Convolutional Neural Networks (CNN)

LIONEL PREVOST
HEAD OF LEARNING, DATA & ROBOTICS LAB – ESIEA
lionel.prevost@esiea.fr

Outline

- Classical MLP’s drawbacks for complex tasks
- Convolution filters
  - 1D/2D convolution process in brief
  - Basic filters for image processing
  - High level filters for object detection
- Convolutional Neural networks
  - “Novelties”: convolution, pooling, activation function
  - Building and training a CNN using KERAS
- Improving generalization: Batch Normalization & Drop Out
- Pre-trained models

MLP for image analysis

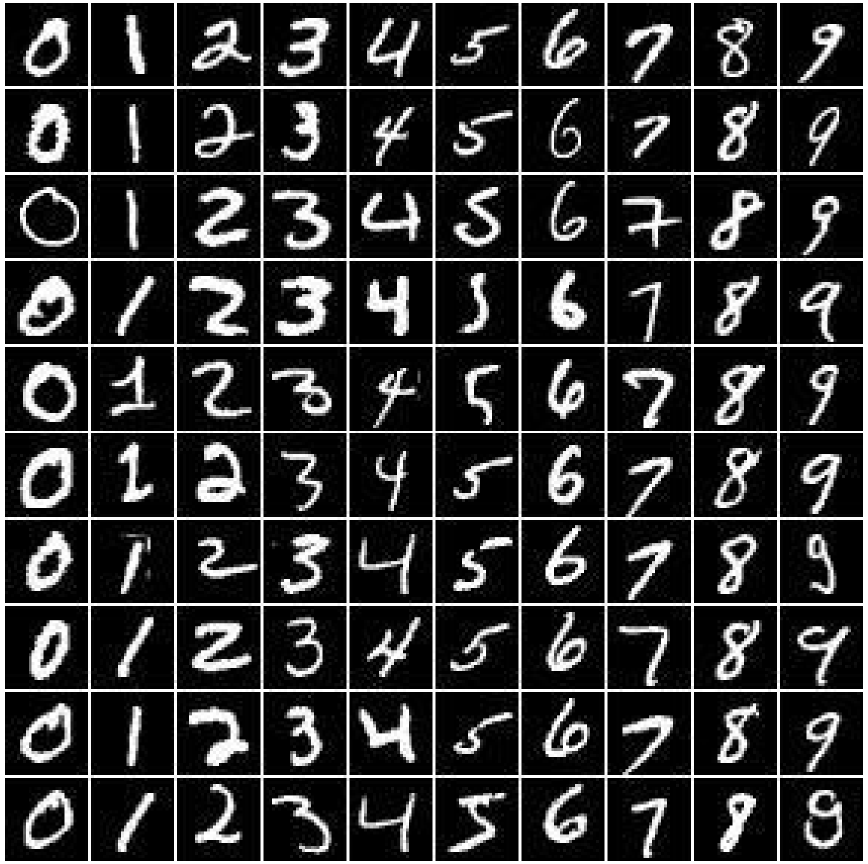
- **Optical Character Recognition (OCR)**

Benchmark: MNIST database

Image size: **28x28=784** pixels

Training set: 60,000 examples

Test set: 10,000 examples.



(...)

- **Object recognition**

Benchmark: CIFAR-10

Image size: **32x32=1024** pixels

Training set: 50,000 examples

Test set: 10,000 examples.

10 classes

CIFAR-100 (100 classes)

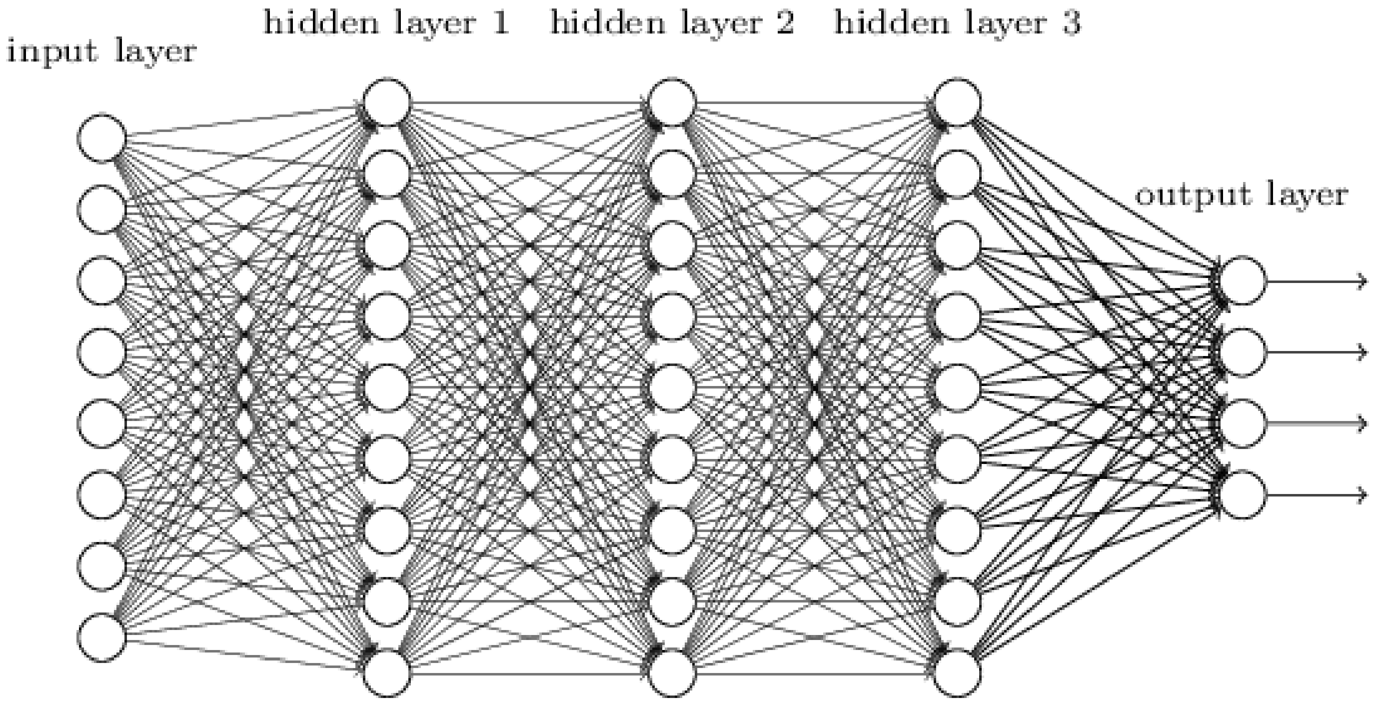
ImageNet (1.2M images, 1000 classes)

(...)

- **MLP's drawbacks**

Many input and output

→ **Too many parameters to learn**



From this fully connected model, do we really need all the edges?

Can some of these be shared?

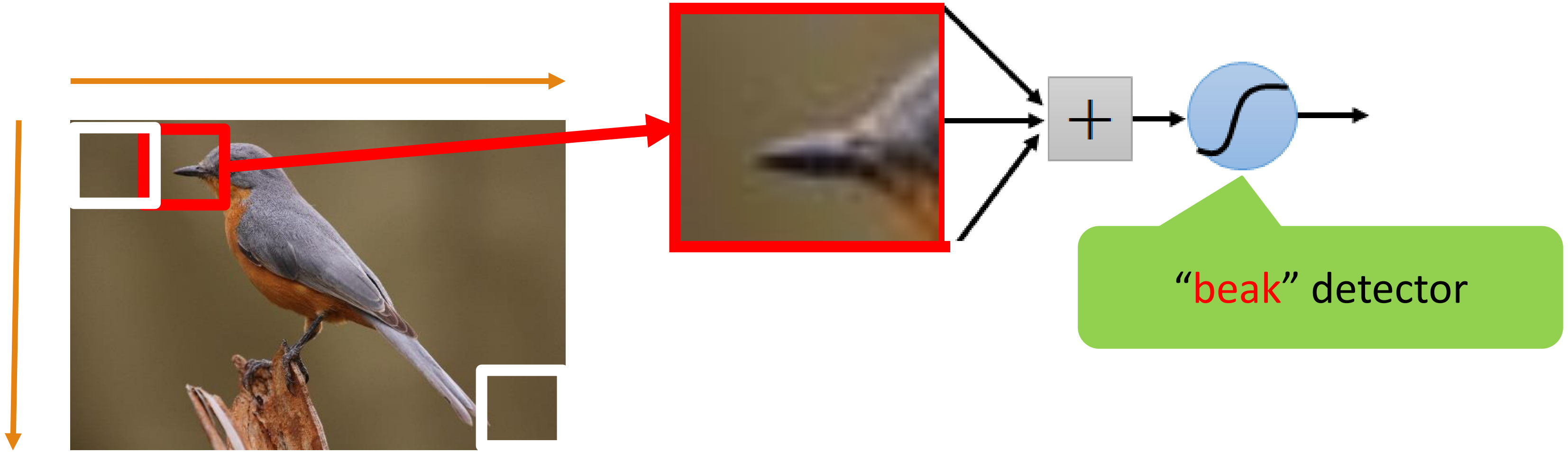
A little cue

Some patterns are much smaller than the whole image

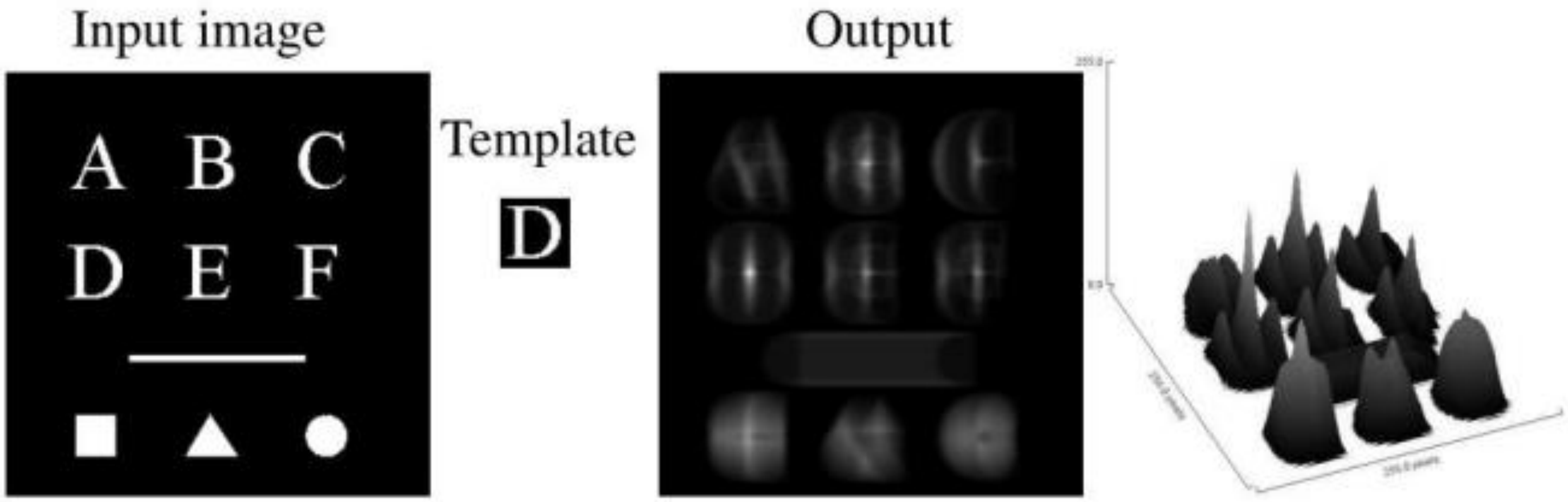
This detector can be learned on "beak" images

(...)

The detector moves through the whole image and output +1 when « seeing » a beak



Back to the 90’s: template matching



The higher the output is, the better the match

Outline

- Classical MLP's drawbacks for complex tasks
- Convolution filters
  - 1D/2D convolution process in brief
  - Basic filters for image processing
  - High level filters for object detection
- Convolutional Neural networks
  - "Novelties": convolution, pooling, activation function
  - Building and training a CNN using KERAS
- Improving generalization: Batch Normalization & Drop Out
- Pre-trained models

Solution: convolution product

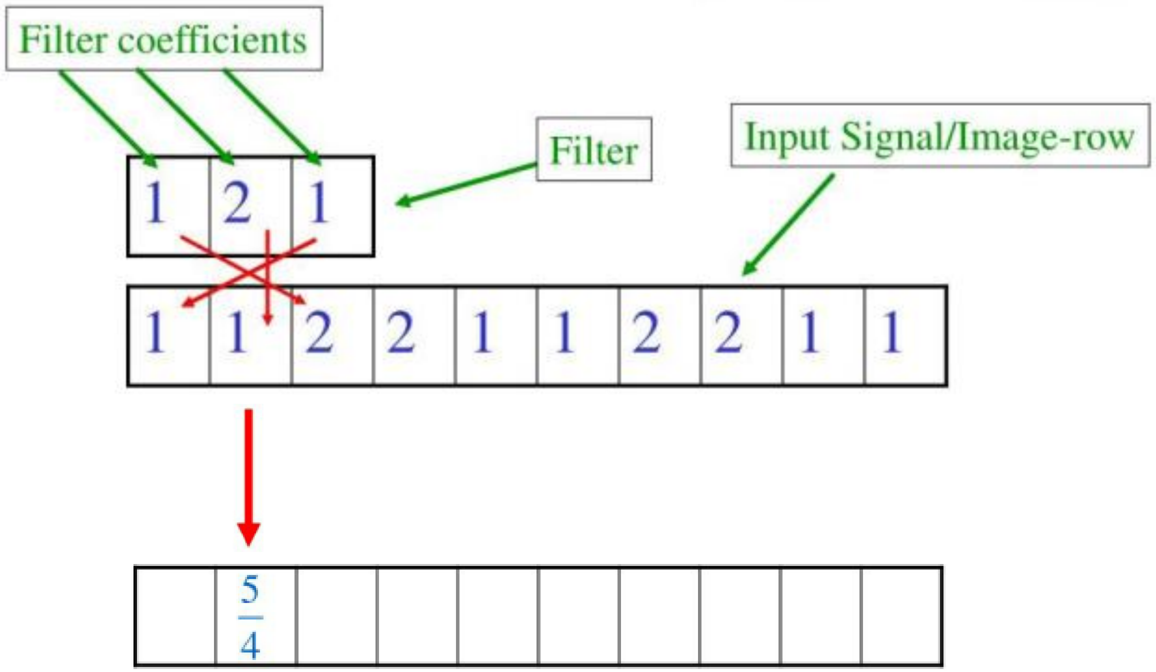
Quiet beginning in dimension 1 (see 3rd year's signal processing course)

Filter = [1, 2, 1]

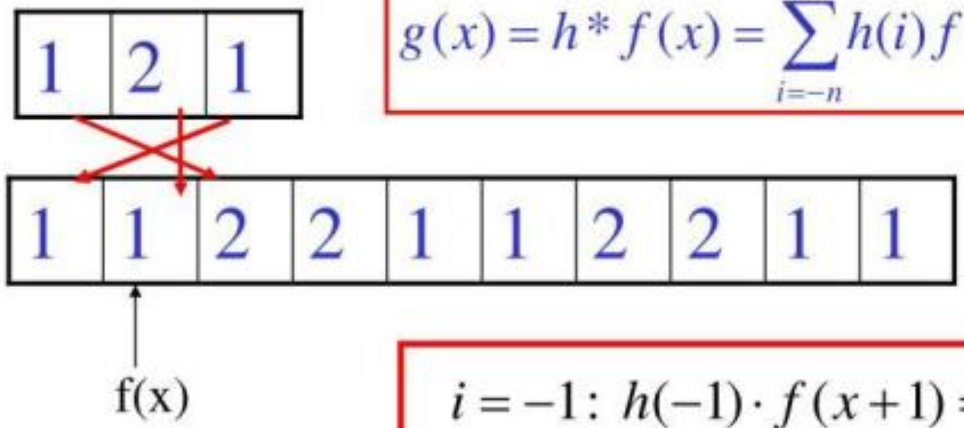
Sum(Filter)=4

Normalized filter output

(invariant to filter size)



Math of convolution



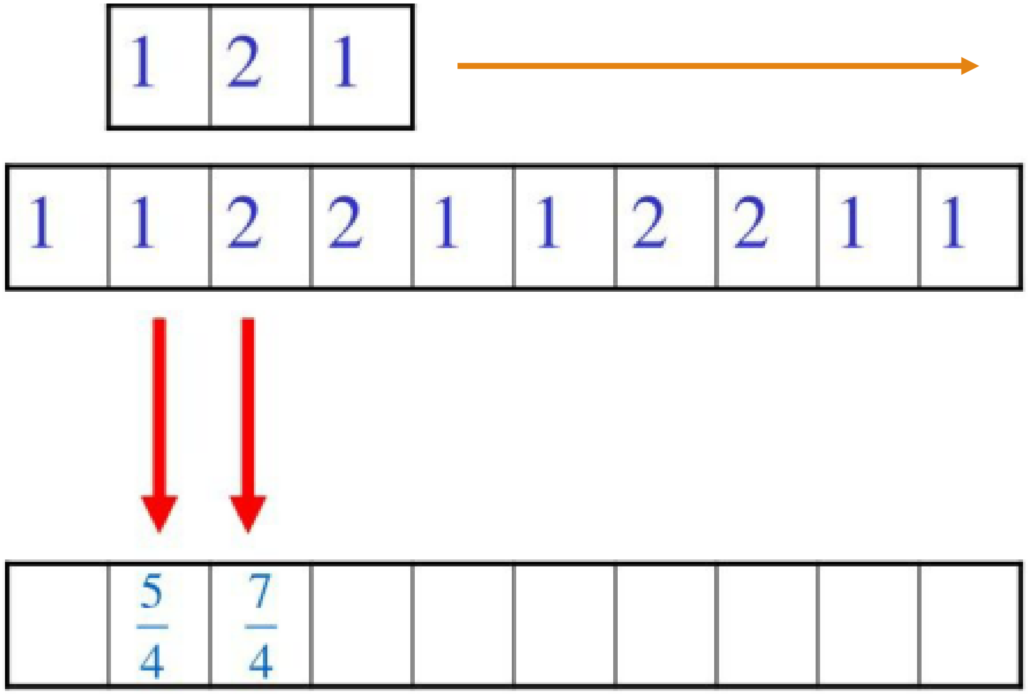
$$g(x) = h * f(x) = \sum_{i=-n}^{n} h(i) f(x-i)$$

$$i = -1: \; h(-1) \cdot f(x+1) = 1 \cdot 2 = 2$$
$$i = 0: \quad h(0) \cdot f(x) = 2 \cdot 1 = 2$$
$$i = 1: \quad h(1) \cdot f(x-1) = 1 \cdot 1 = 1$$
$$g(x) = 2 + 2 + 1 = 5$$
$$\text{Normalise} \; : \; g(x) = 5/4$$

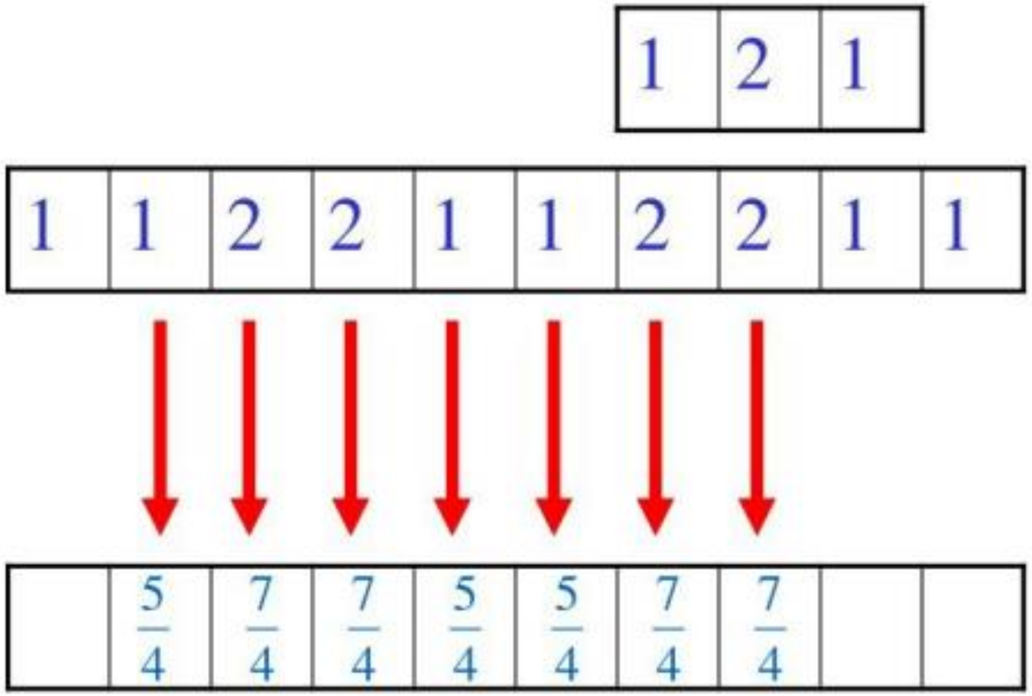
The filter is <u>symmetric</u>:

The convolution is just a dot product!

(...)


1 2 1
1 1 2 2 1 1 2 2 1 1
5/4 7/4

(...)



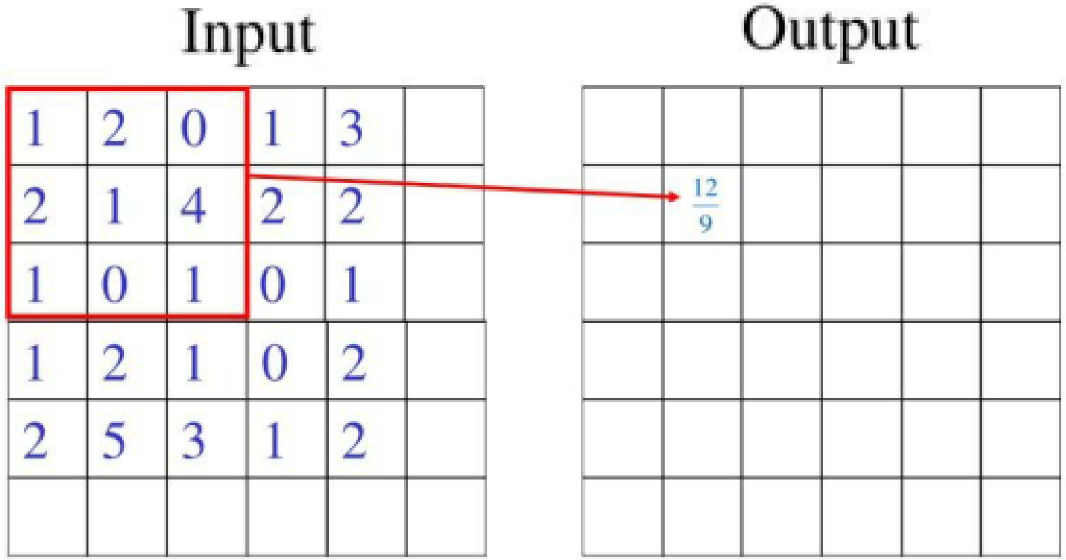
1 2 1
1 1 2 2 1 1 2 2 1 1
5/4 7/4 7/4 5/4 5/4 7/4 7/4

Convolution on images

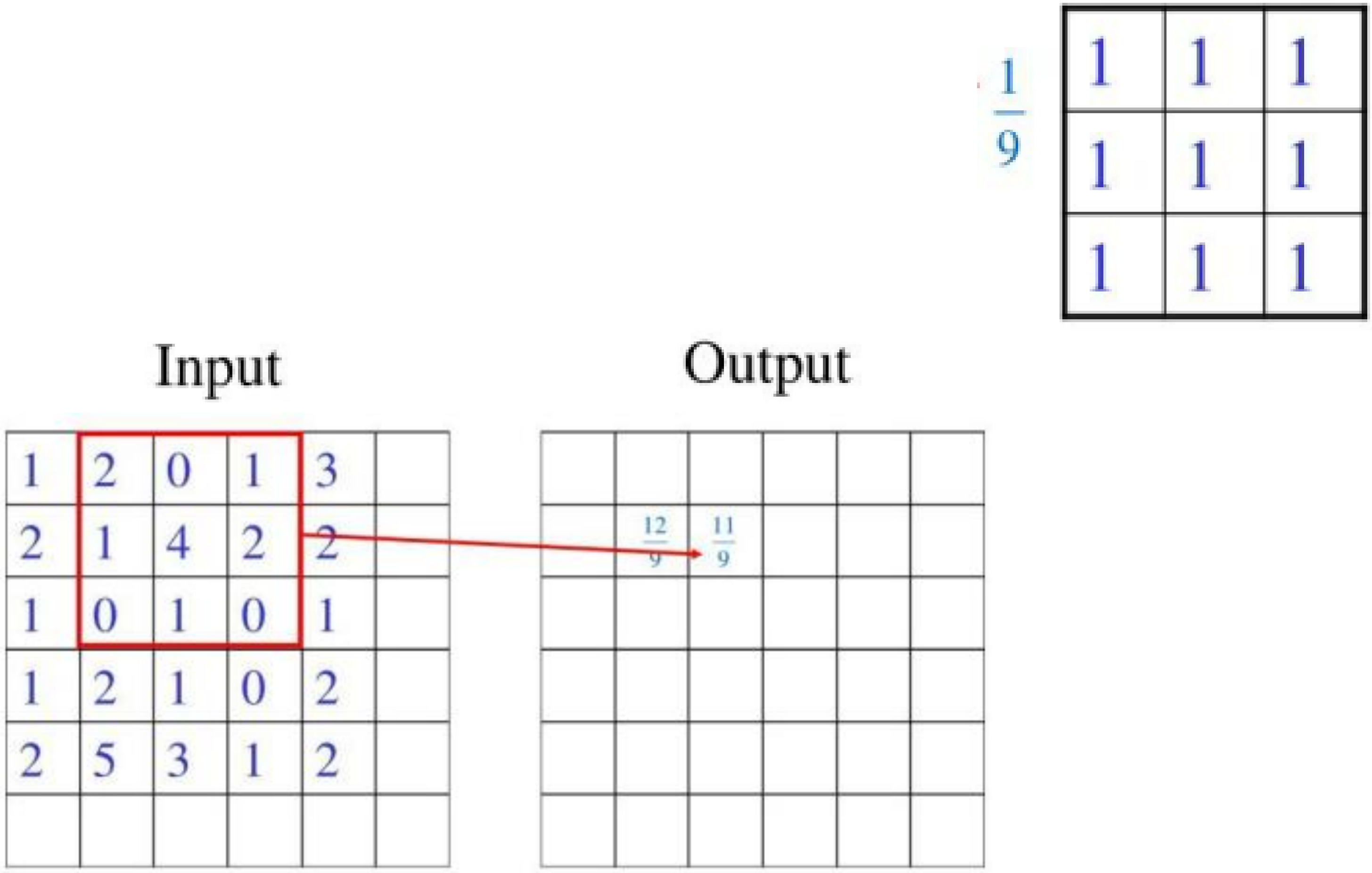
The filter is now 2D

Size : **3x3**, 5x5, 7x7 …

normalized filter $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$



Step 2



Final step

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Input

1	2	0	1	3	
2	1	4	2	2	
1	0	1	0	1	
1	2	1	0	2	
2	5	3	1	2	

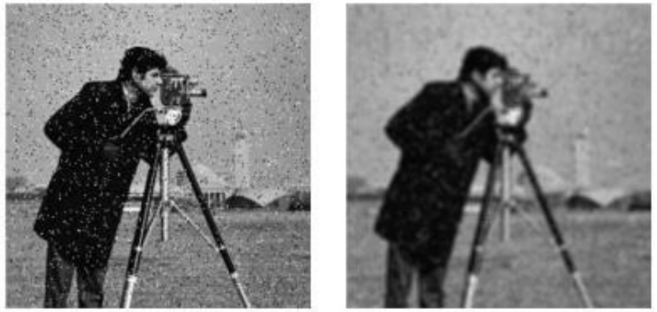
Output

	$\frac{12}{9}$	$\frac{11}{9}$	$\frac{14}{9}$		
	$\frac{13}{9}$	$\frac{11}{9}$	$\frac{13}{9}$		
	$\frac{16}{9}$	$\frac{12}{9}$	$\frac{11}{9}$		

Application to image processing

- **“basic” filters**
  - image blurring

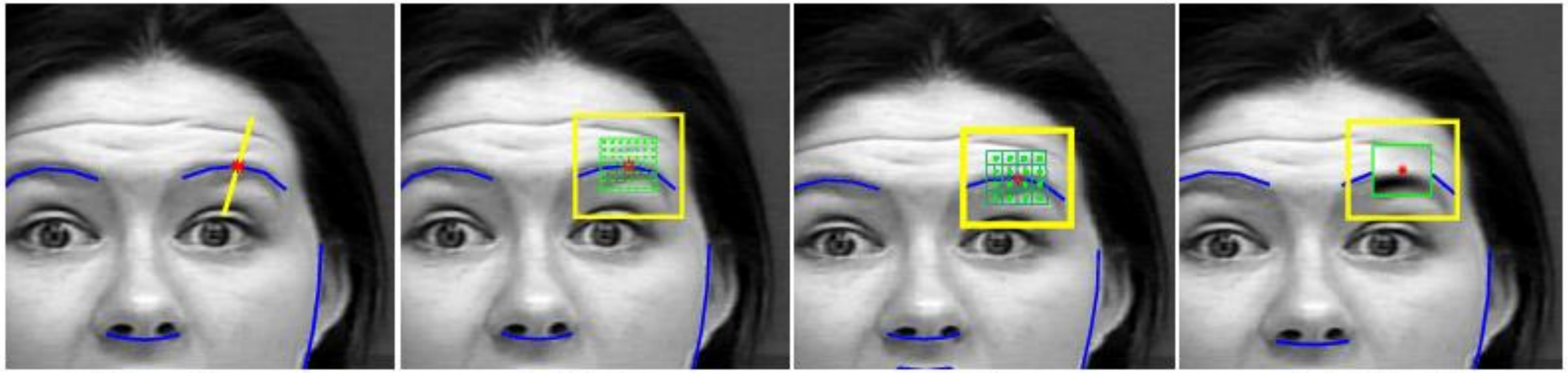
  - noise removal



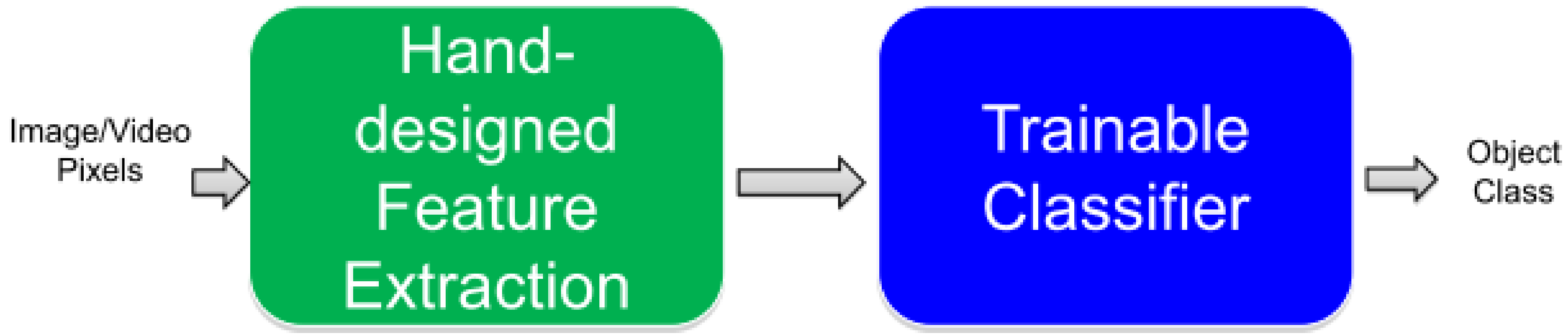
  - **edge detection**

(…)

- **Complex operators for object detection**
  - Histogram of oriented Gradient (HoG)
  - Scale Invariant Feature Transform (SIFT)
  - …



Recognition pipeline

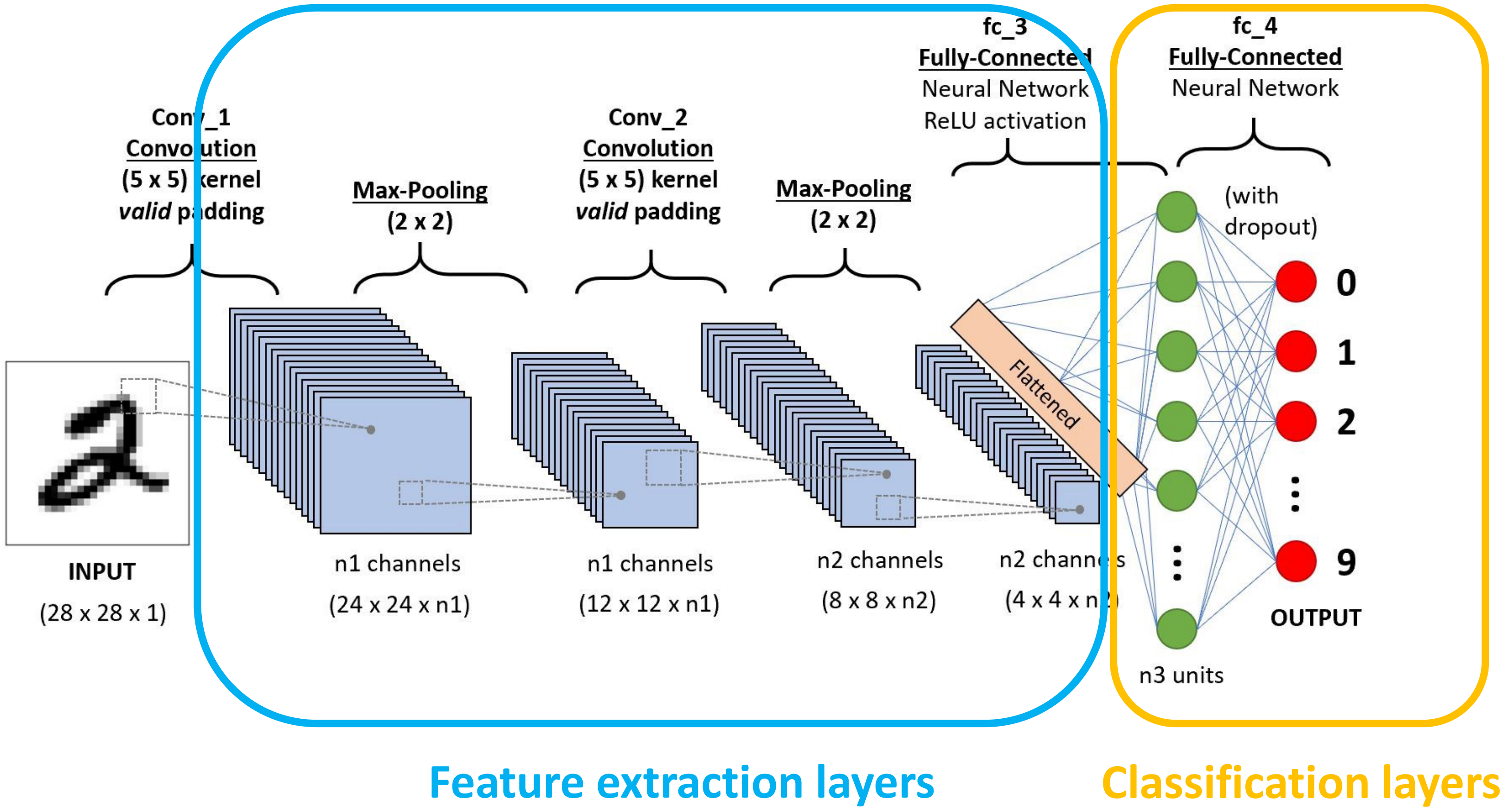


- Feature are **not** learned and not specific enough for complex recognition problems
- 2-step solutions: feature extraction and classifier training

Outline

- Classical MLP's drawbacks for complex tasks
- Convolution filters
  - 1D/2D convolution process in brief
  - Basic filters for image processing
  - High level filters for object detection
- Convolutional Neural networks
  - "Novelties": convolution, pooling, activation function
  - Building and training a CNN using KERAS
- Improving generalization: Batch Normalization & Drop Out
- Pre-trained models

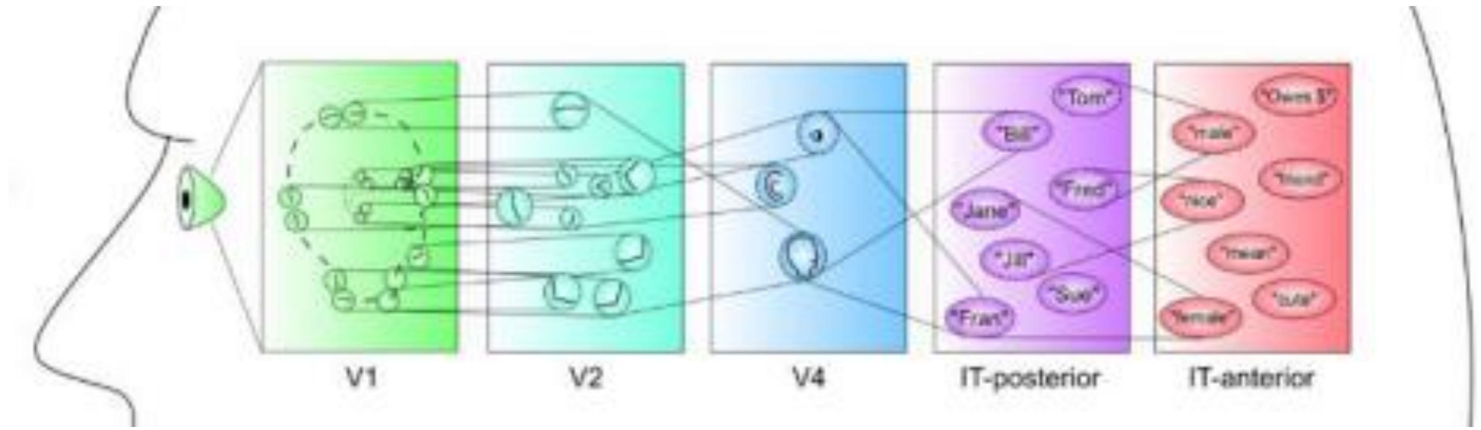
Solution: “End to end” learning



Biological inspiration

Within the visual cortex, complex functional responses generated by «complex cells» are constructed from more simplistic responses from «simple cells».

Simple cells would respond to oriented edges, etc, while complex cells will also respond to oriented edges but with a degree of spatial invariance.



(...)

The architecture of deep convolutional neural networks was inspired by:

- Local connections
- Layering to improve abstraction level (from low to high)
- Spatial invariance

These abstractions are thus invariant to size, contrast, rotation, orientation, ..)

Architecture

There are four main operations (and three novelties) in the CNN :

1. Convolution

2. Pooling (Sub-Sampling)

3. Non Linearity (ReLU)

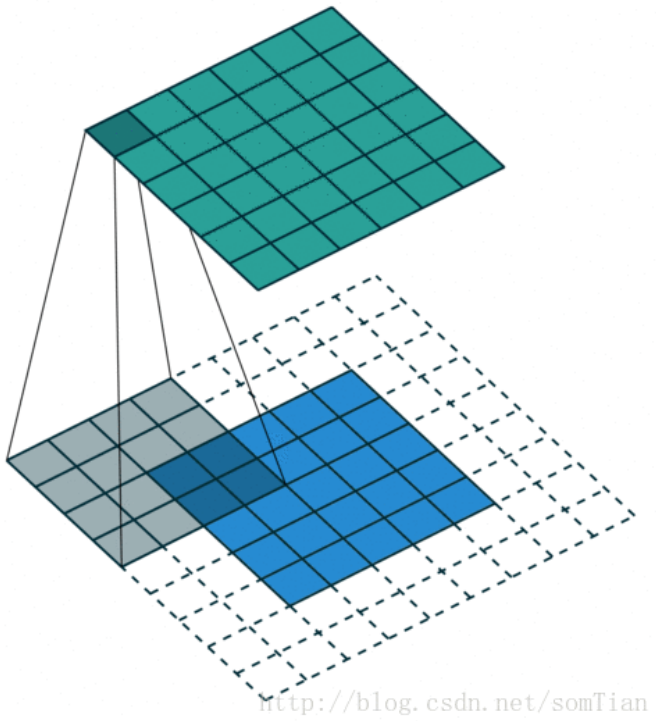
4. Classification (Fully Connected Layer)

Convolution layer

It includes **several** convolution filters.

Each filter is trained to detect a **specific** pattern.

Main parameters:
- Filter size
- Stride



(...)

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).

Stride

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

(...)

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

Dot product

3 -3

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

Filter 1

Filter 1 output

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

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

Filter 1
diagonal

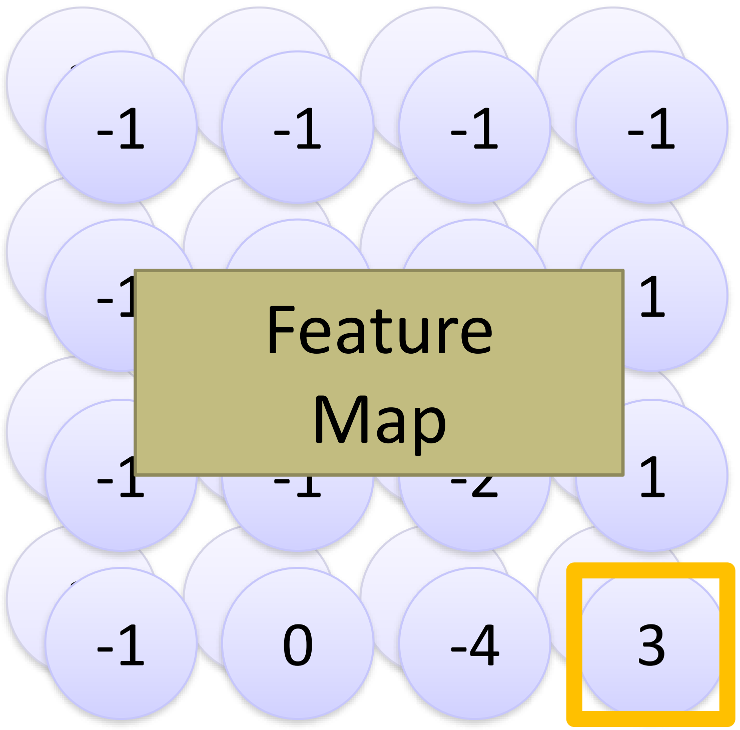
Global output

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



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

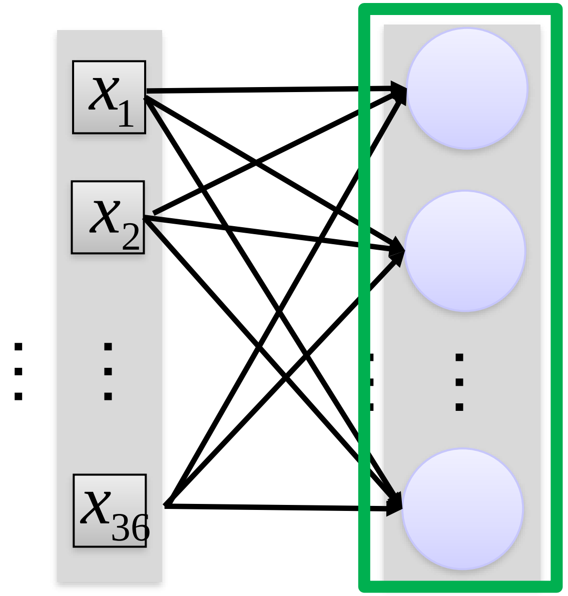
Filter 2
vertical

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

Why is it better than full connections?

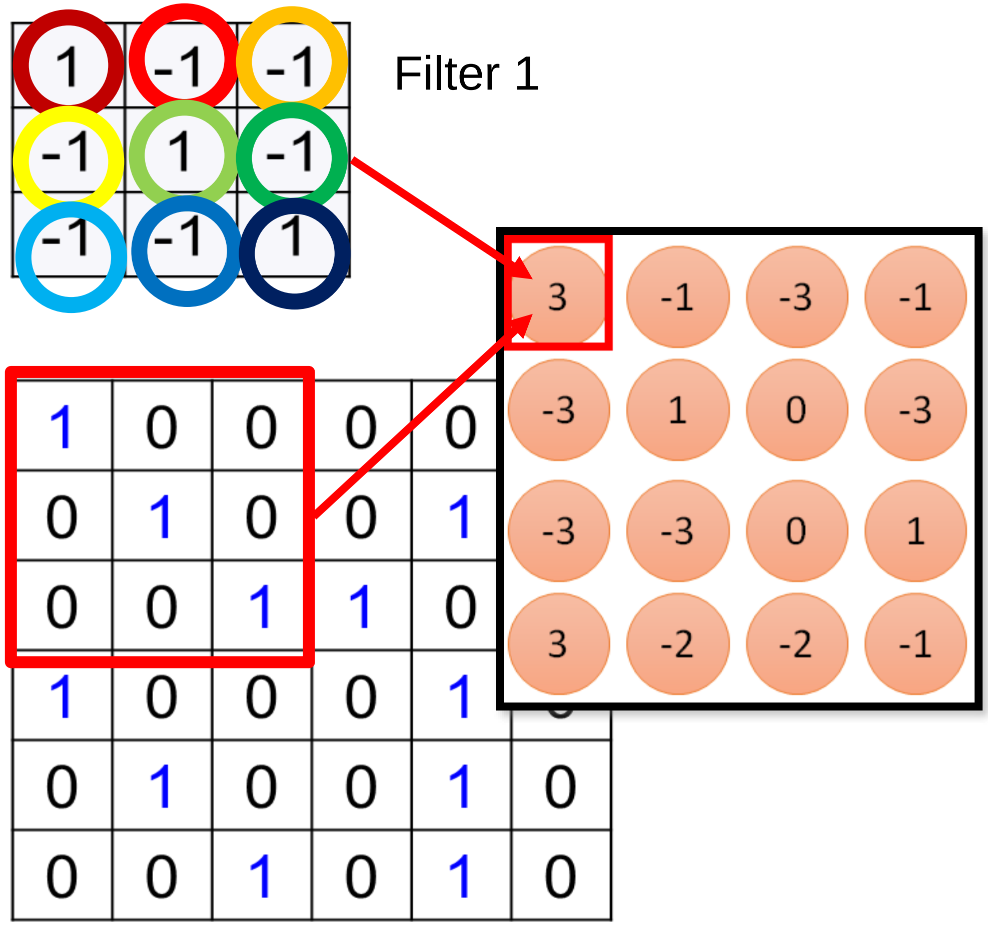
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



For 10 neurons: 360 weights

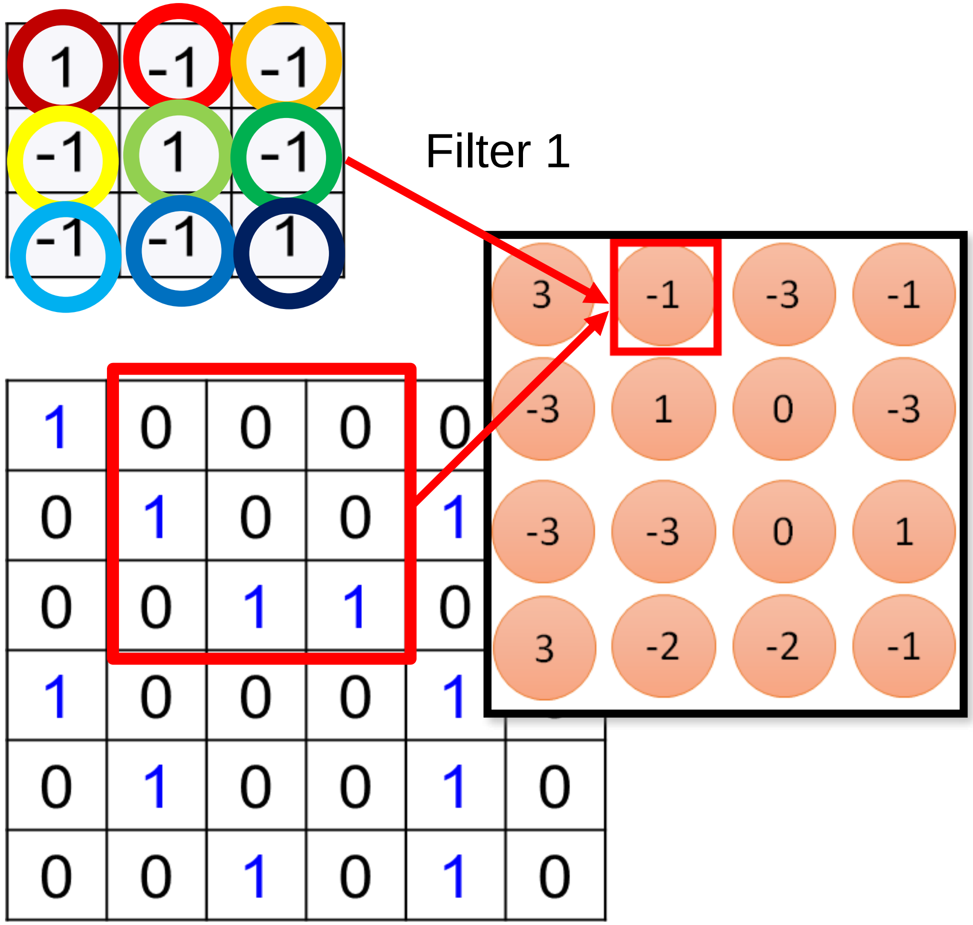
Local connectivity



Only connect to 9 inputs, not fully connected

Fewer parameters

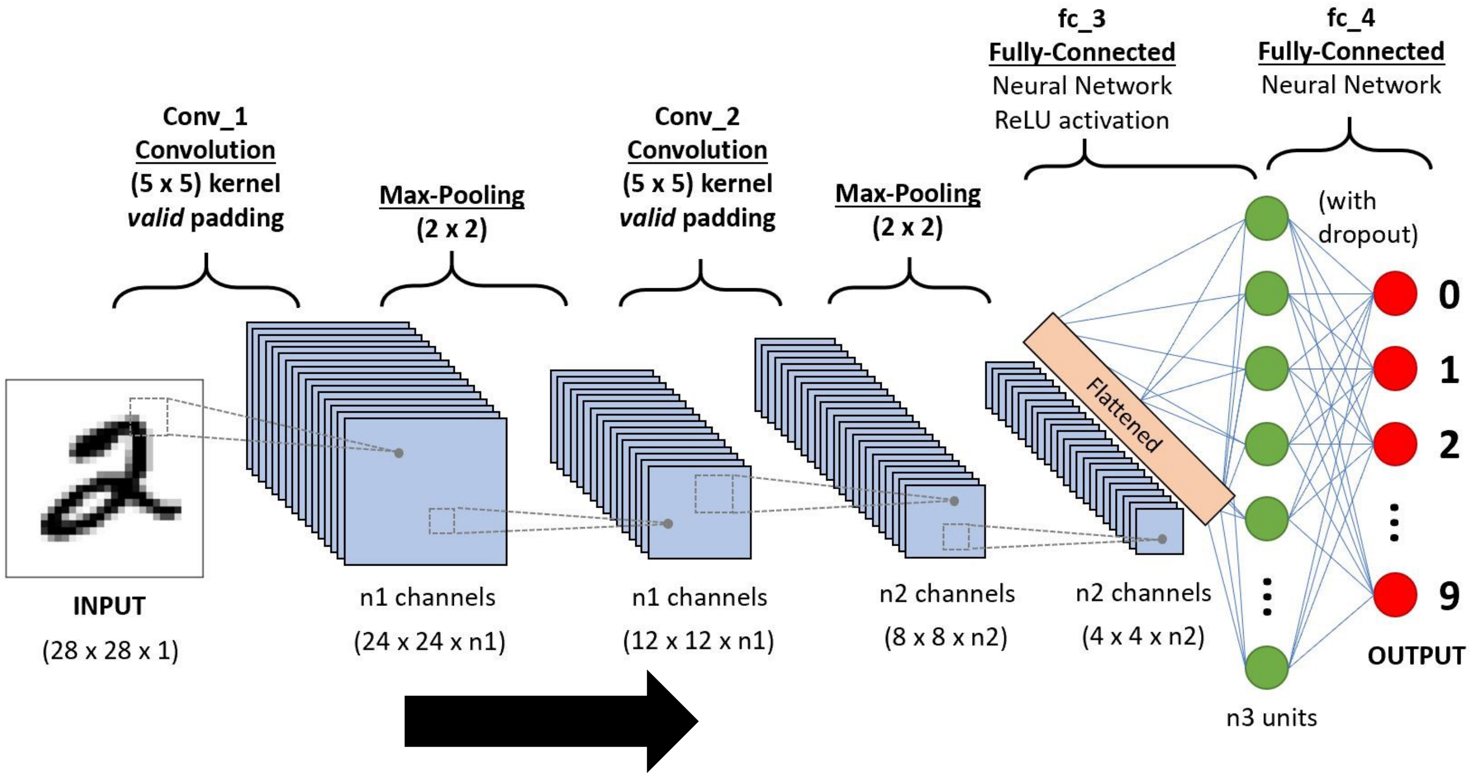
Shared weights



Even fewer parameters!

For 10 filters : 90 weights

Sub-sampling



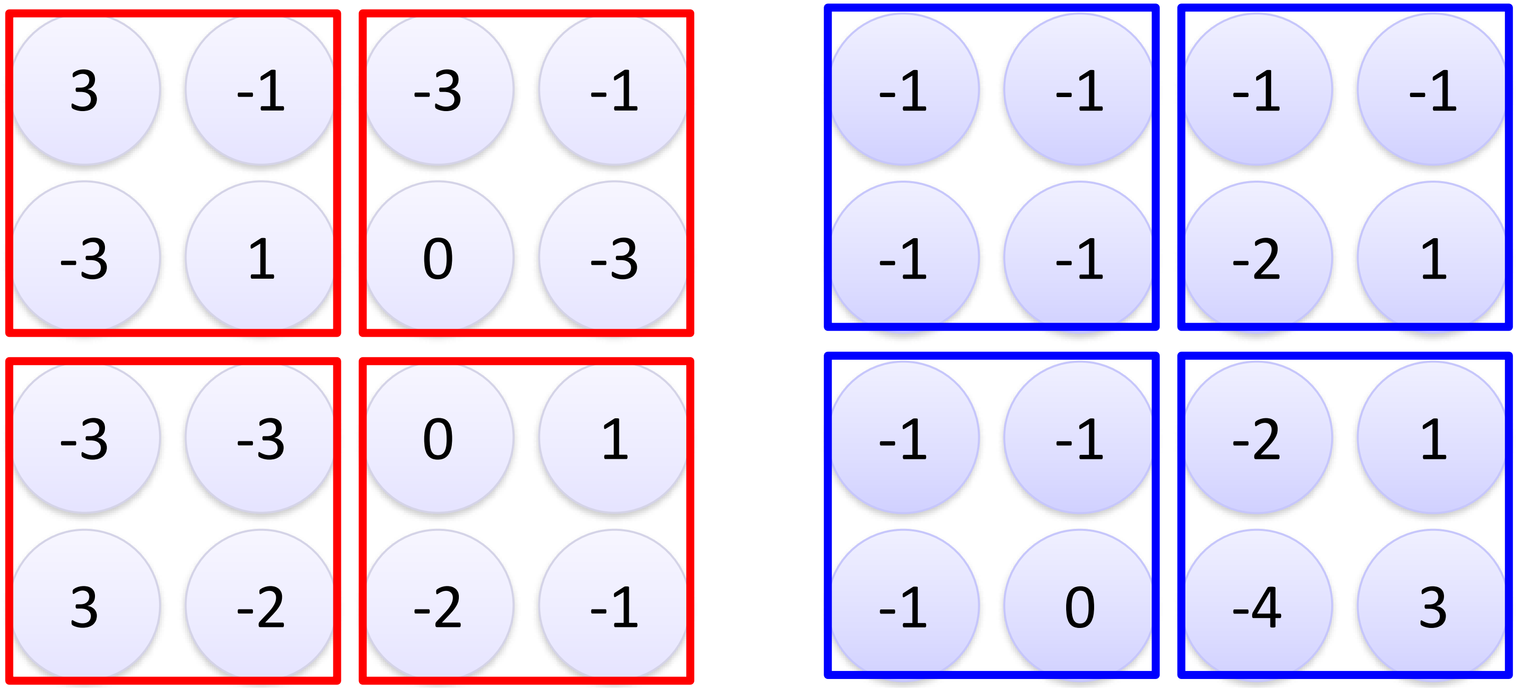
Subsampling: reduce the number of parameters

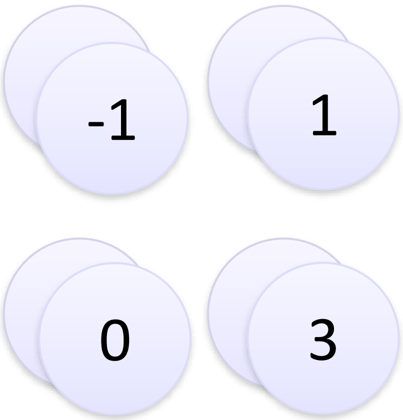
Sub-sampling: max-pooling

Parameter: Size (here: 2x2)

Filter 1 output

Filter 2 output

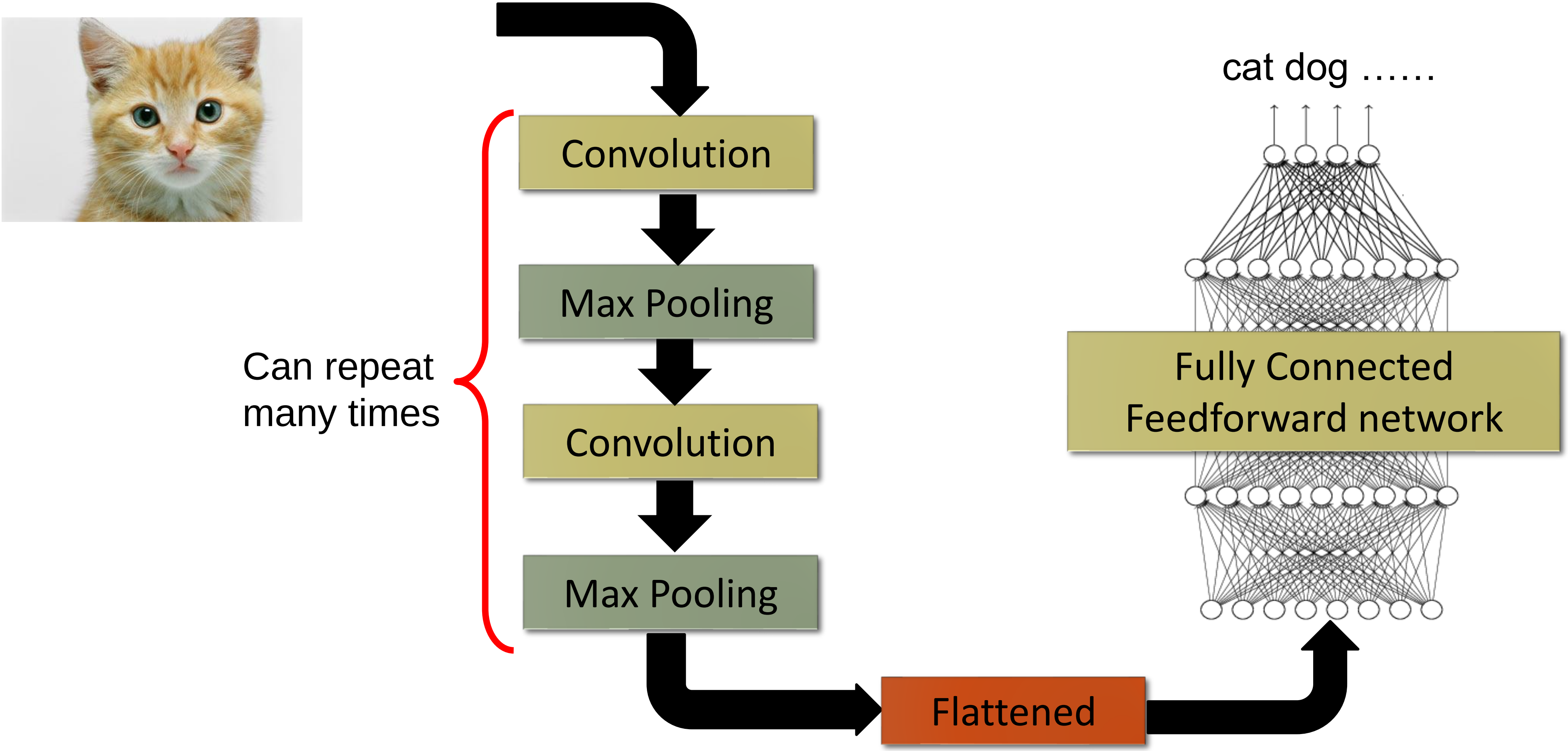




2 x 2 image

New image but smaller
Edge enhancement

The whole CNN



CNN in KERAS

1 x 28 x 28

How many parameters for each filter?

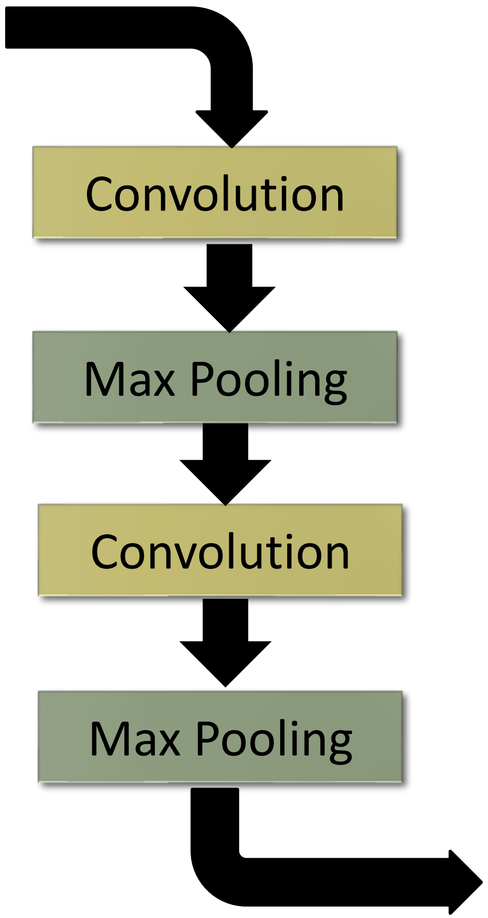
9

How many parameters for first layer ?

25x9

How many parameters for second layer ?

50x9




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

25 x 26 x 26

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

25 x 13 x 13

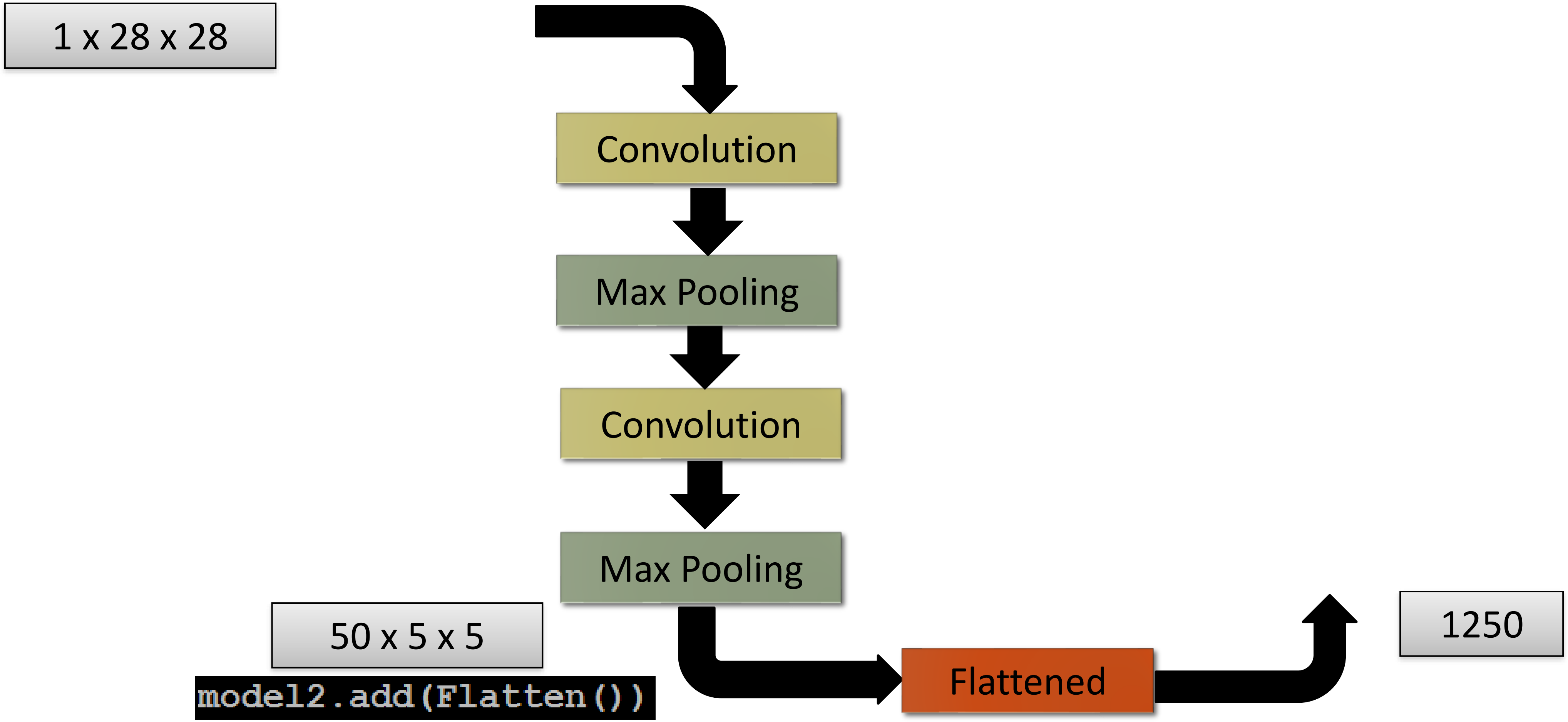
```
model2.add(Convolution2D(50,3,3))
```

50 x 11 x 11

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

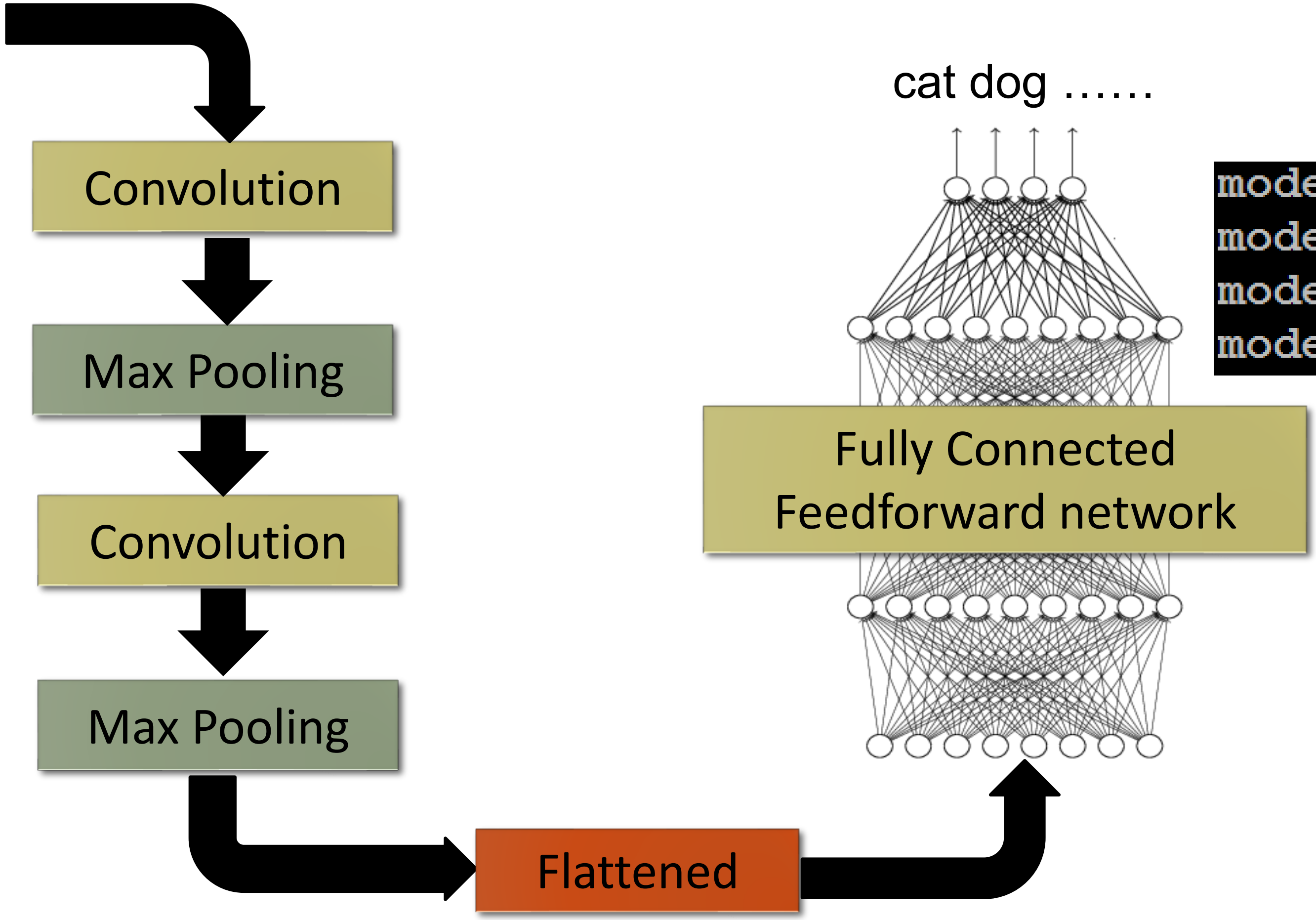
50 x 5 x 5

(...)




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

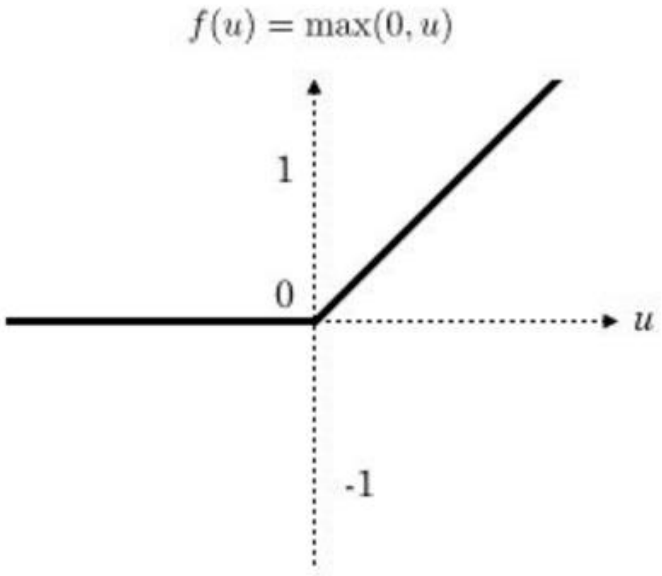
(…)




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

Activation function RELU

rectified linear function, $f(x) = max(0, x)$

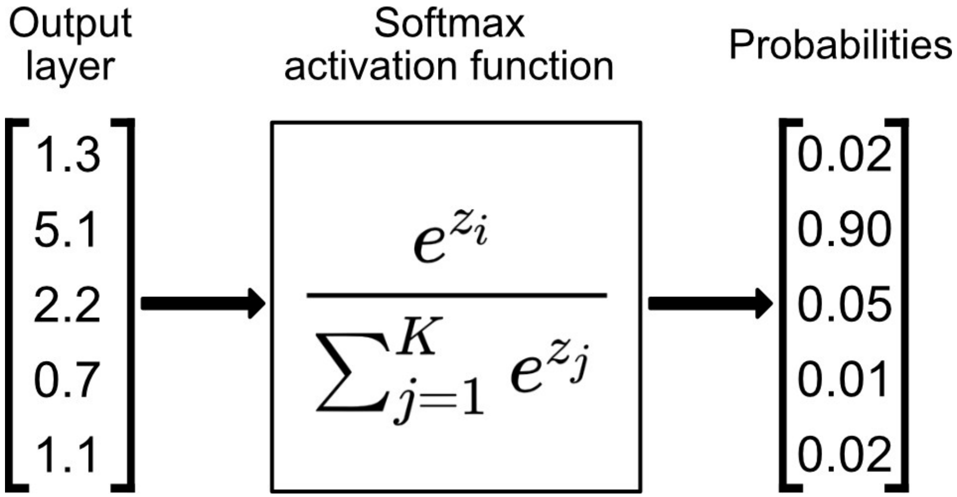


Easy to derive (→ faster):

- f'(x) = 0 for x < 0
- f'(x) = 1 for x > 0

SoftMax layer

Output: posterior probabilities p(Ci|x)



Network training

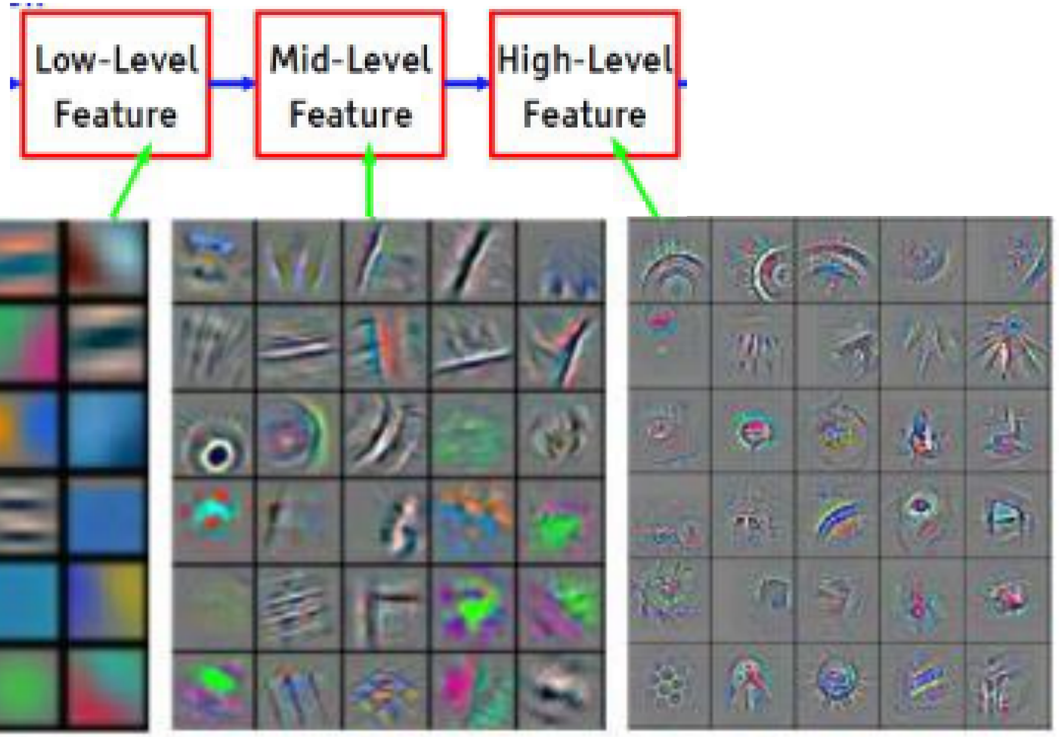
Back propagation algorithm!

<u>Example:</u>

```
model.compile(loss='binary_crossentropy', optimizer='adam',metrics=['accuracy'])

model.fit(X_train, y_train,epochs=20, batch_size=20, verbose=1)
```

Vizualization



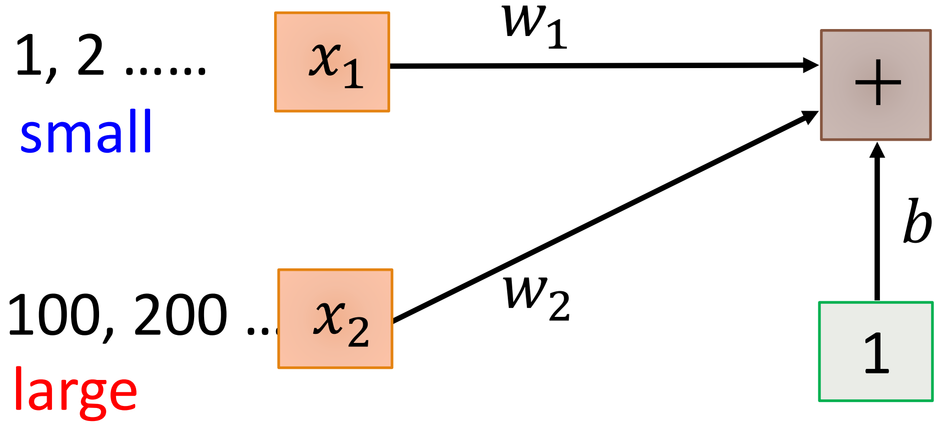
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

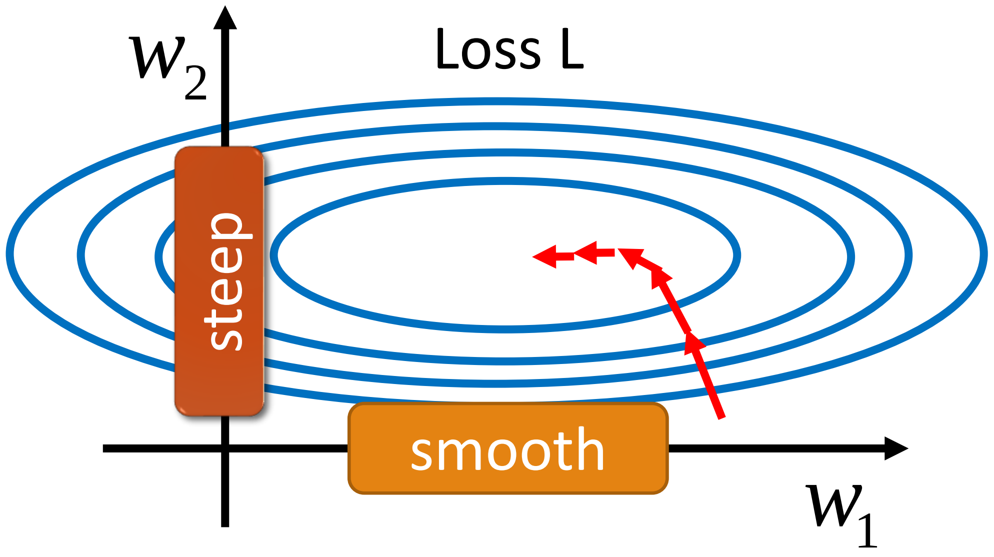
Outline

- Classical MLP’s drawbacks for complex tasks
- Convolution filters
  - 1D/2D convolution process in brief
  - Basic filters for image processing
  - High level filters for object detection
- Convolutional Neural networks
  - “Novelties”: convolution, pooling, activation function
  - Building and training a CNN using KERAS
- Improving generalization: Batch Normalization & Drop Out
- Pre-trained models

Batch Normalization

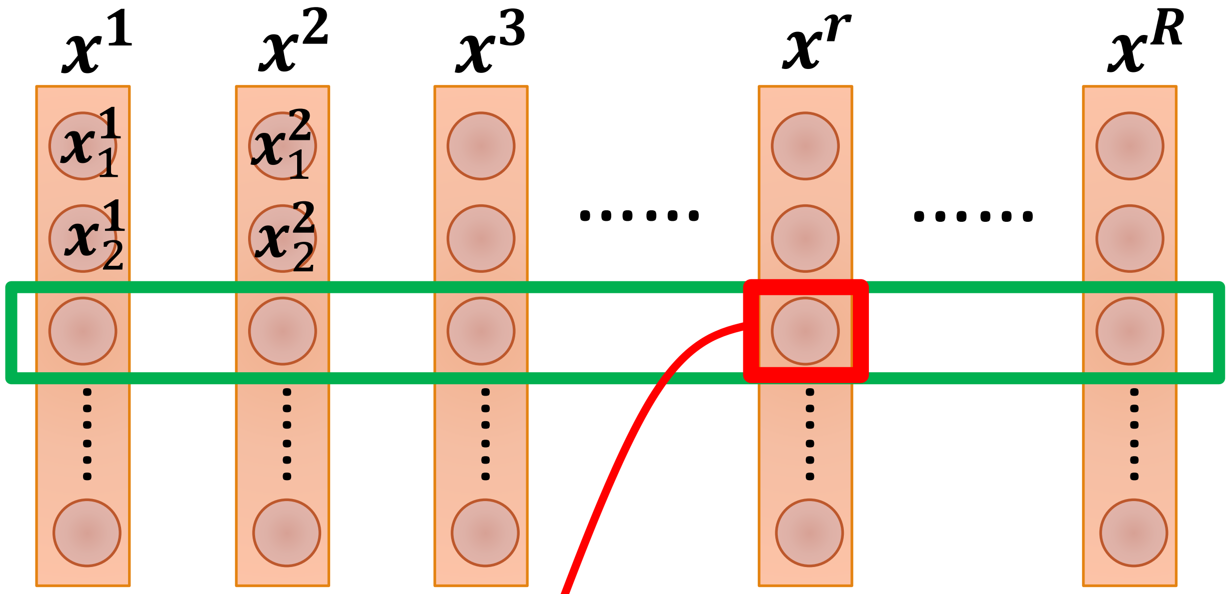
Feature normalization **revisited**:





$\Delta \mathbf{w}_i \propto \mathbf{x}_i \rightarrow$ if x_i is large then Δw_i is **large** too!

(…)



For each dimension i:

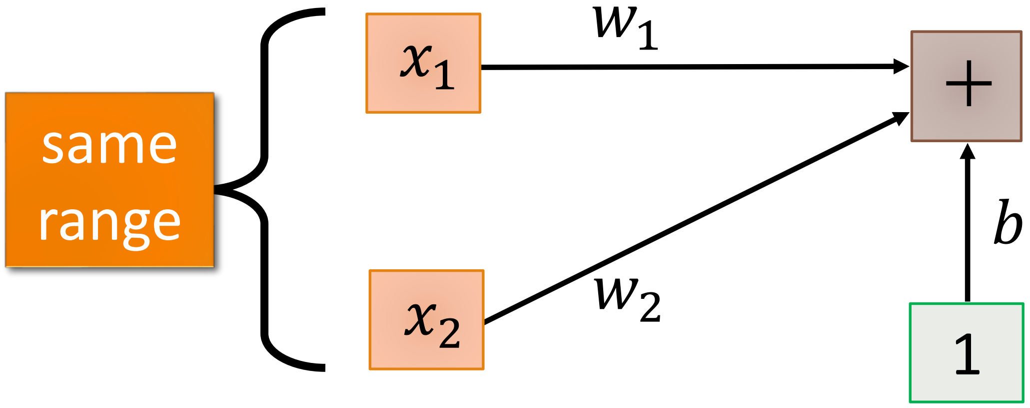
mean: m_i

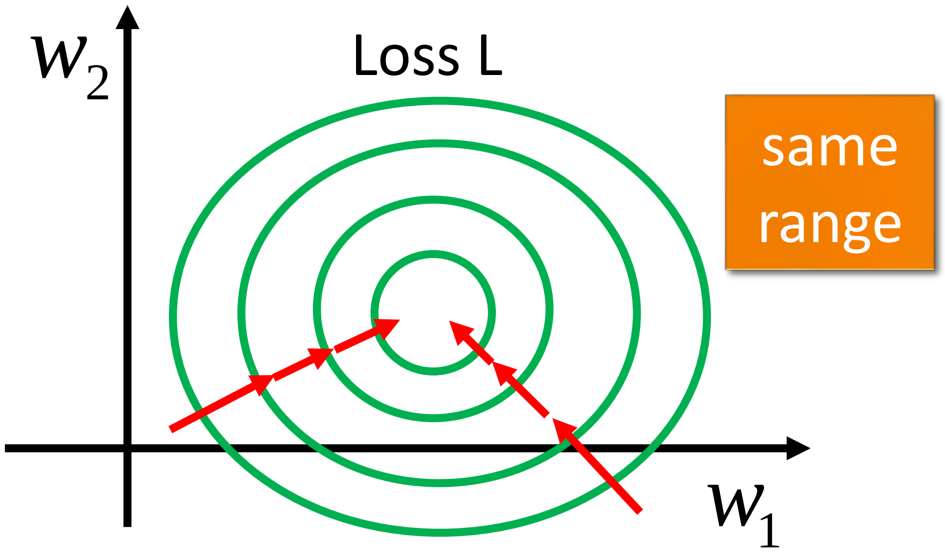
standard deviation: σ_i

$$\tilde{\boldsymbol{x}}_i^r \leftarrow \frac{\boldsymbol{x}_i^r - m_i}{\sigma_i}$$

The means of all dims are 0, and the variances are all 1

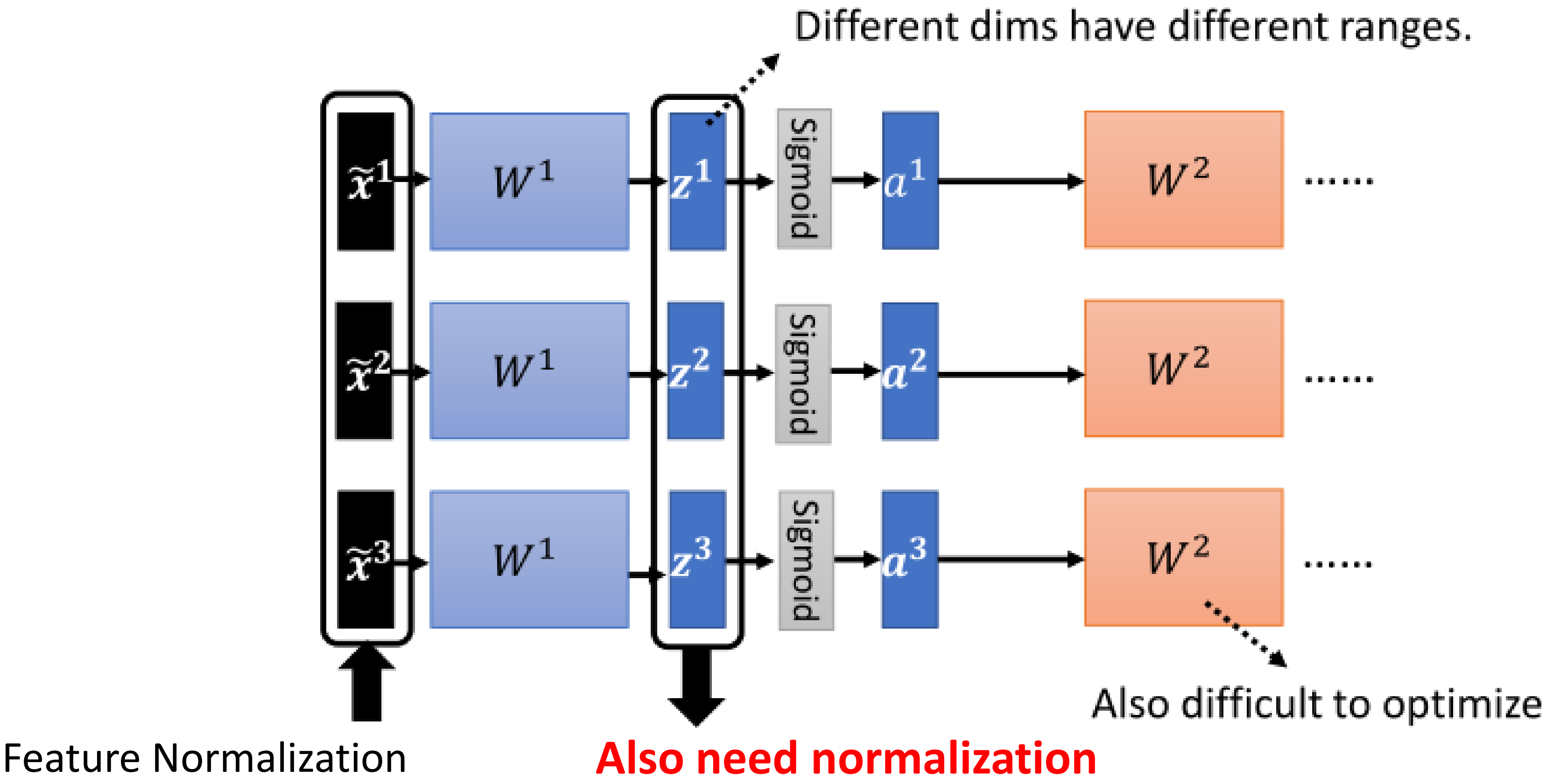
(…)



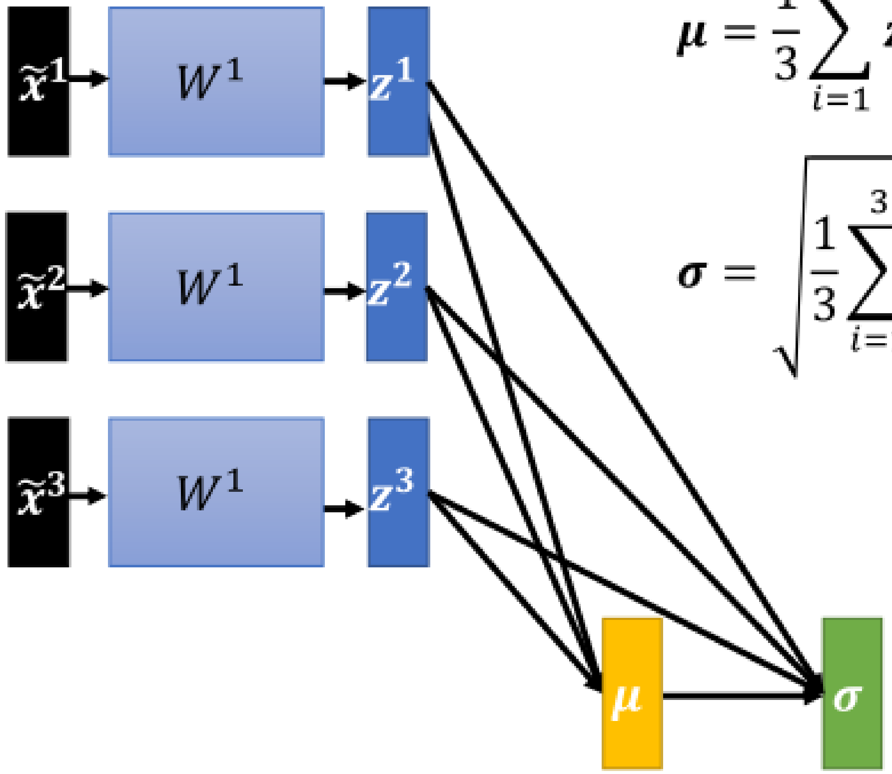


→ Feature normalization makes gradient descent **converge faster**.

Batch normalization in deep networks



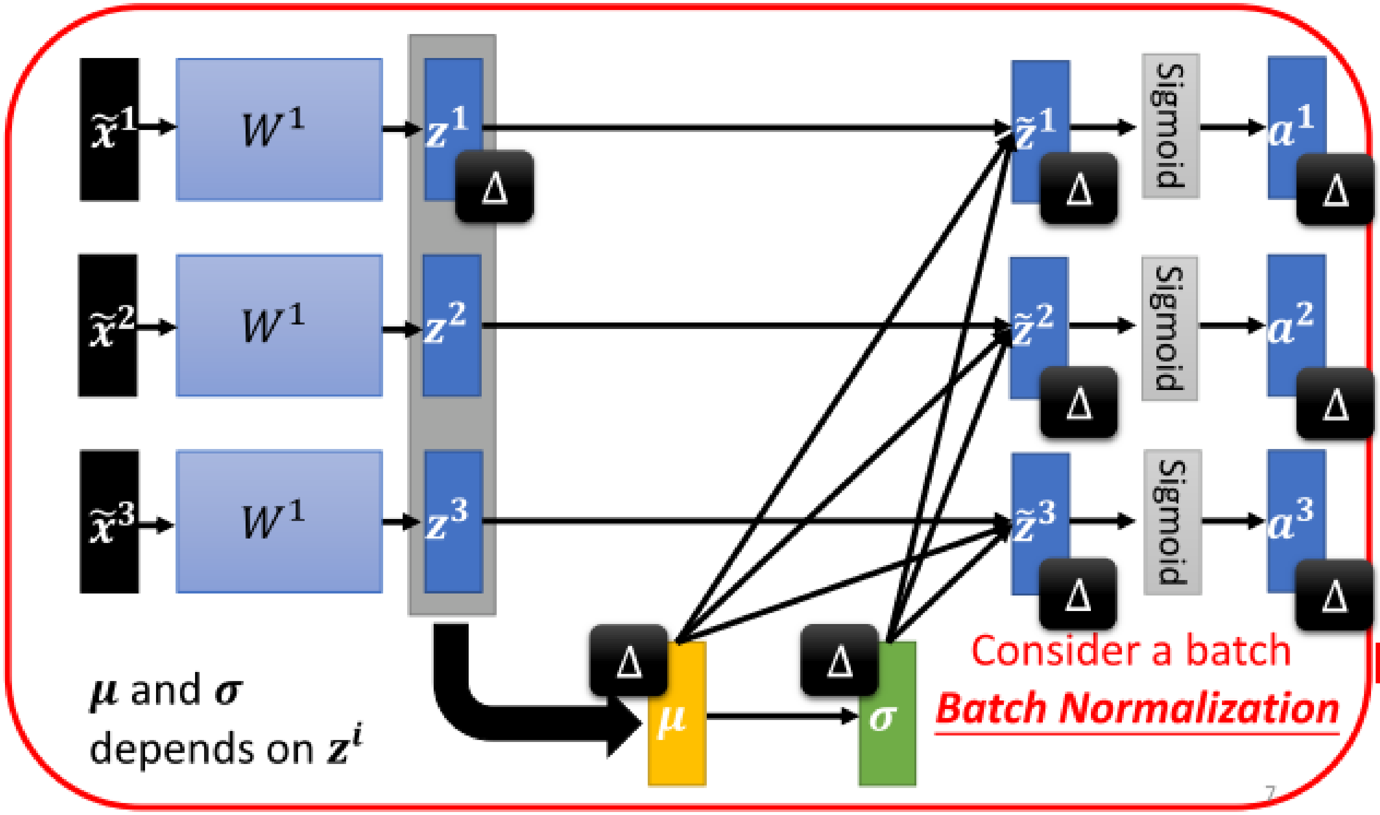
(...)



$$\boldsymbol{\mu} = \frac{1}{3}\sum_{i=1}^{3} \boldsymbol{z}^i$$

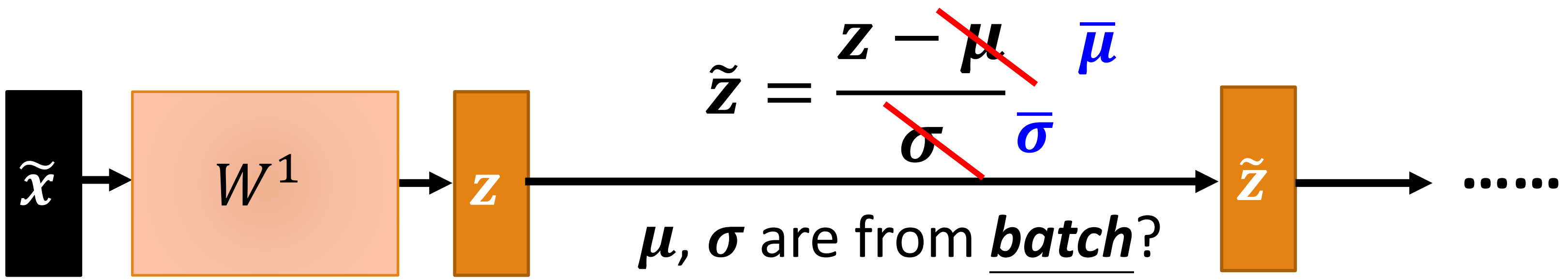
$$\boldsymbol{\sigma} = \sqrt{\frac{1}{3}\sum_{i=1}^{3} (\boldsymbol{z}^i - \boldsymbol{\mu})^2}$$

(…)



$$\tilde{z}^i = \frac{z^i - \mu}{\sigma}$$

Batch normalization: testing

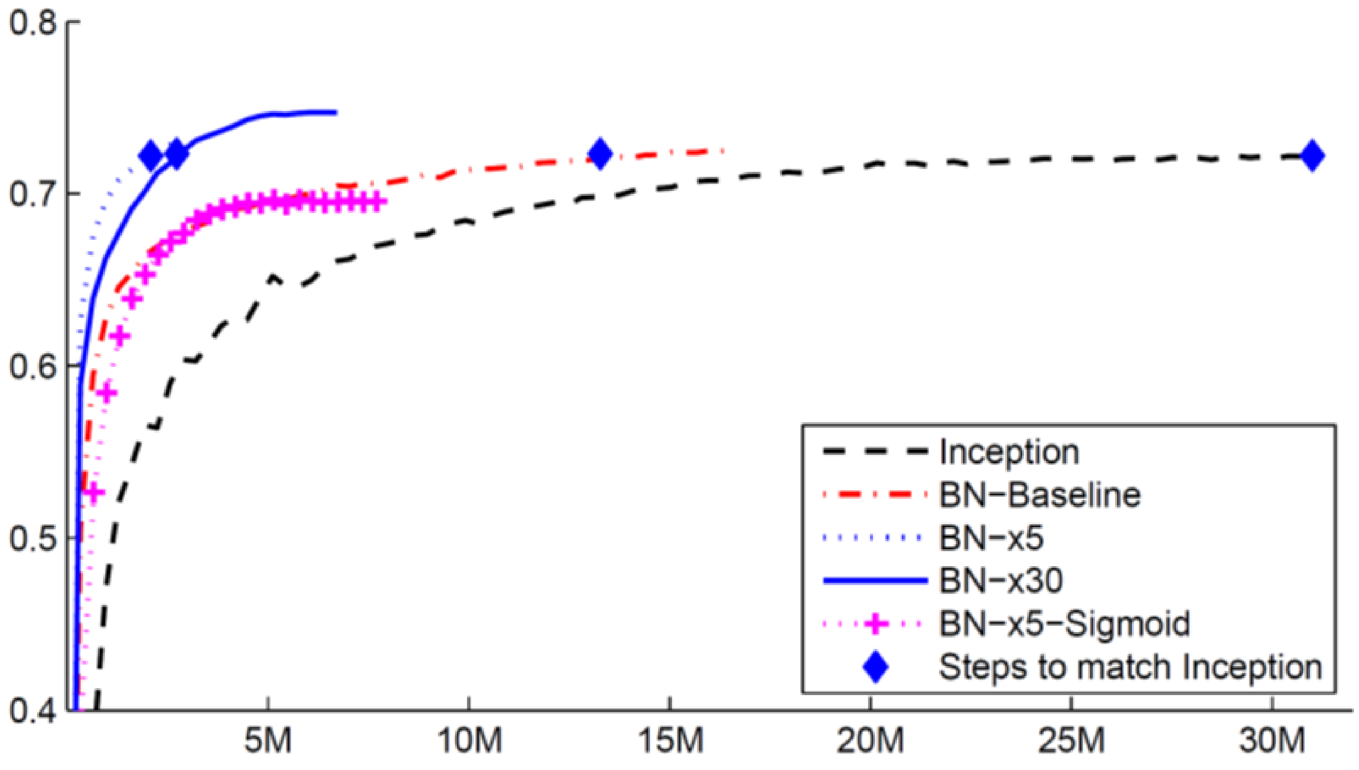


No ***batch*** at testing stage!

Computing the moving average of μ and σ of the batches during training.

μ^1 μ^2 μ^3 …… μ^t $\bar{\mu} \leftarrow p\bar{\mu} + (1-p)\mu^t$

Batch normalization: impact



Original paper: https://arxiv.org/abs/1502.03167

Drop Out: why?

Deep nets have many non-linear hidden layers

– Making them very expressive to learn complicated relationships between inputs and outputs

– But with limited training data, many complicated relationships will be the result of training noise

Many methods developed to reduce overfitting

– Early stopping with a validation set

– Weight sharing

(...)

Best way to regularize a fixed size model is:

– Average the predictions of all possible settings of the parameters

– Weighting each setting with the posterior probability given the training data

Dropout does this using considerably less computation

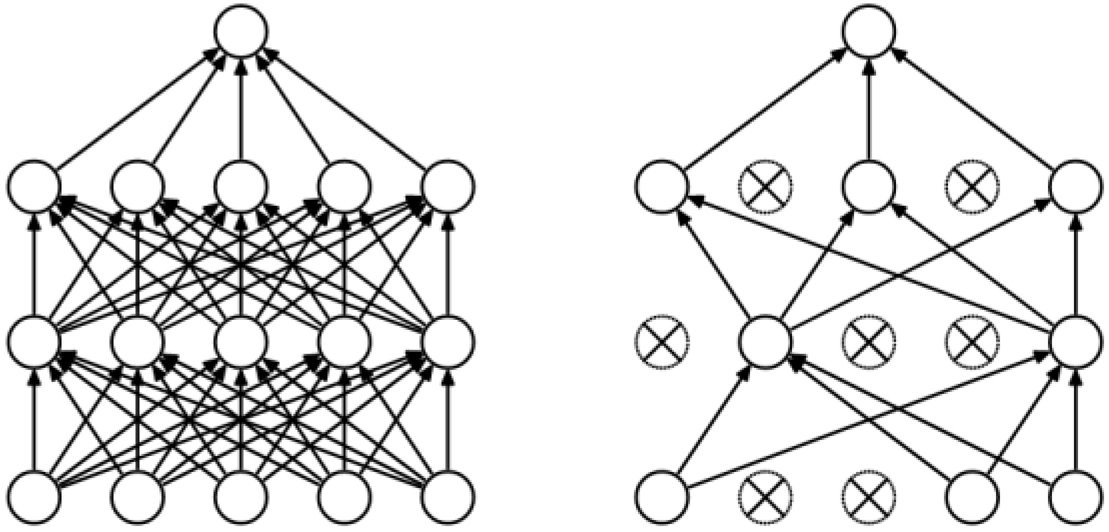
– By approximating an equally weighted geometric mean of the predictions of an exponential number of learned <u>models that share parameters</u>

Drop Out: How?

Removing units creates networks!

– Subnetworks formed by removing <u>non-output units</u> from the underlying base network

→ subnetwork: example

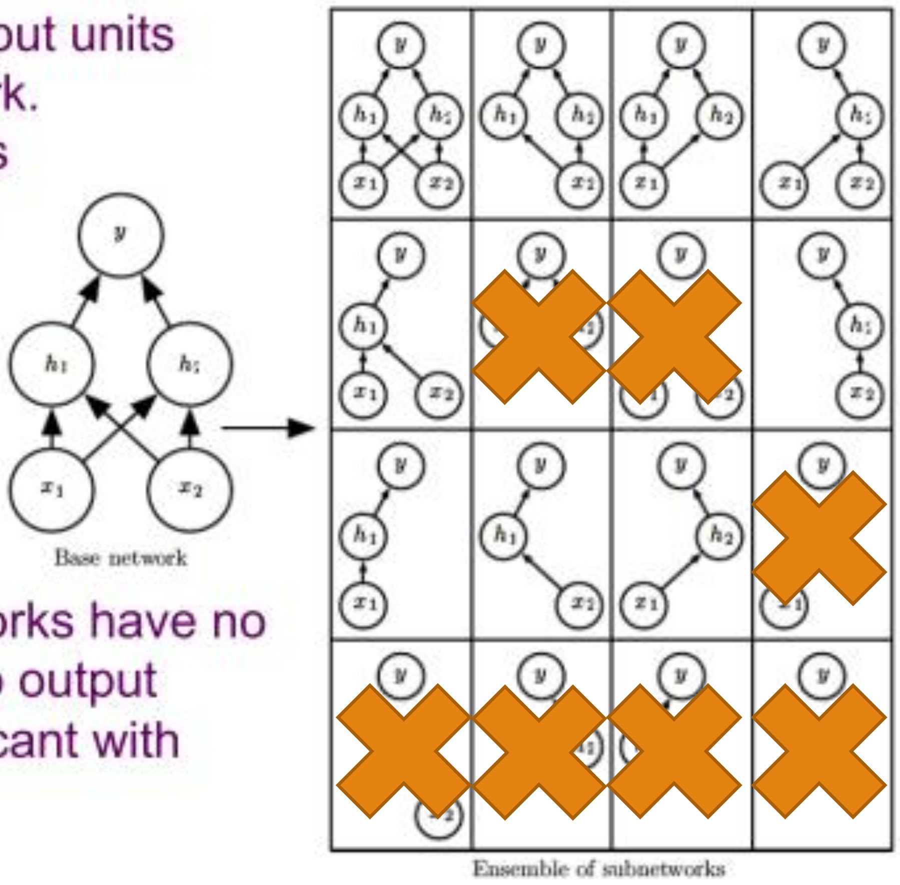


(a) Standard Neural Net

(b) After applying dropout.

(...)

- Remove non-output units from base network.
- Remaining 4 units yield 16 networks



- Here many networks have no path from input to output
- Problem insignificant with large networks

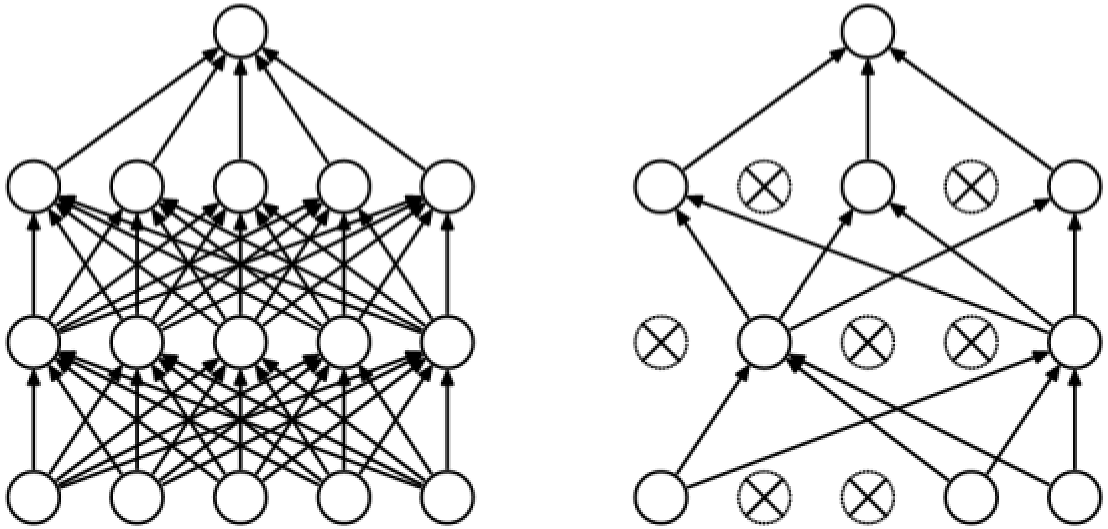
(...)

Drop hidden and visible units from net, i.e., temporarily remove it from the network with all input/output connections.

Choice of units to drop is random, determined by a probability p.

Bernoulli Distribution

$$f(x|p) = \begin{cases} p & x = 1 \\ 1 - p & x = 0 \end{cases}$$



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

Drop Out in practice: training

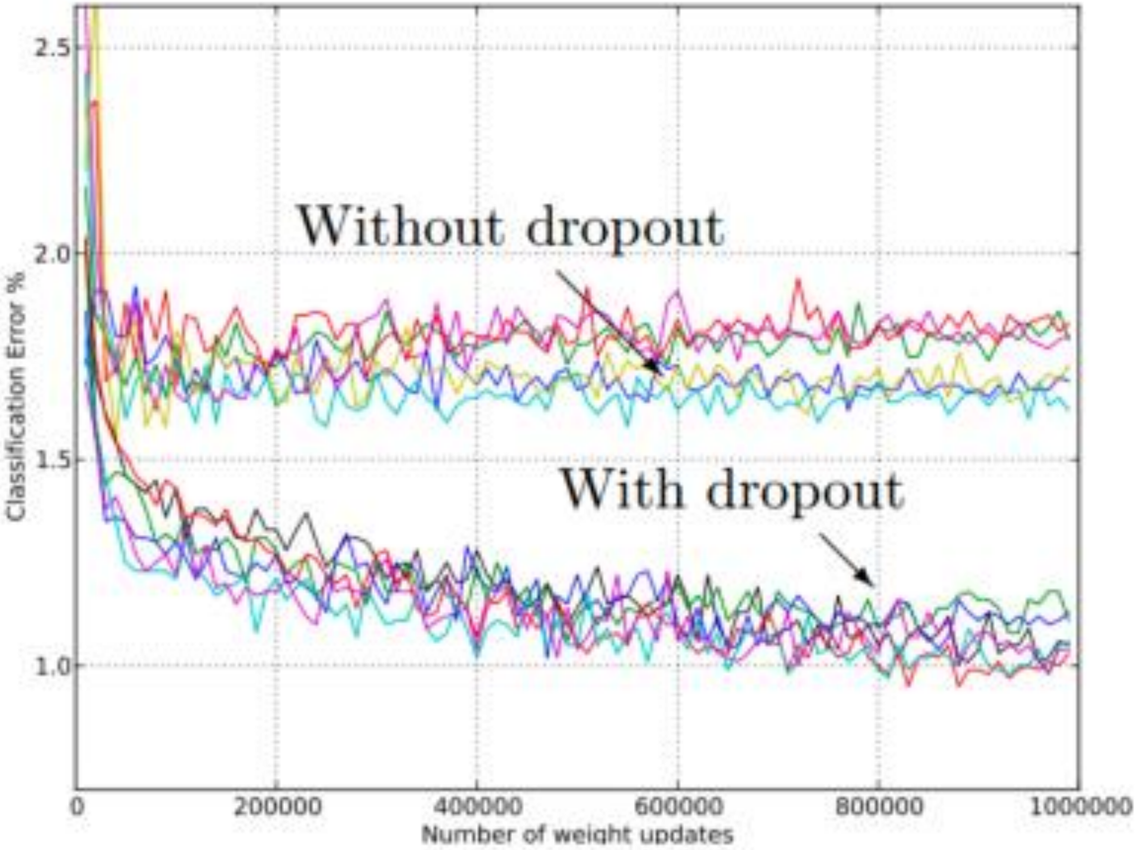
To train with dropout:

- we use minibatch based learning algorithm that takes small steps such as SGD

  → At each step randomly sample a binary mask
  → Probability of including a unit is a hyperparameter
  (for example: 0.5 for hidden units and 0.8 for input units)

- We run forward & backward propagation as usual

Drop Out in practice: performance



Outline

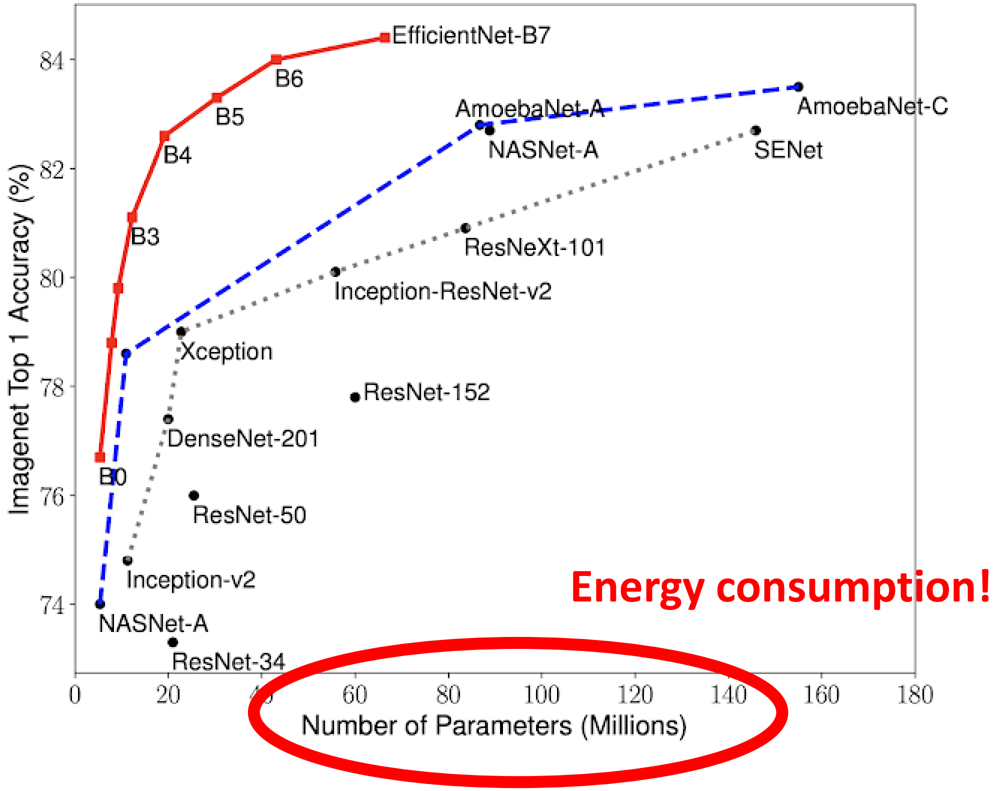
- Classical MLP’s drawbacks for complex tasks
- Convolution filters
  - 1D/2D convolution process in brief
  - Basic filters for image processing
  - High level filters for object detection
- Convolutional Neural networks
  - “Novelties”: convolution, pooling, activation function
  - Building and training a CNN using KERAS
- Improving generalization: Batch Normalization & Drop Out
- Pre-trained models

Pre-trained models

Pre-Trained Models for Image Classification

Winners of Imagenet challenge

- VGG-16 (2014)
- ResNet50 (2015)
- Inceptionv3 (2019)
- EfficientNet (2019)



[Top 4 Pre-Trained Models for Image Classification with Python Code]

Sources

Deep-Learning-2017-Lecture5CNN

https://speech.ee.ntu.edu.tw/~hylee/ml/ml2021-course-data/normalization_v4.pptx

https://cedar.buffalo.edu/~srihari/CSE676/7.12%20Dropout.pdf