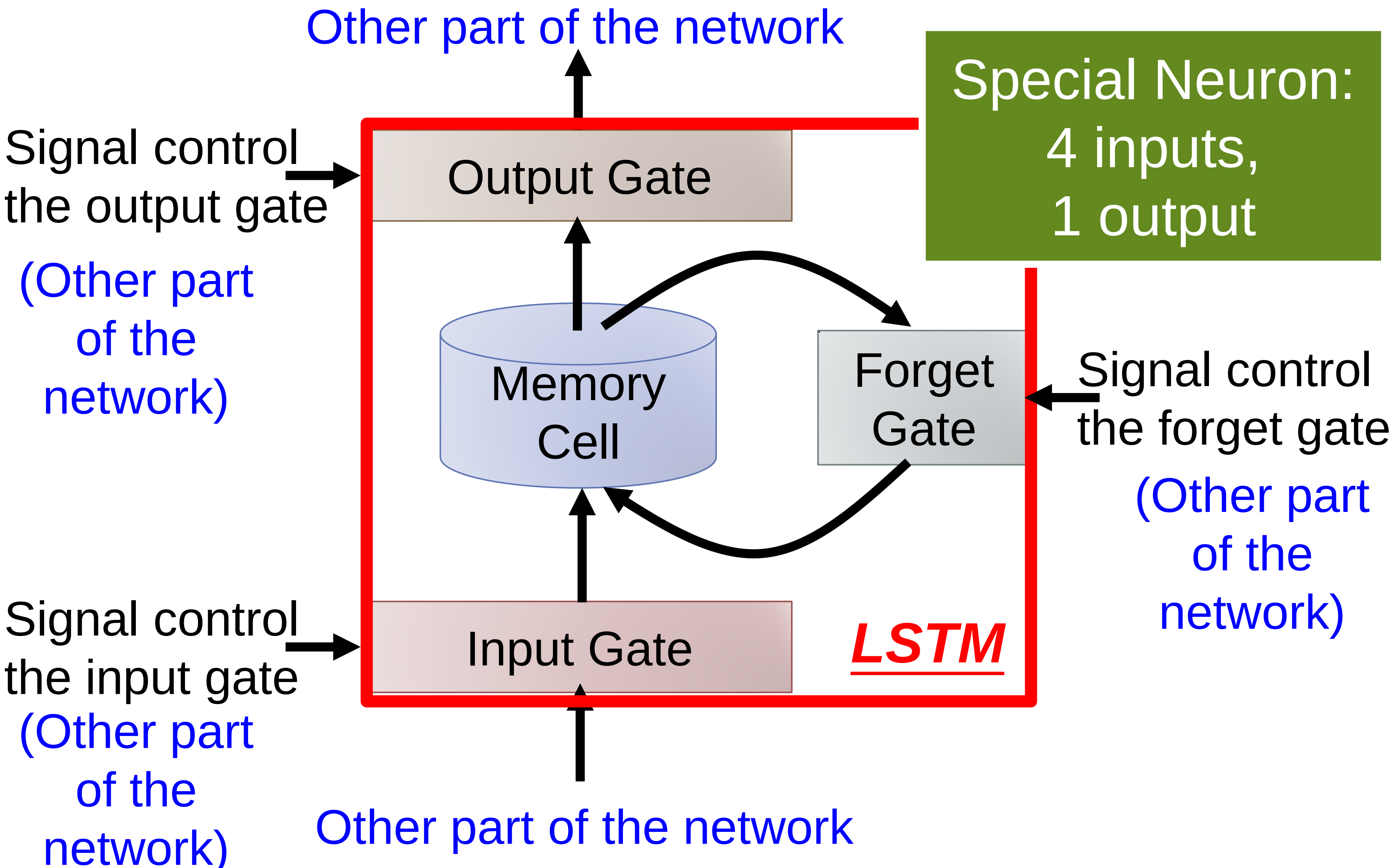
- RNN




- LSTM

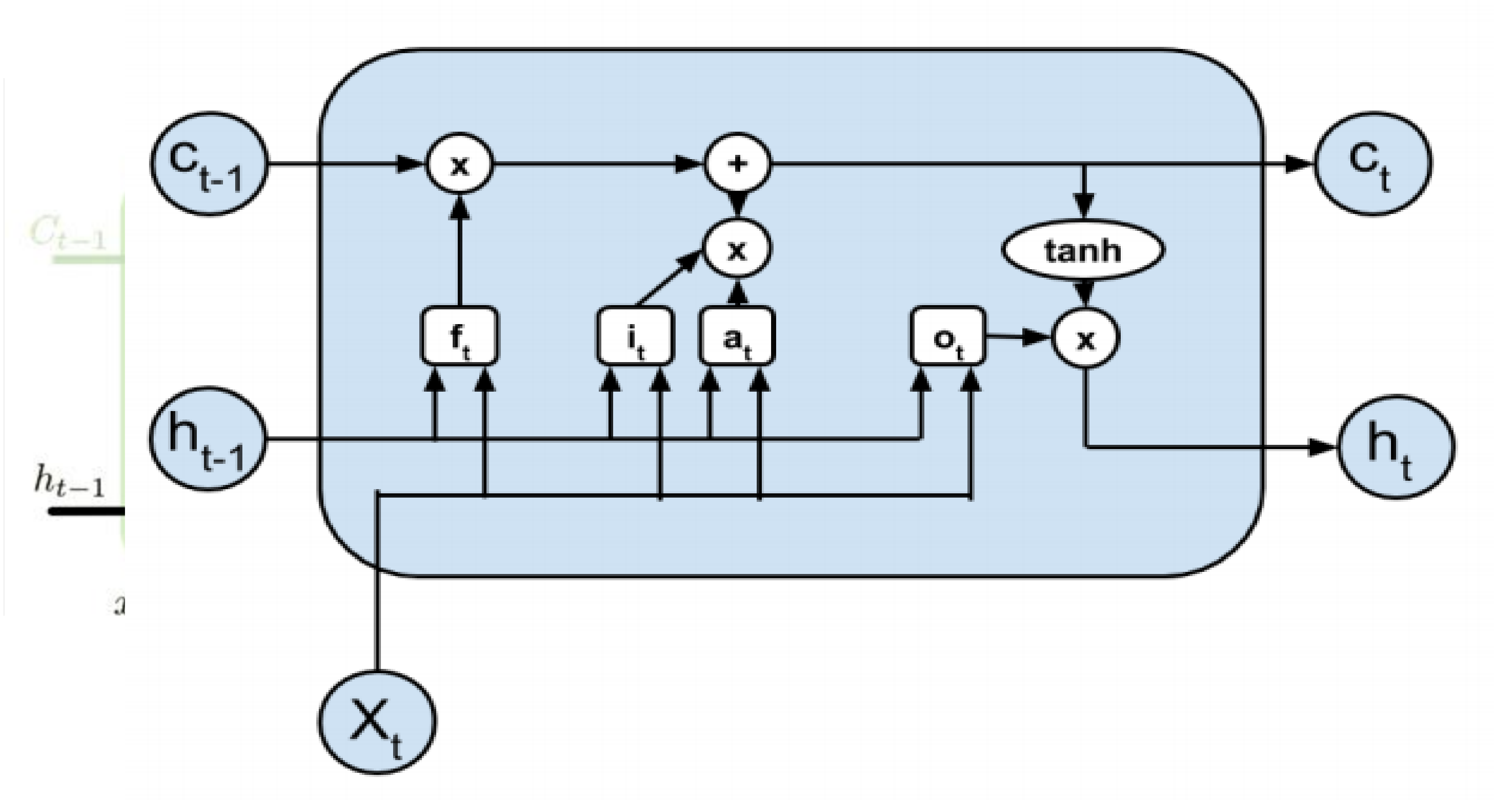


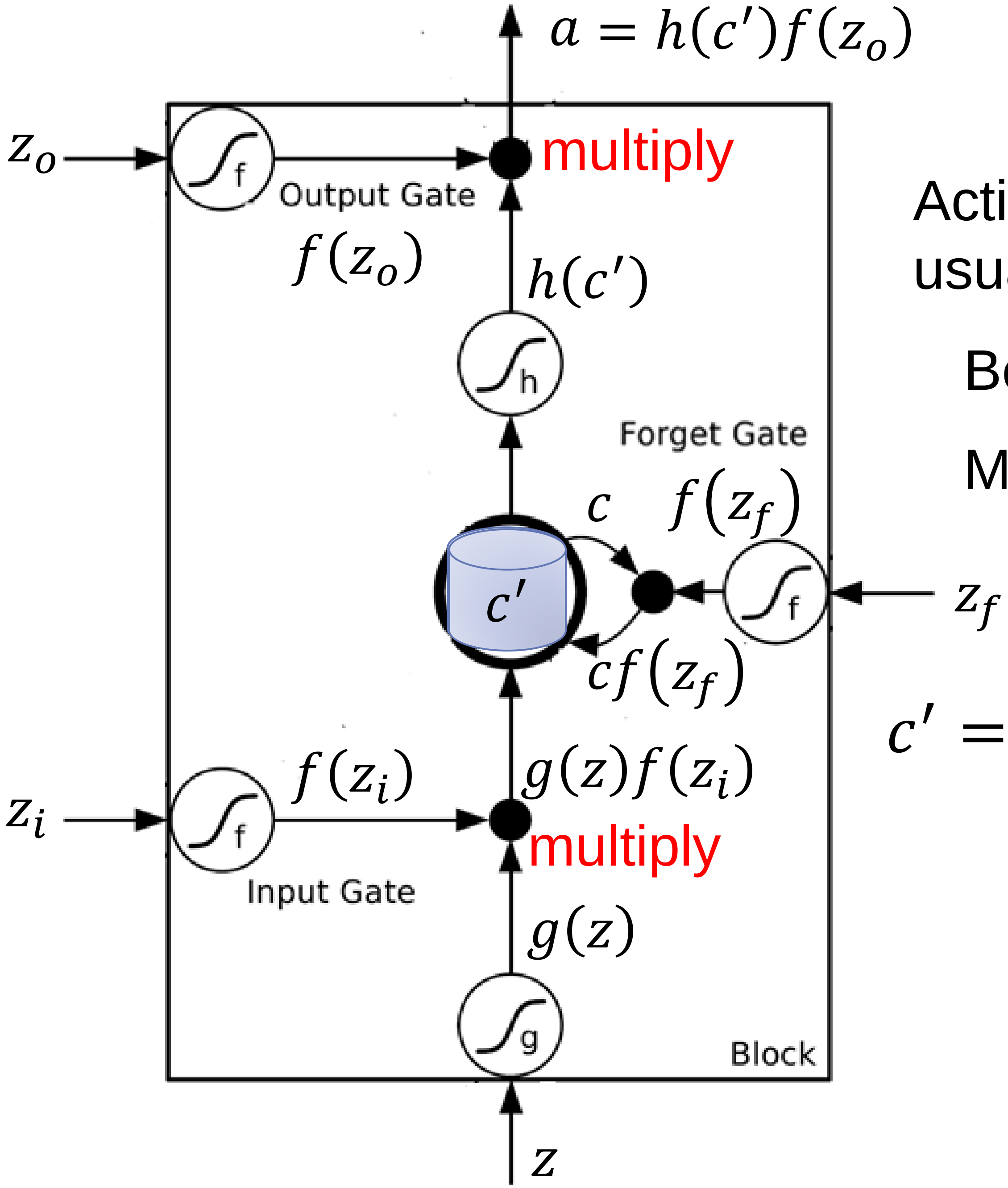
# Long Short-term Memory (LSTM)



# Core concepts behind LSTM

- Cell state (like a conveyor belt)
- Ability to remove or add information to the cell state






Activation function f is usually a sigmoid function

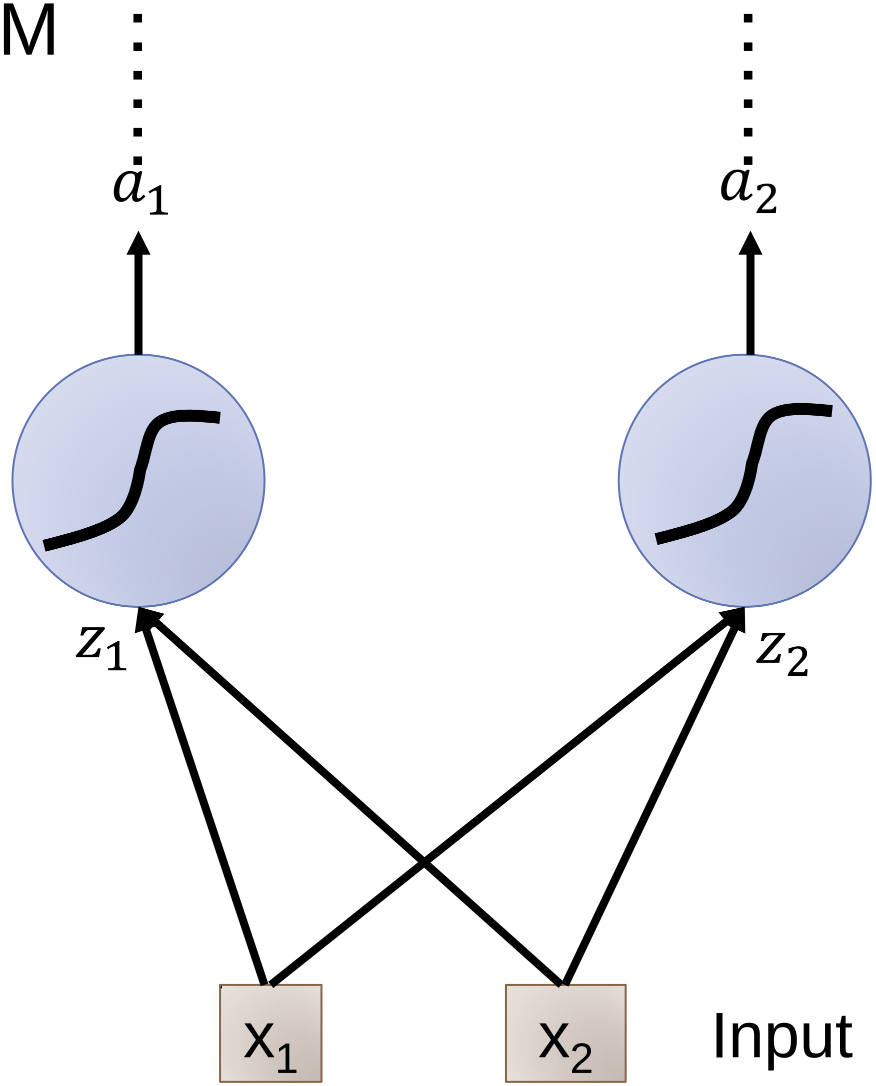
Between 0 and 1

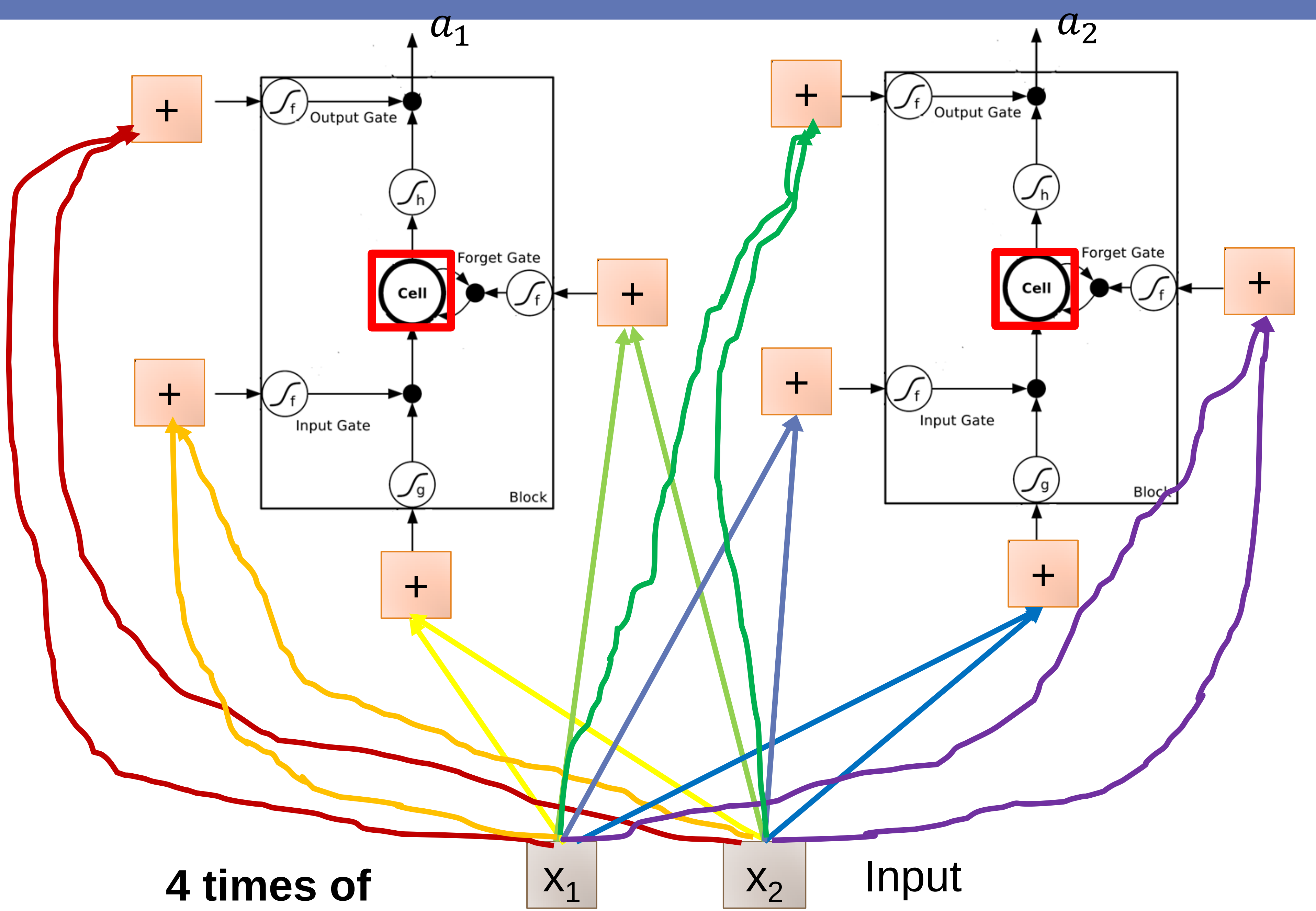
Mimic open and close gate

$$c' = g(z)f(z_i) + cf(z_f)$$

Original Network:

- Simply replace the neurons with LSTM

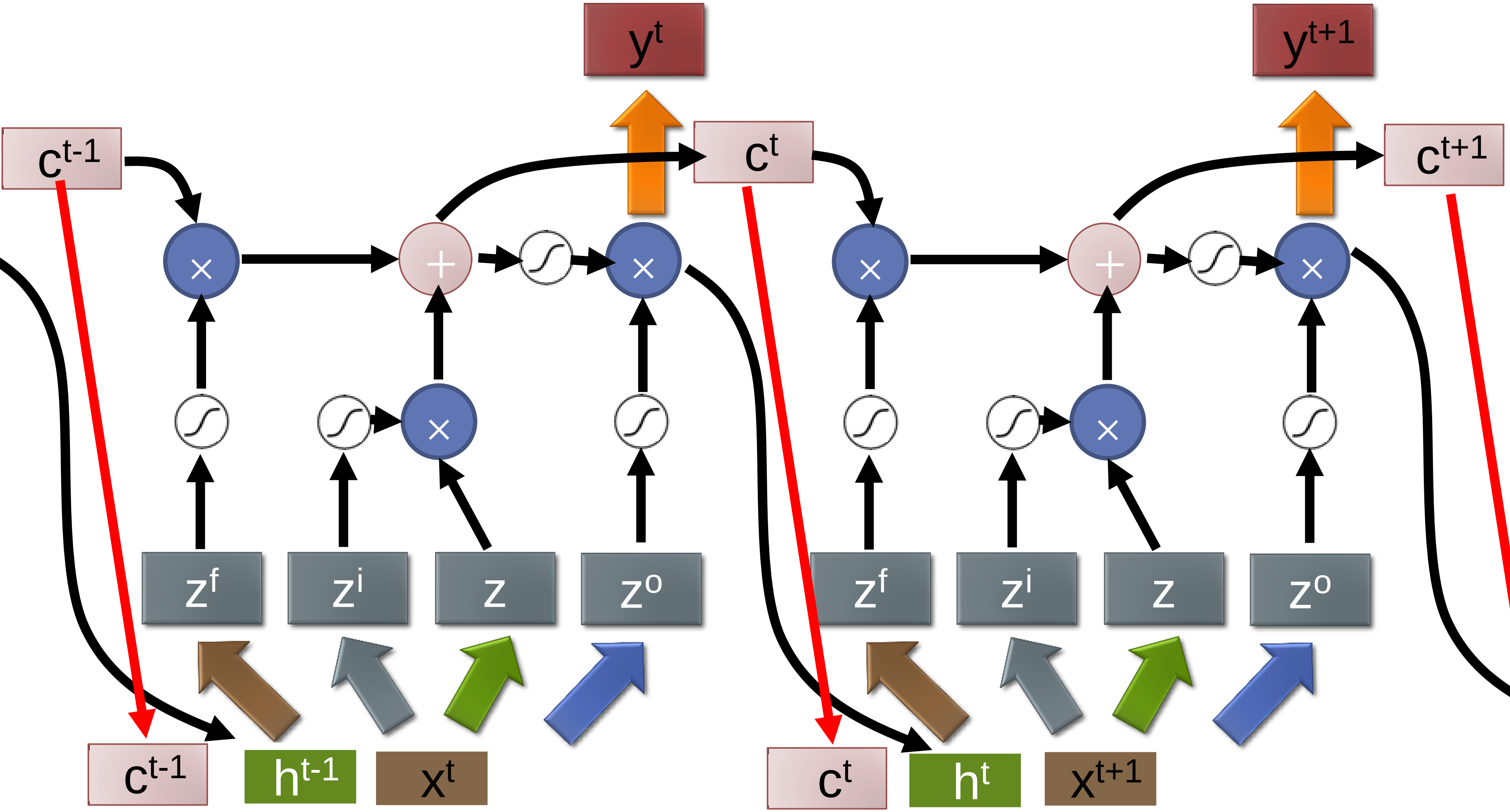





**4 times of parameters**

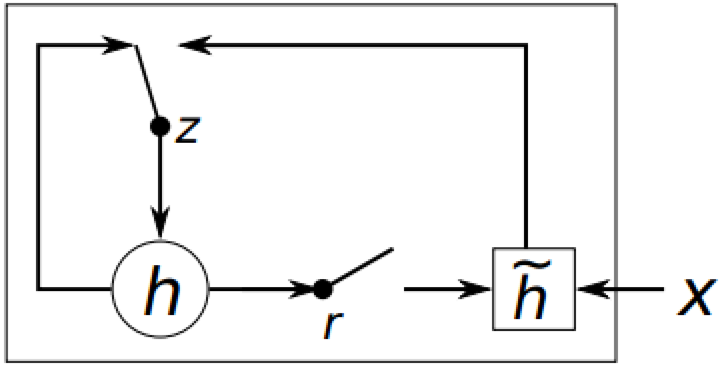
# LSTM

Extension: “peephole”



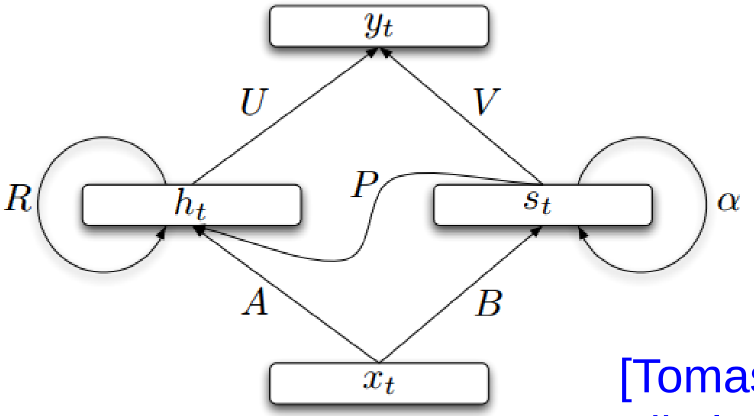
# Other Simpler Alternatives

Gated Recurrent Unit (GRU)




[Cho, EMNLP'14]

Structurally Constrained Recurrent Network (SCRN)



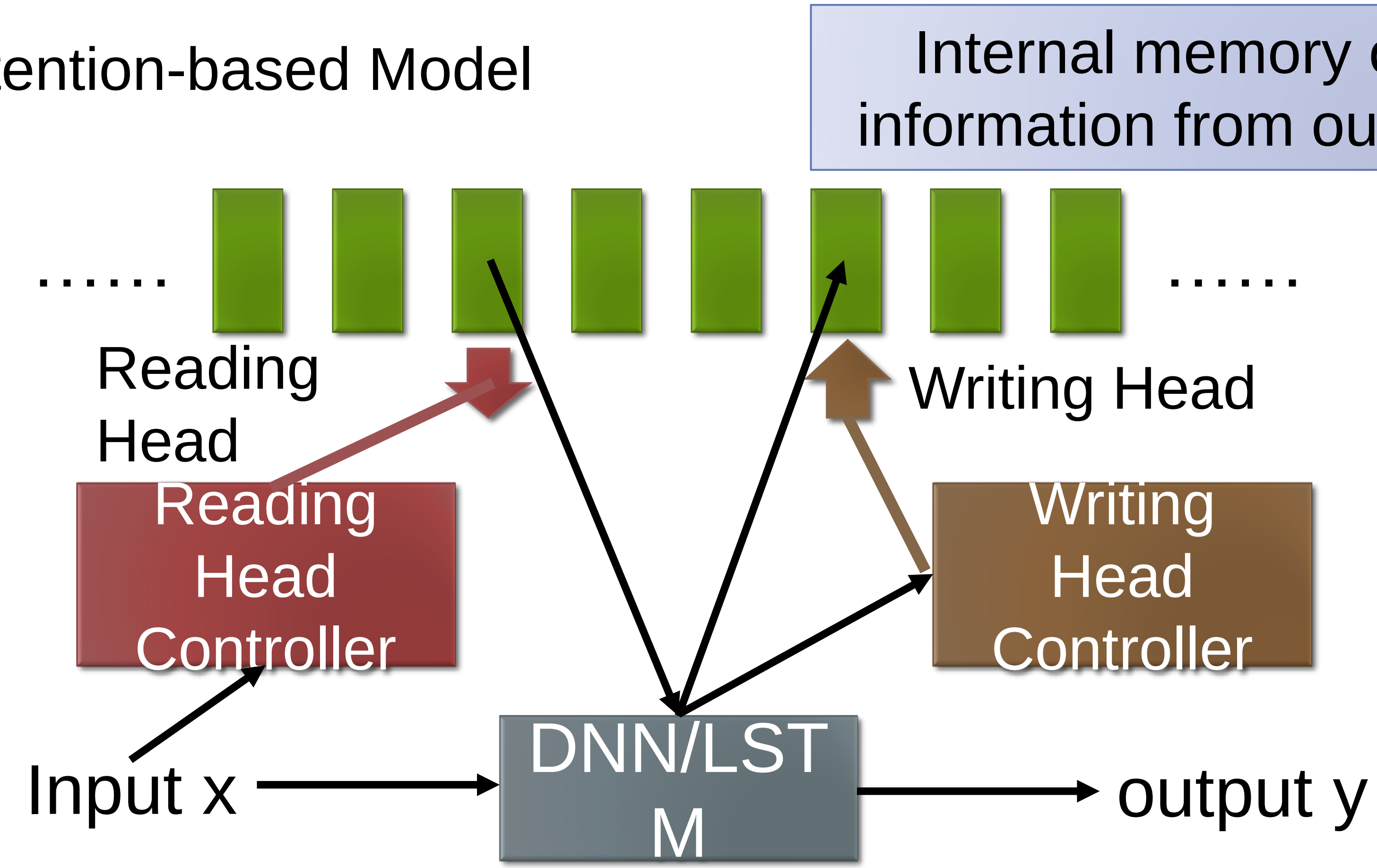

[Tomas Mikolov, ICLR'15]

Vanilla RNN Initialized with Identity matrix + ReLU activation function [Quoc V. Le, arXiv'15]

- ➢ Outperform or be comparable with LSTM in 4 different tasks

# What is the next wave?

- Attention-based Model

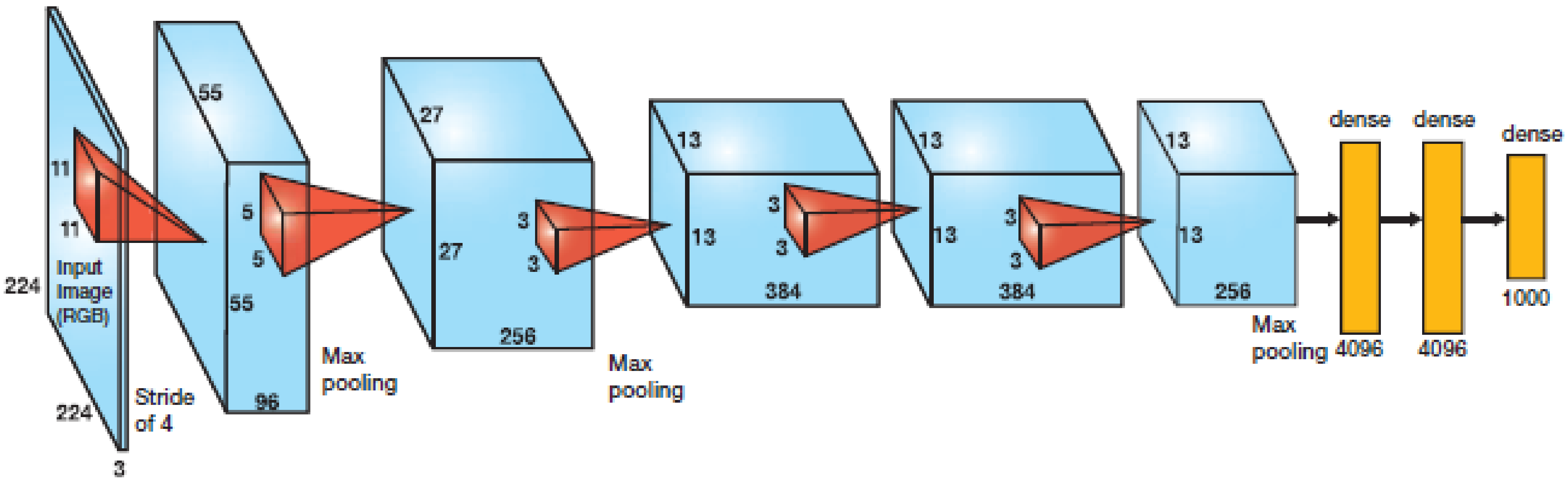



Already applied on speech recognition, caption generation, QA, visual QA

# What is the next wave?

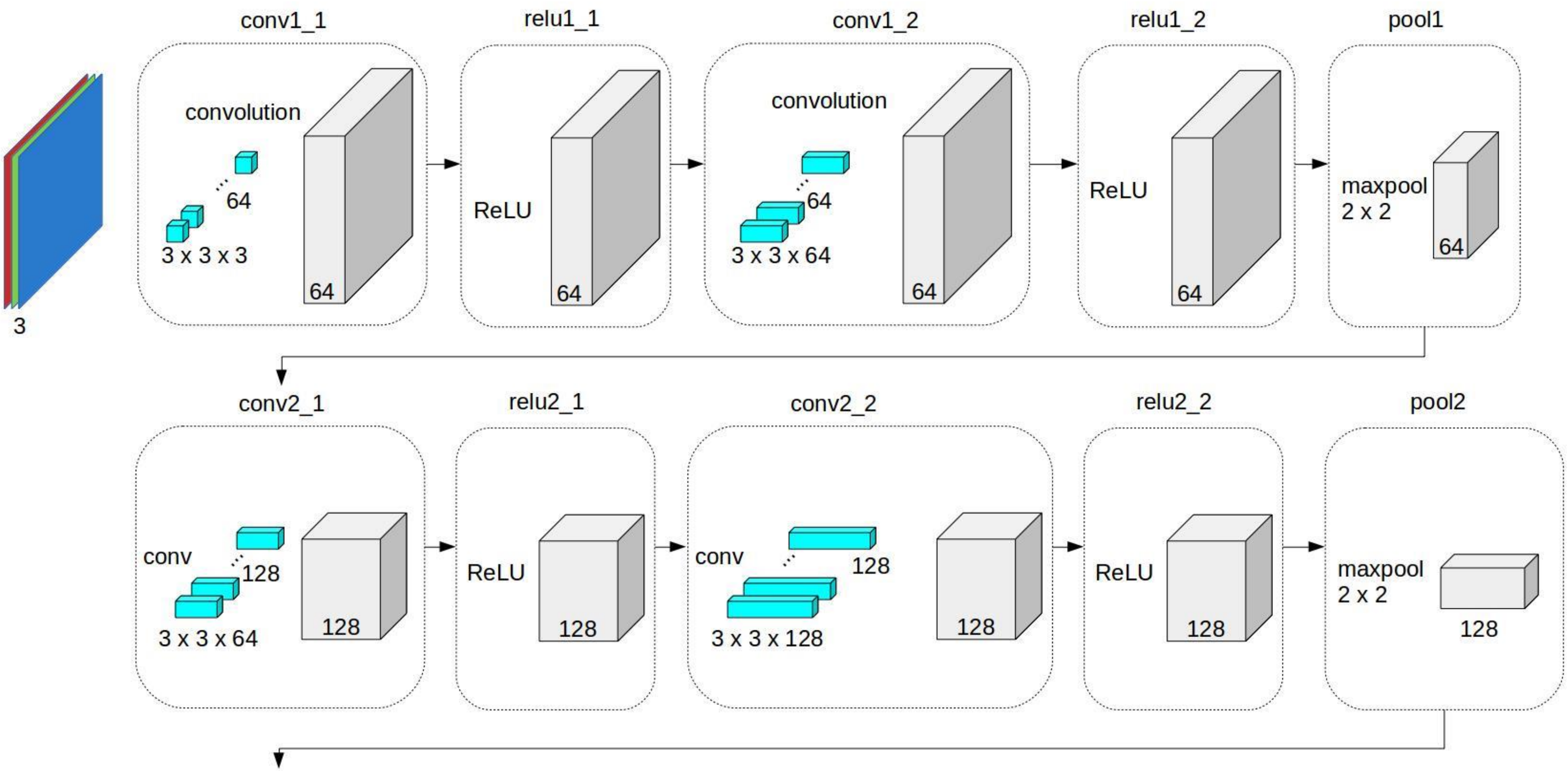
- Attention-based Model
- End-To-End Memory Networks. S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus. arXiv Pre-Print, 2015.
- Neural Turing Machines. Alex Graves, Greg Wayne, Ivo Danihelka. arXiv Pre-Print, 2014
- Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. Kumar et al. arXiv Pre-Print, 2015
- Neural Machine Translation by Jointly Learning to Align and Translate. D. Bahdanau, K. Cho, Y. Bengio; International Conference on Representation Learning 2015.
- Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Kelvin Xu et. al.. arXiv Pre-Print, 2015.
- Attention-Based Models for Speech Recognition. Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, Yoshua Bengio. arXiv Pre-Print, 2015.
- Recurrent models of visual attention. V. Mnih, N. Hees, A. Graves and K. Kavukcuoglu. In NIPS, 2014.
- A Neural Attention Model for Abstractive Sentence Summarization. A. M. Rush, S. Chopra and J. Weston. EMNLP 2015.

# AlexNet



224
224
3
11
11
Input
Image
(RGB)
Stride
of 4
55
55
96
5
5
Max
pooling
27
27
256
3
3
Max
pooling
13
13
384
3
3
13
13
384
3
3
13
13
256
Max
pooling
dense
4096
dense
4096
dense
1000

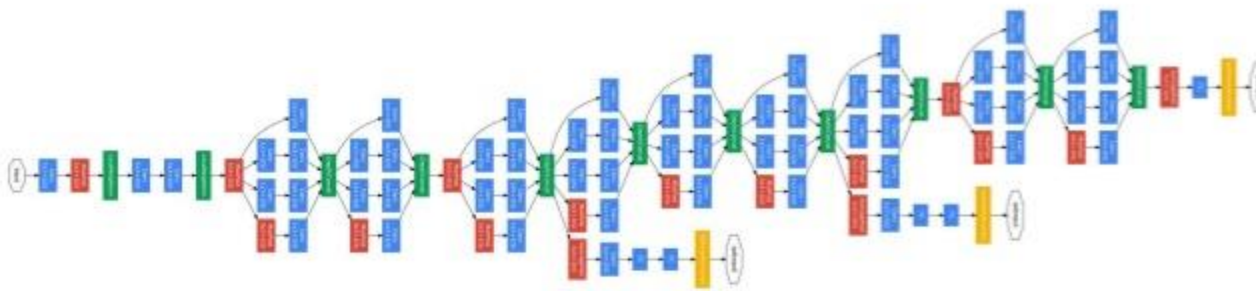
# VGG



3
conv1_1
convolution
64
3 x 3 x 3
64
relu1_1
ReLU
64
conv1_2
convolution
64
3 x 3 x 64
64
relu1_2
ReLU
64
pool1
maxpool
2 x 2
64
conv2_1
conv
128
3 x 3 x 64
128
relu2_1
ReLU
128
conv2_2
conv
128
3 x 3 x 128
128
relu2_2
ReLU
128
pool2
maxpool
2 x 2
128

# GoogleNet

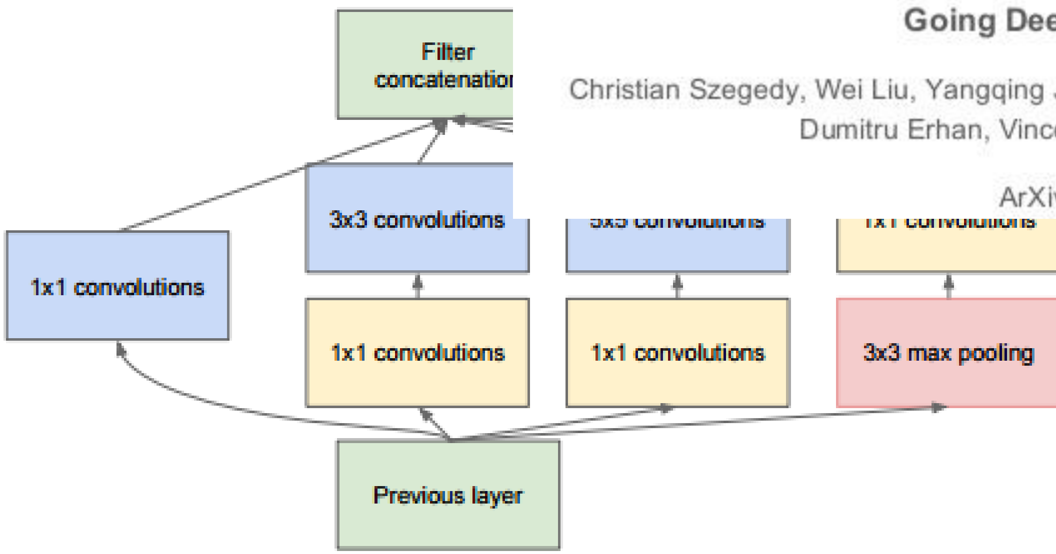
The Inception Architecture (GoogLeNet, 2014)



**Going Deeper with Convolutions**

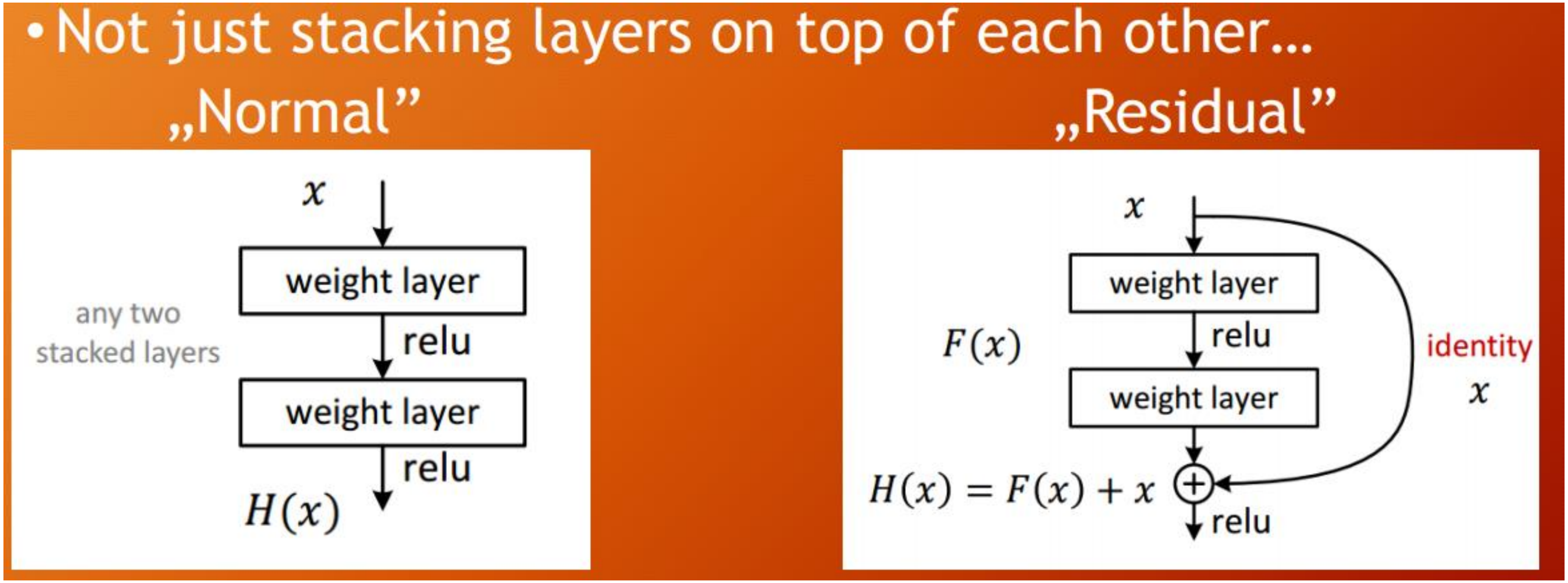
Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

ArXiv 2014, CVPR 2015




(b) Inception module with dimension reductions

# ResNet



• Not just stacking layers on top of each other...
„Normal”
„Residual”
$x$
weight layer
relu
weight layer
relu
any two stacked layers
$H(x)$
$x$
weight layer
relu
weight layer
$F(x)$
identity
$x$
$H(x) = F(x) + x$
relu

# R-FCN

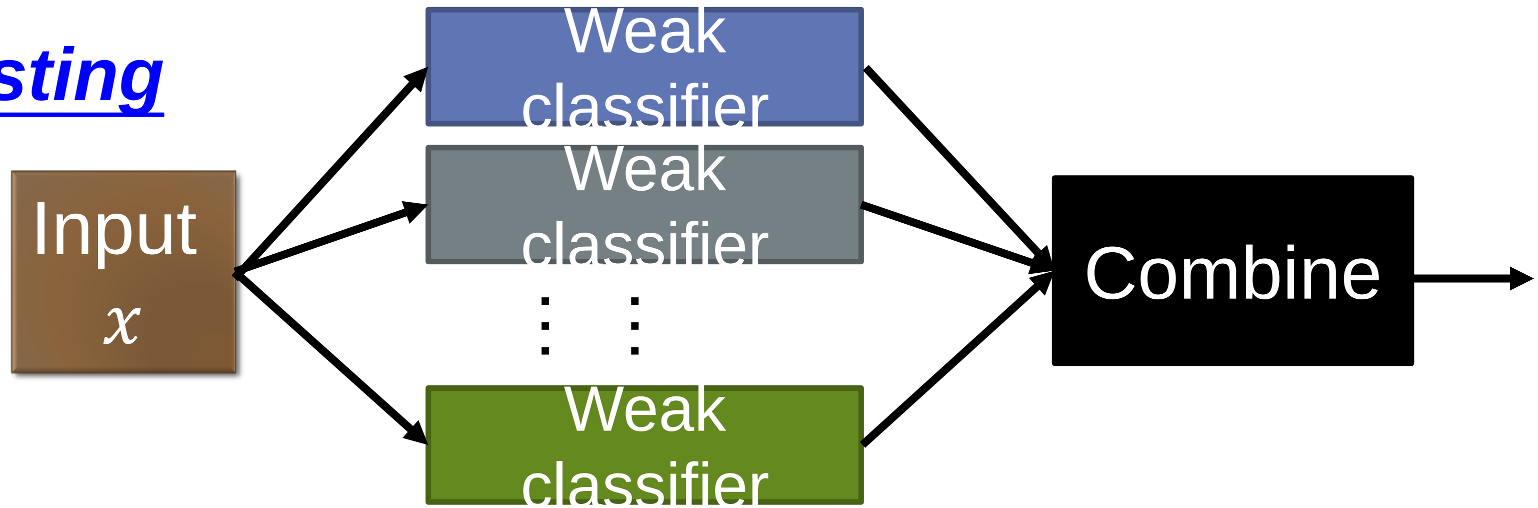
- Object Detection by Region based fully connected CNN
  - Regression to boundary box coordinates + object classification inside the bounding box

- Networks are rarely trained from scratch. Pre-trained weights are often available for popular networks. Proceed with fine tuning (transform learning)

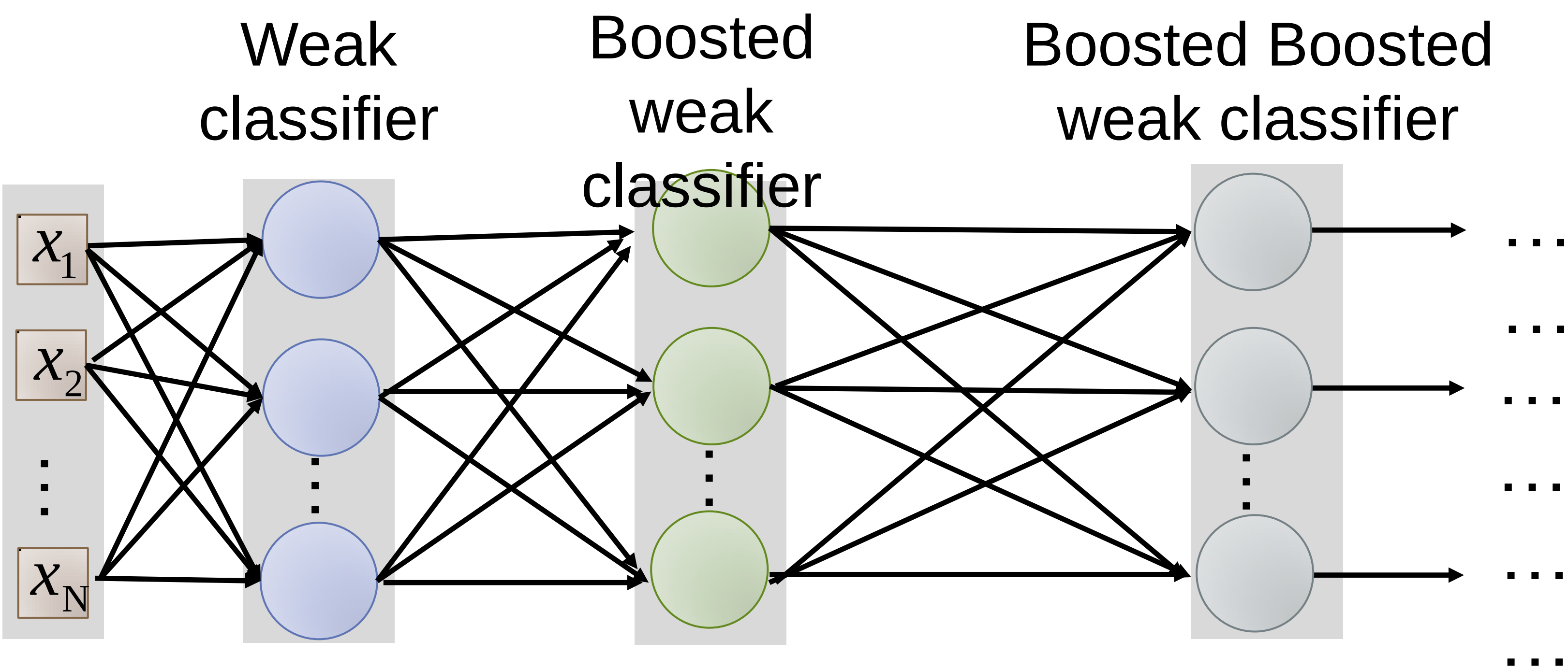
THANK YOU

FOR YOUR ATTENTION!

## Boosting

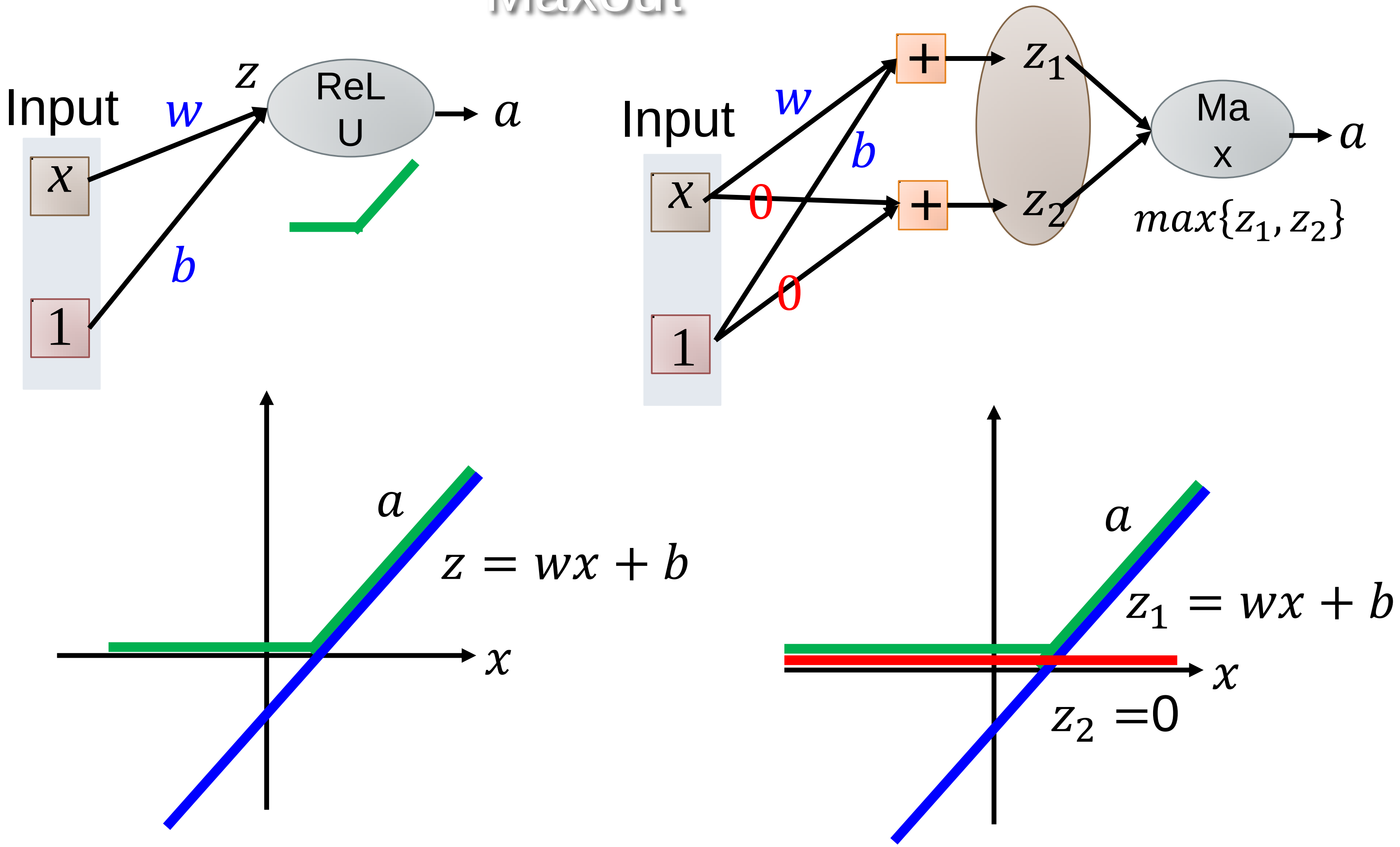



## Deep Learning



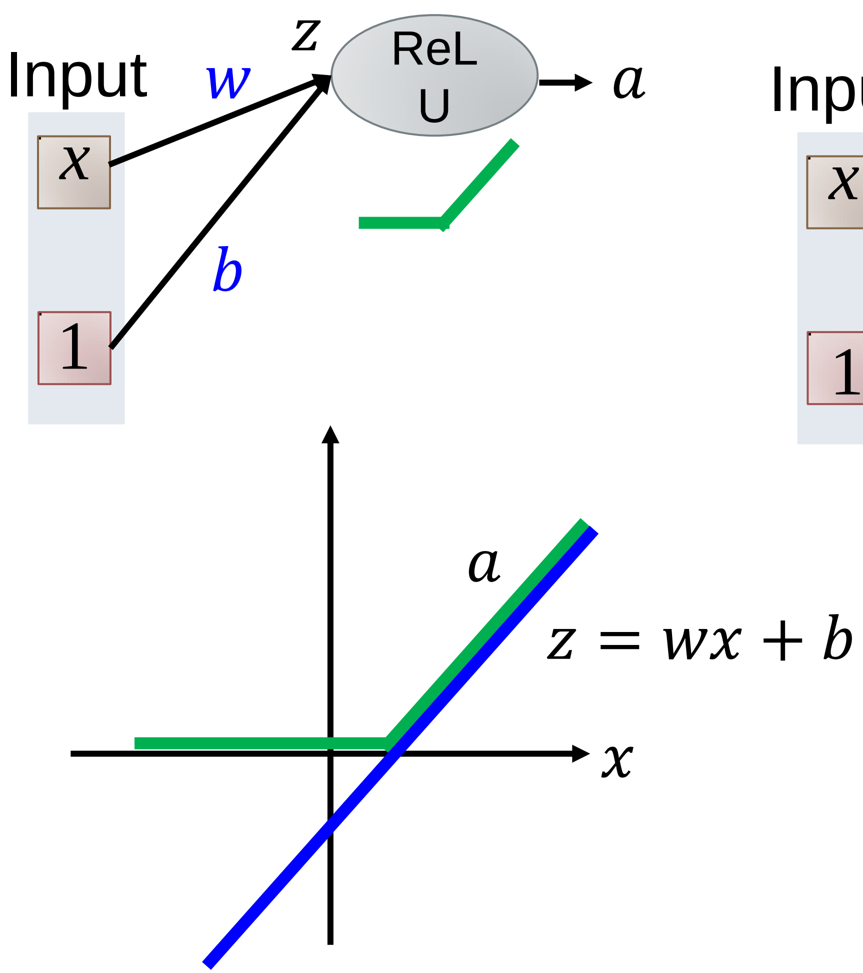
# Maxout

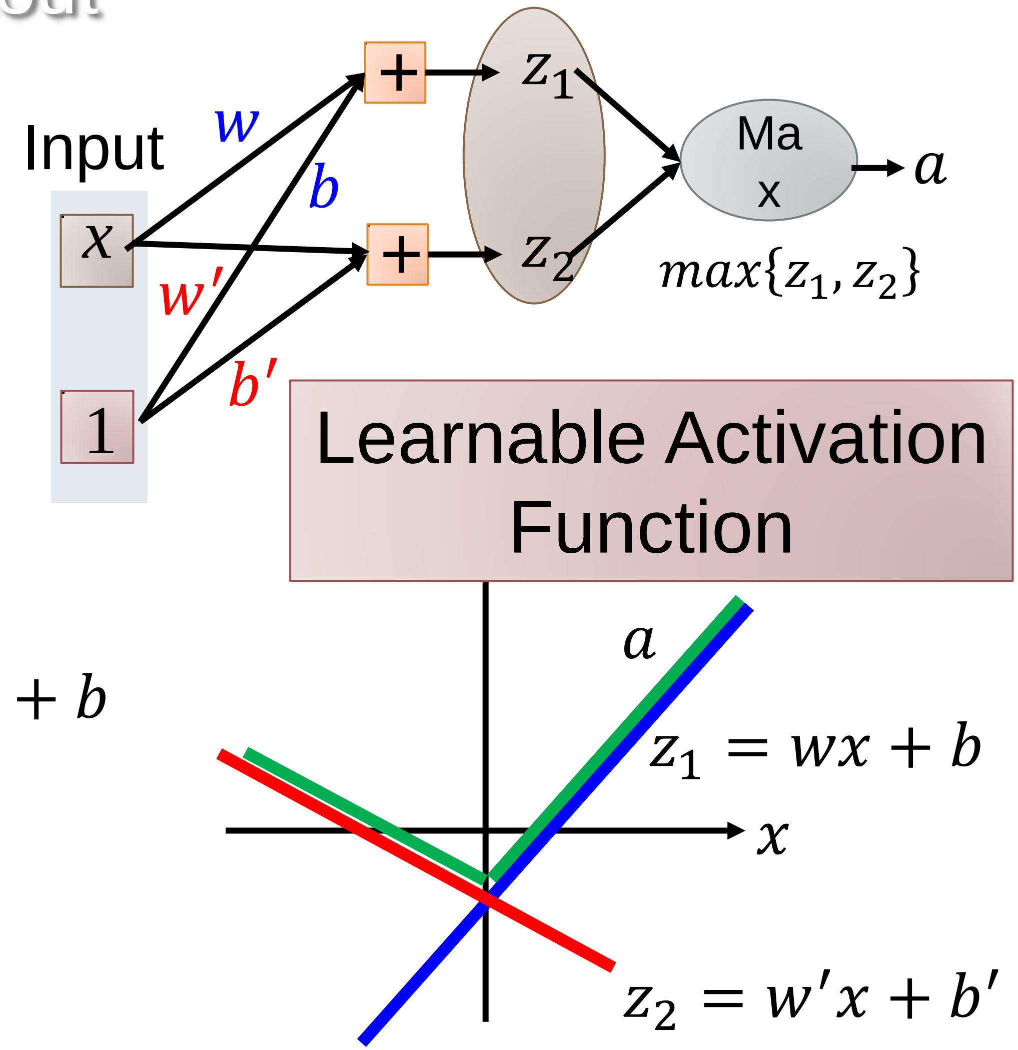
ReLU is a special cases of Maxout



# Maxout

ReLU is a special cases of Maxout




Learnable Activation Function