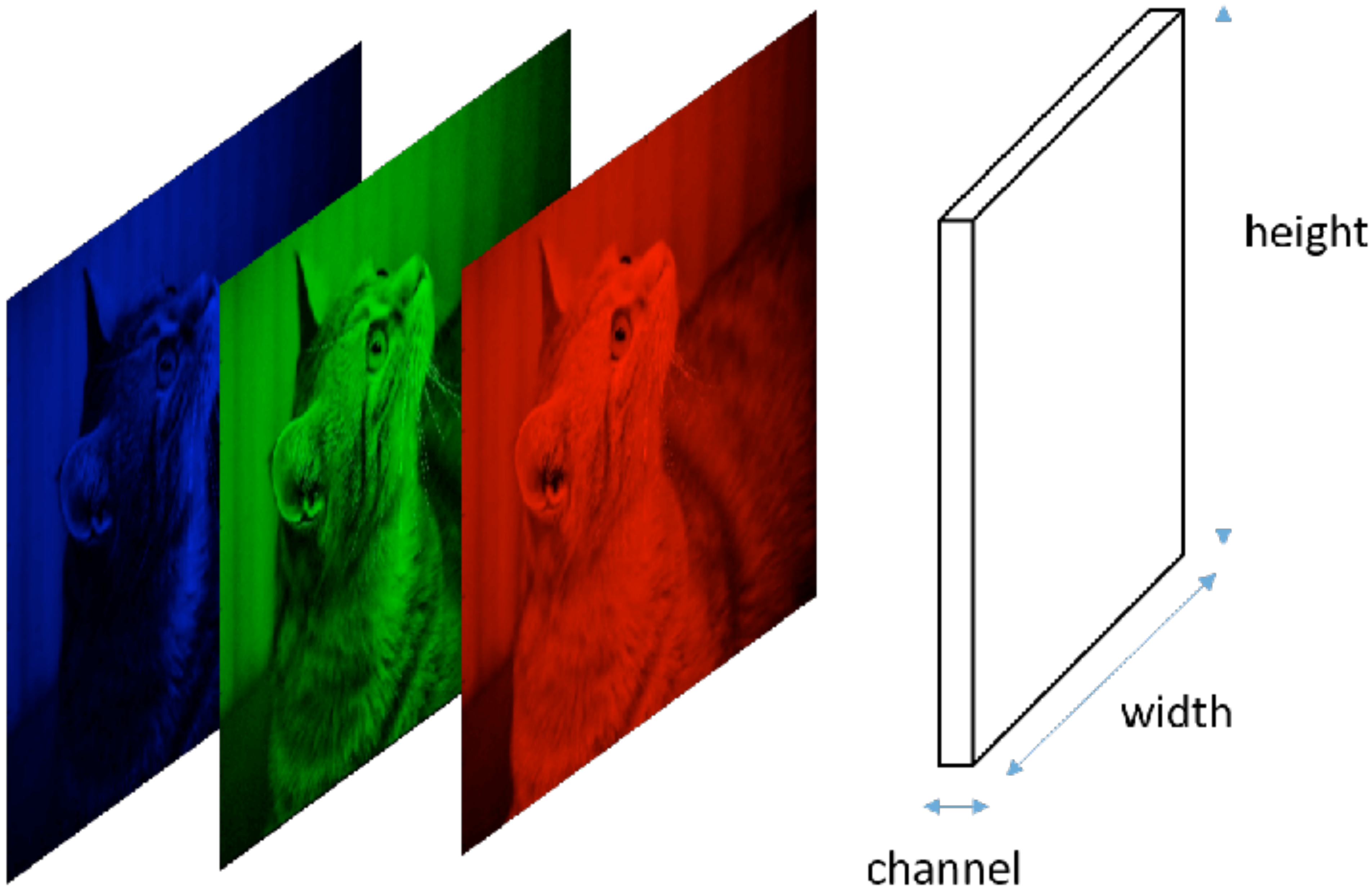
Images and
Convolutional neural networks

Romain Tavenard (Université de Rennes)
A course @UR2

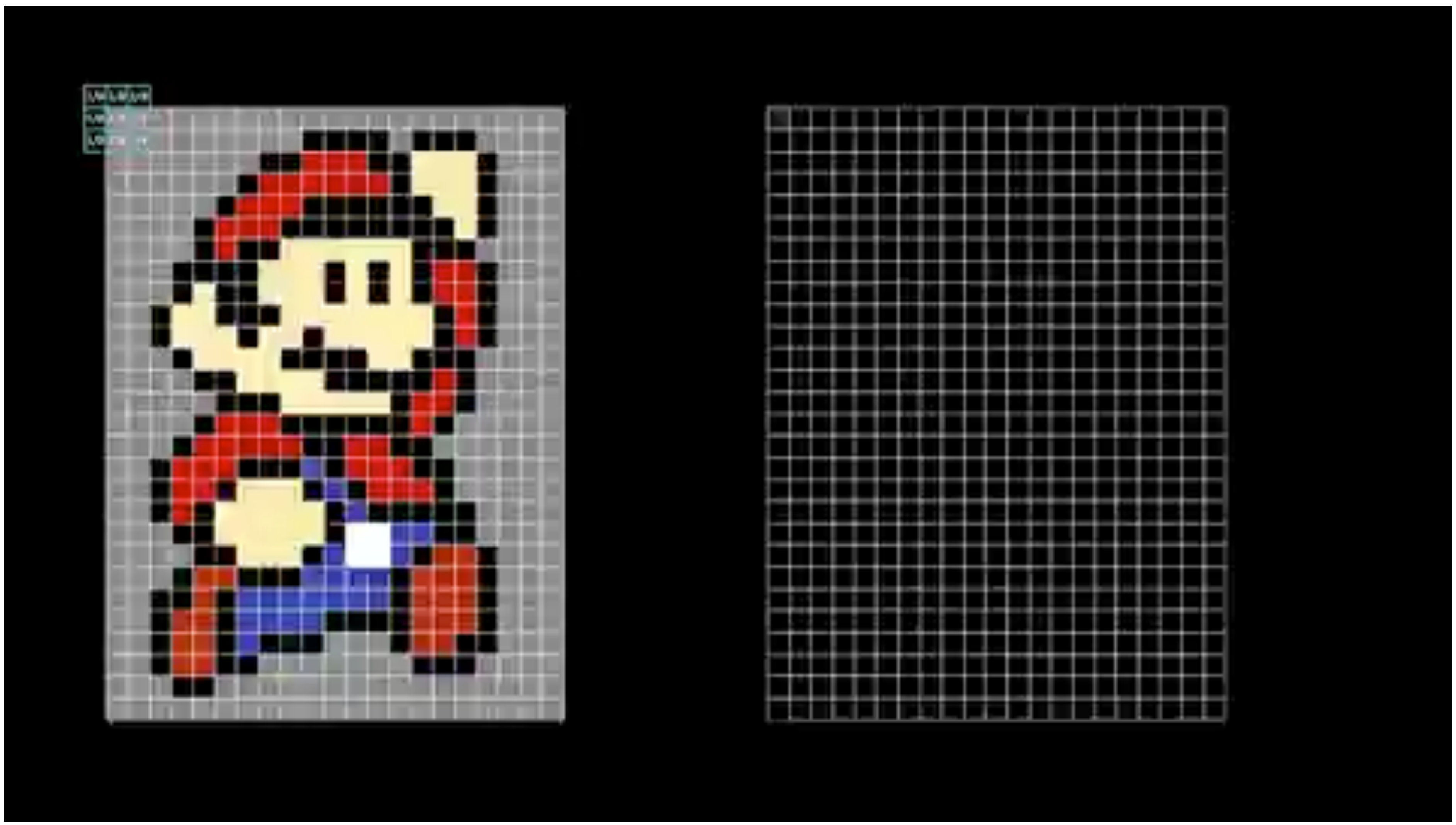
NB: Most figures in these slides are from
Dumoulin & Visin. A guide to convolution arithmetic for deep learning. 2016

Preamble: What's an image?



Source : « Understanding Images with skimage-Python », Towards Data Science

Convolution in practice

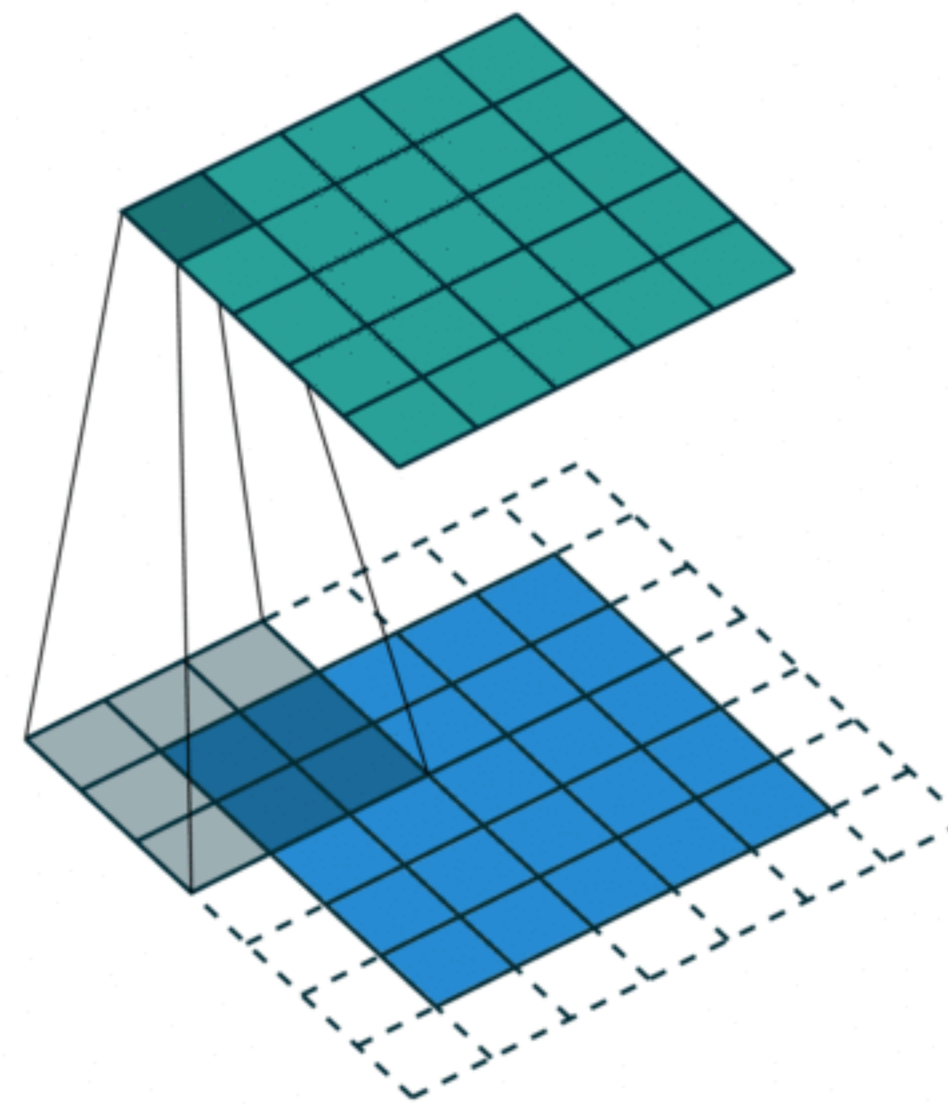


Source : Grant Sanderson, Twitter
https://twitter.com/3blue1brown/status/1303489896519139328?s=20

The convolution operator

- 2D convolution
  - Blue: input image
  - Gray: convolution kernel
  - Cyan: activation map

- Convolution operation = Dot product between
  - convolution kernel (aka filter)
  - subpart of the input

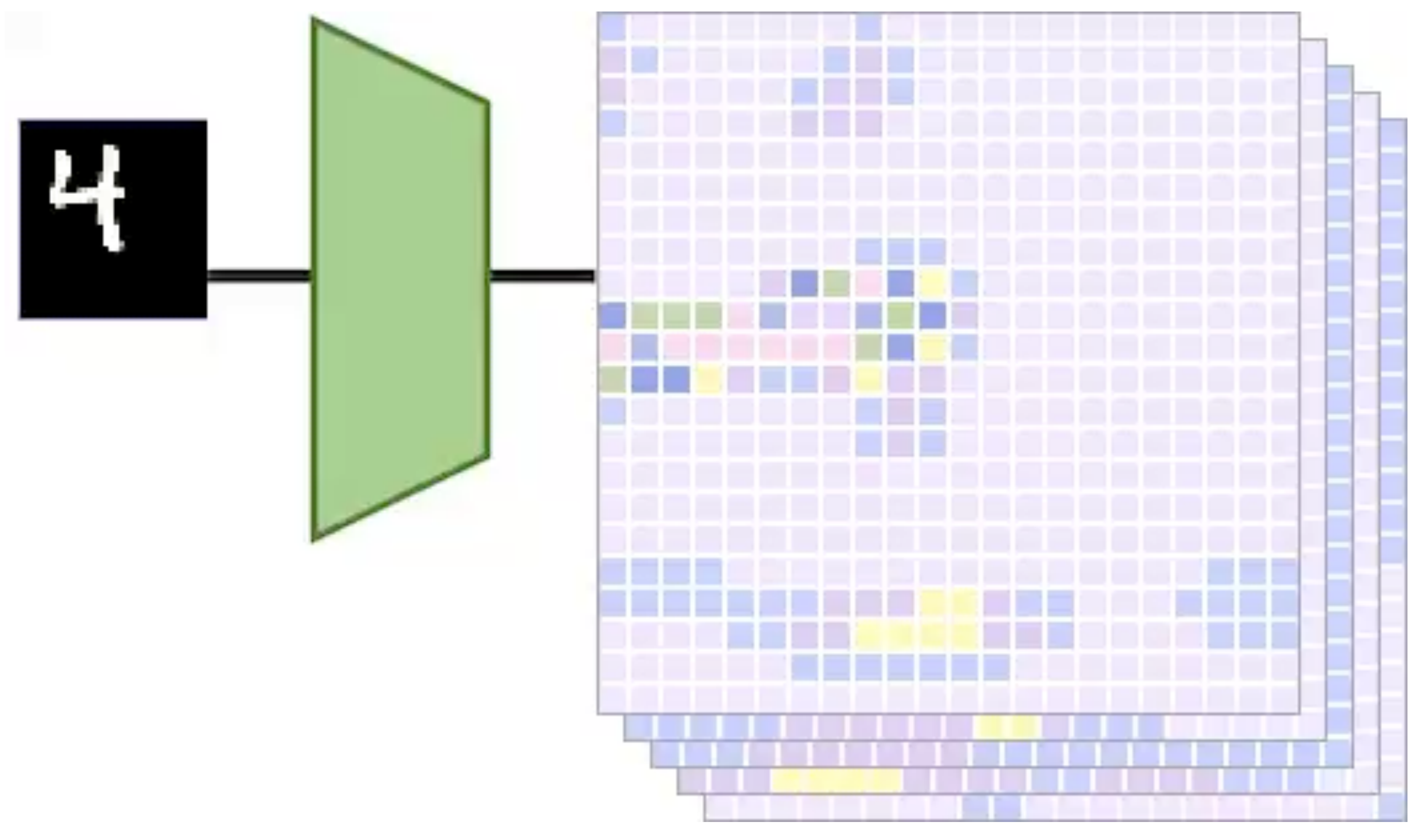


Convolutional layers in NN (1/2)

- A convolution layer is made of
  - convolution kernels
  - biases (1 per kernel)
  - an activation function

- Useful because
  - reduces #parameters
  - encodes translation equivariance
    (translation in the input induces translation in the output, cf. next slide)

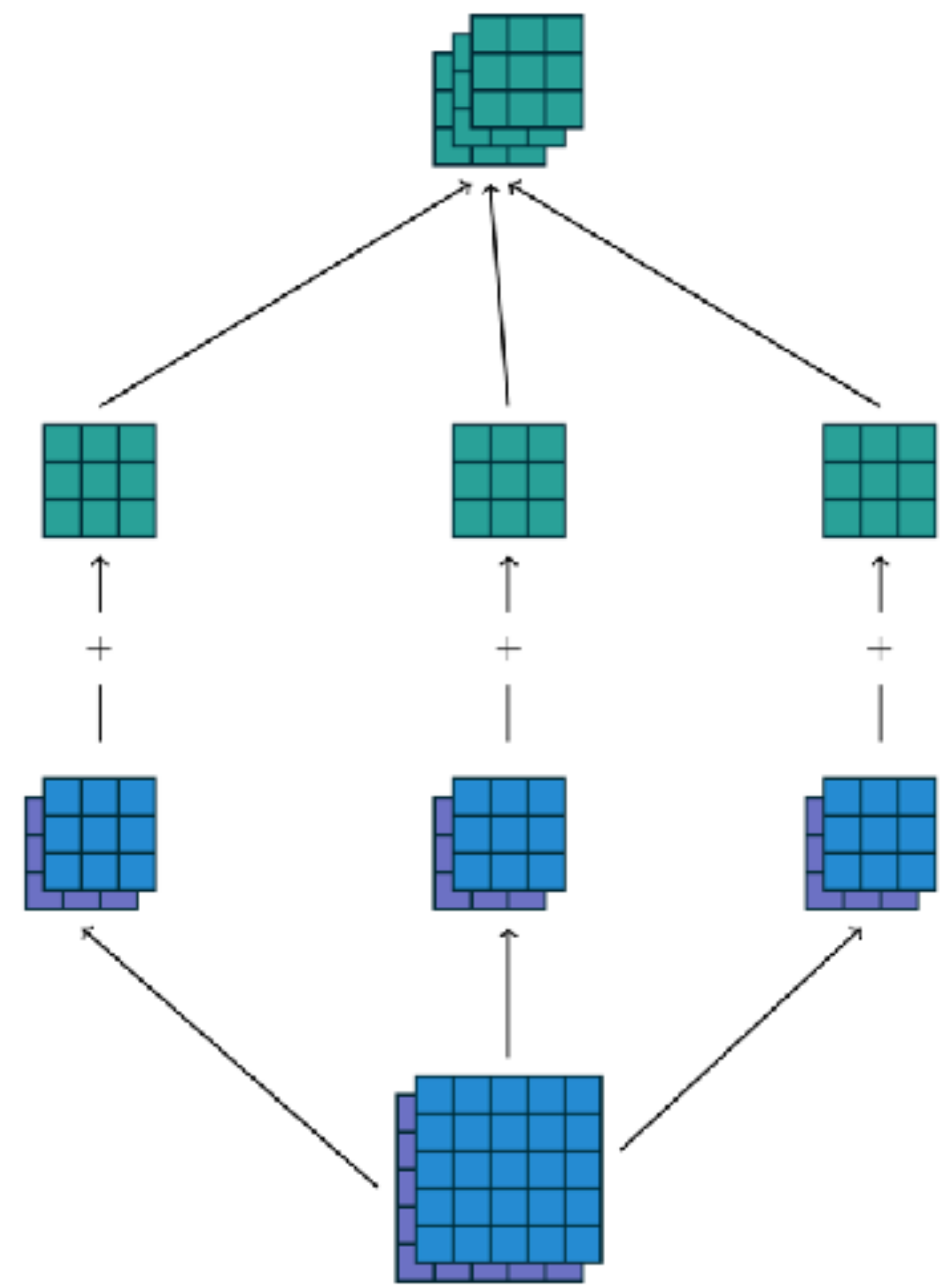
Convolution and translation



Source : Christian Wolf, Twitter
https://twitter.com/chriswolfvision/status/1313059518574718977?s=20

Convolutional layers in NN (2/2)

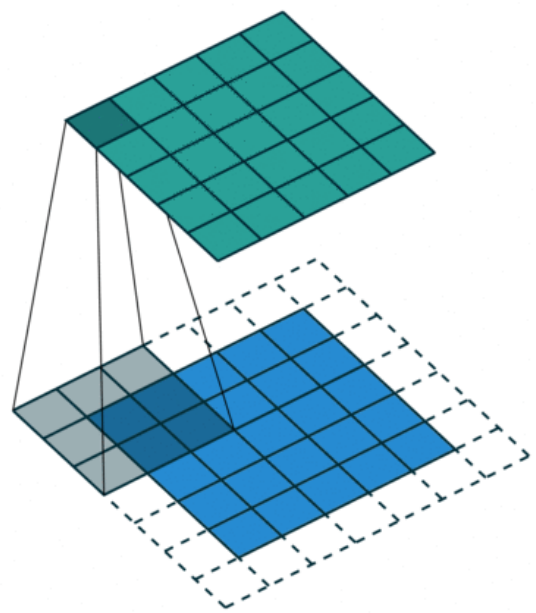
- Multiple input channel case
  - sum the response over all channels
- Multiple kernel case
  - each kernel leads to one output channel



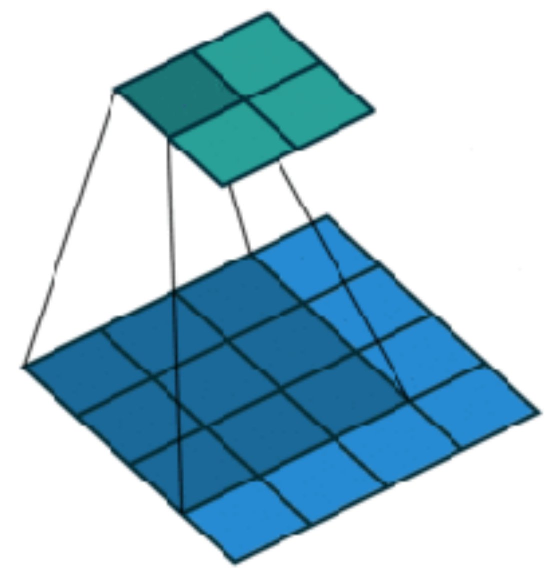
2 input channels, 3 kernels

Convolutional layers in NN: hyper parameters

- Padding

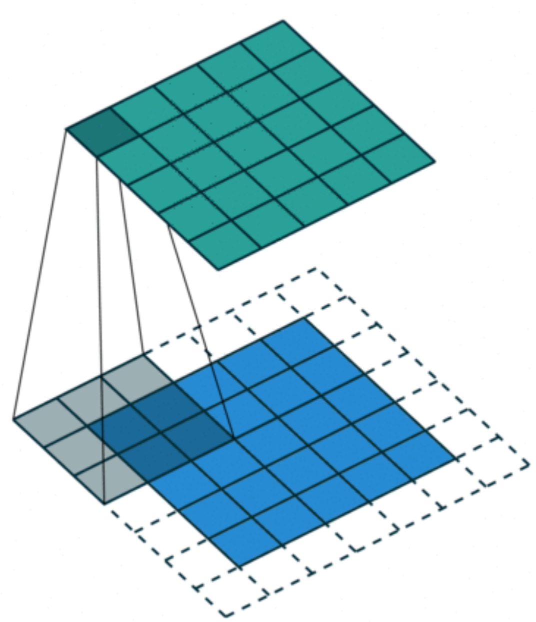


padding="same"

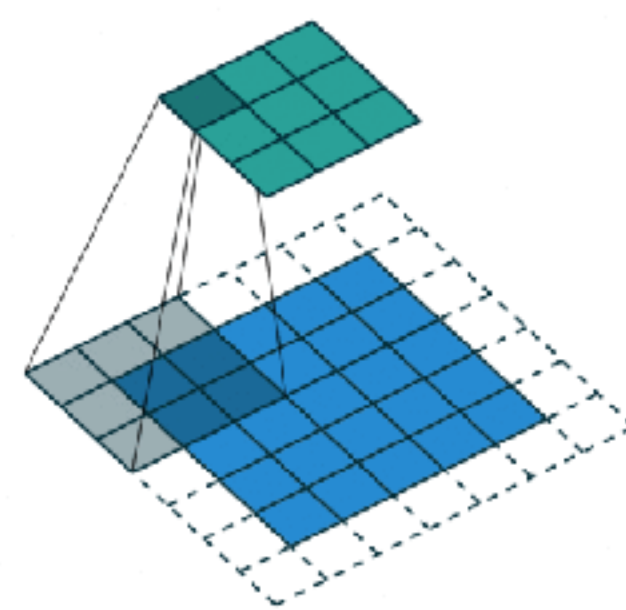


padding="valid"

- Strides



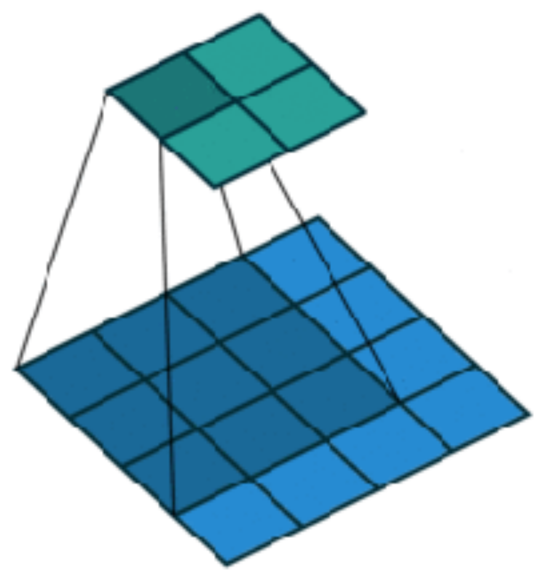
strides=1



strides=2

Computing the size of an activation map

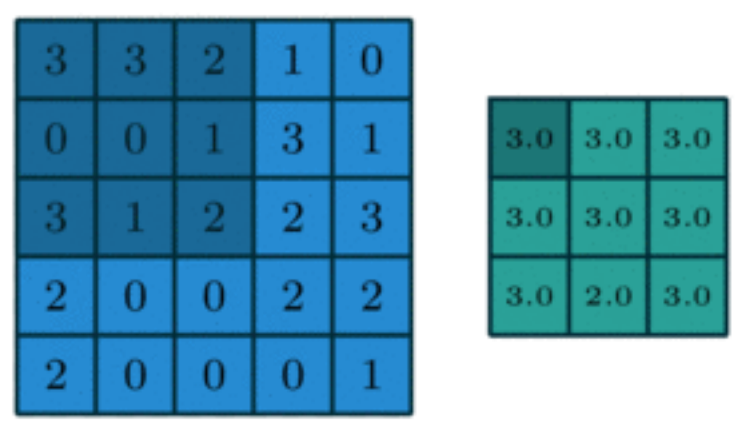
- Assumption: no padding ("valid"), unit strides



$$W_{\text{out}} = W_{\text{in}} - W_k + 1$$
$$H_{\text{out}} = H_{\text{in}} - H_k + 1$$

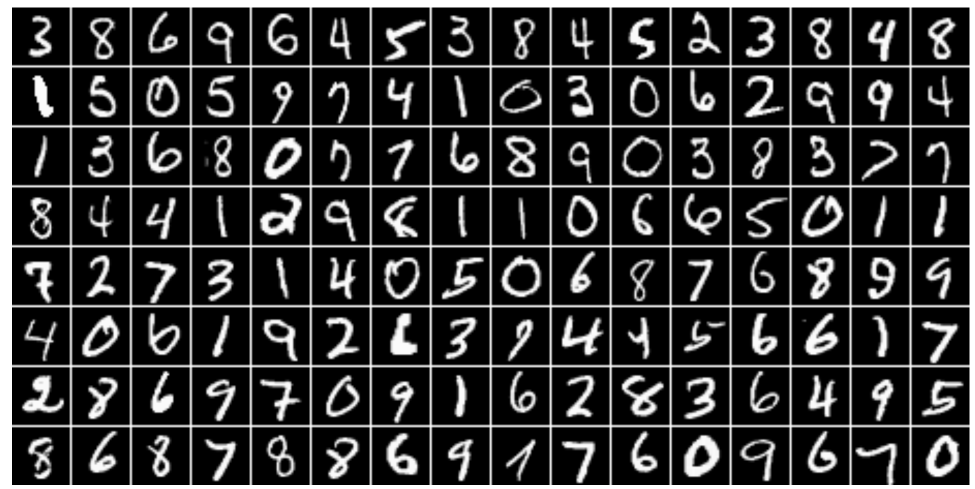
Pooling (aka subsampling) layers in NN

- Max pooling / Average pooling
- Hyper-parameters
  - pool size
  - strides (use None in keras)
  - padding (use "valid" in keras)

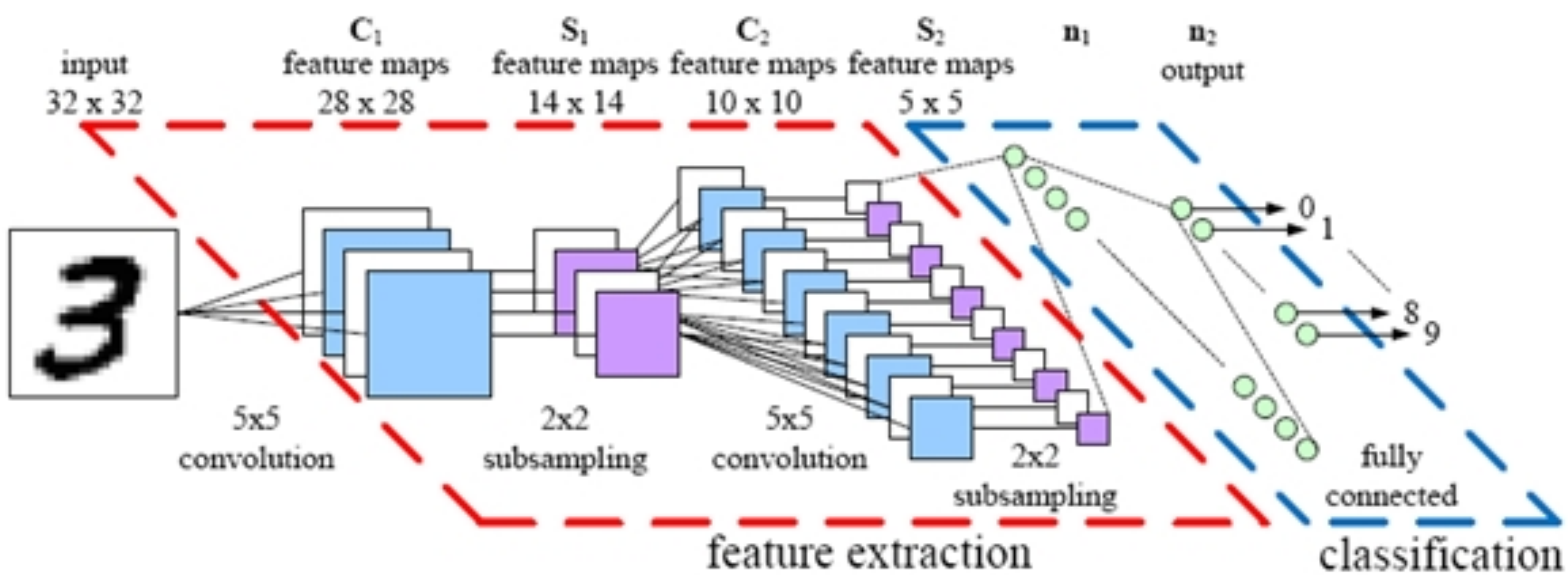


pool_size=3, strides=1
(not recommended)

A history of Convolutional neural networks (CNN) (LeCun, 1989)

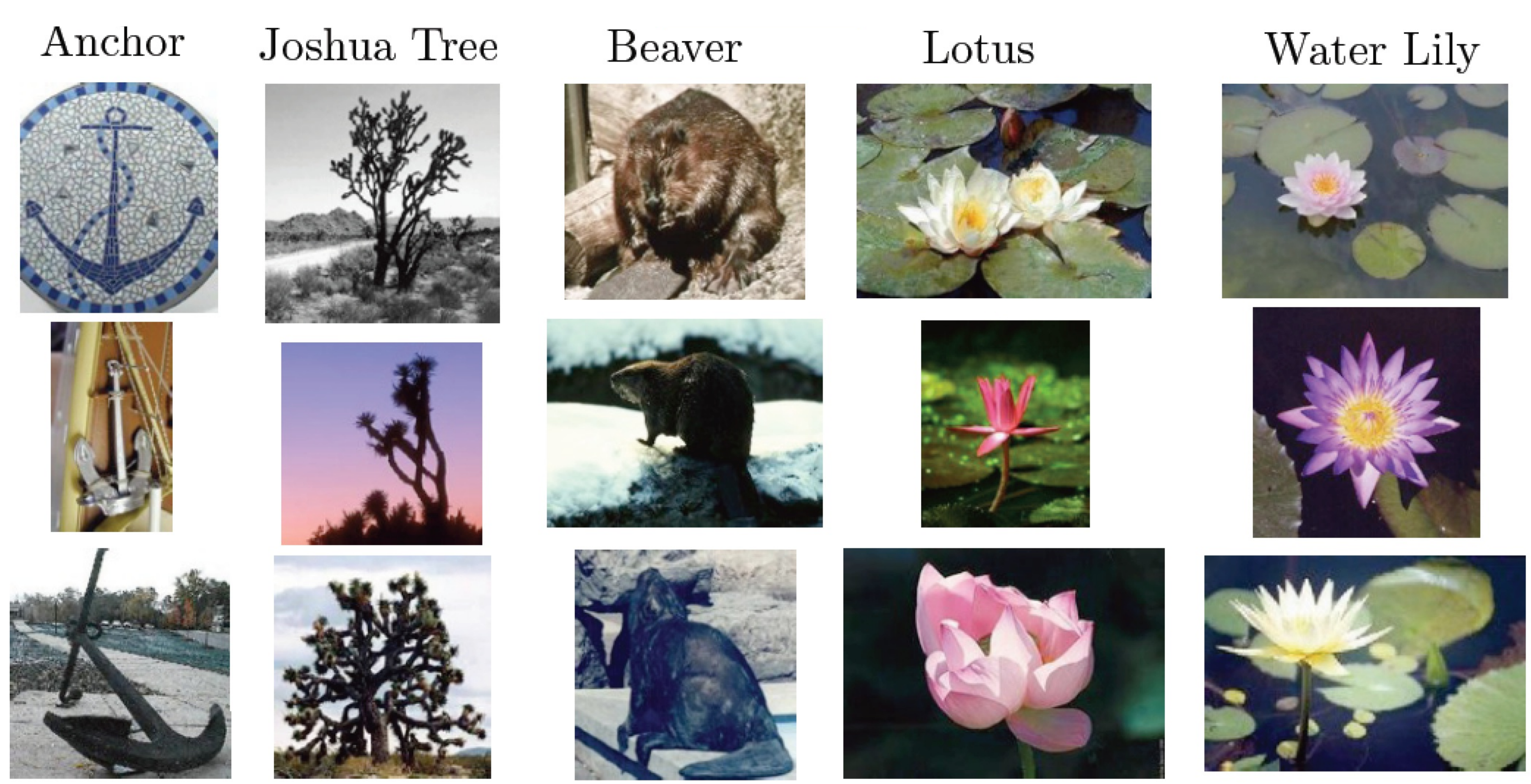


MNIST dataset
10 classes
60,000 images



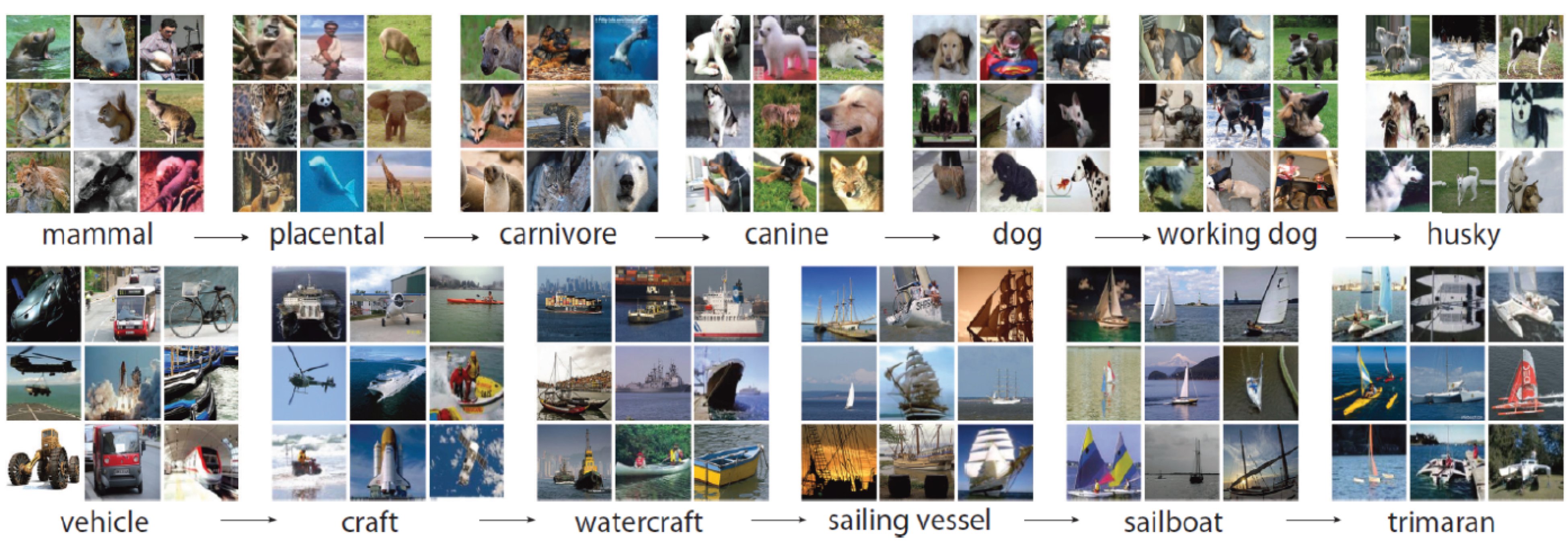
A history of Convolutional neural networks (CNN)
Caltech 101: the second winter (2004)

- 101 classes
- 30 training images per class
- NN are bad competitors here
  - Dataset is too small



A history of Convolutional neural networks (CNN)
ImageNet & LSVRC (2012)

- ImageNet
  - 15M images
  - 22k classes
- LSVRC
  - Subset of ImageNet (1.2M images, 1k classes)



A history of Convolutional neural networks (CNN)
A drastic improvement on performance (LSVRC)

2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

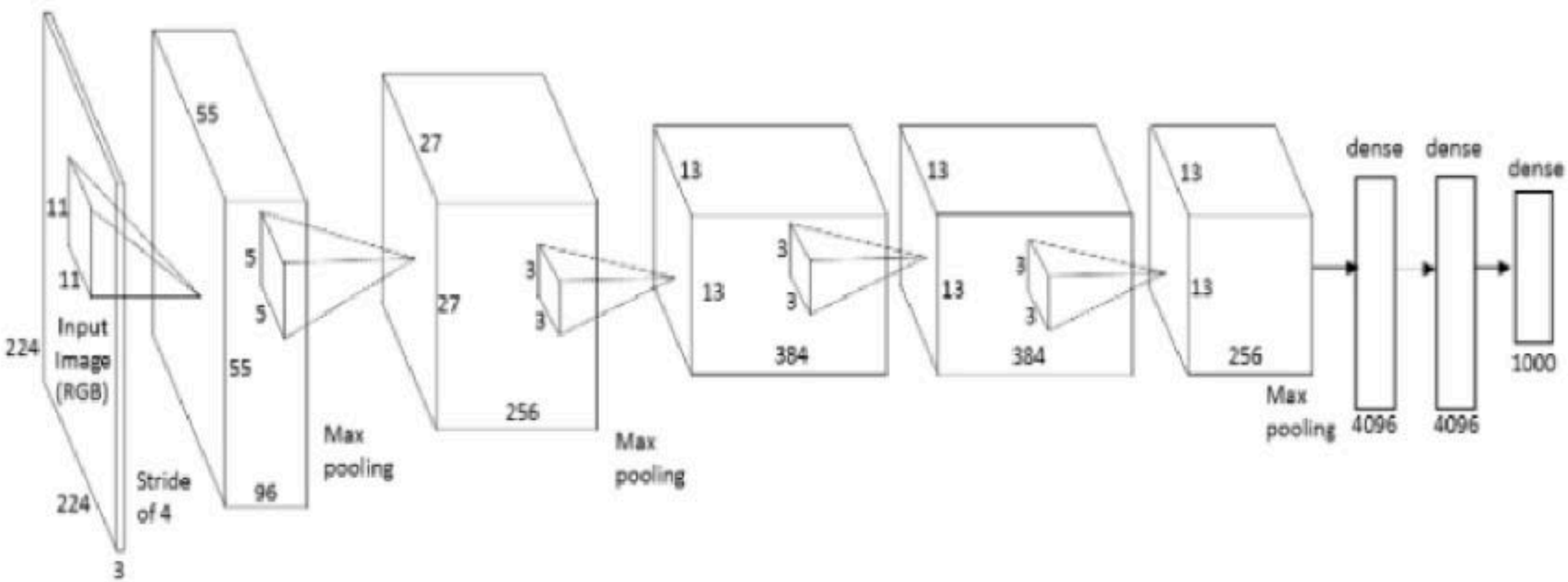
2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

shallow approaches

deep learning

Source : Y. LeCun's tutorial @StatLearn

A history of Convolutional neural networks (CNN)
AlexNet (2012)



- Error rate : 15%
- 60M parameters
- 2 GPUs – 6 days

- Regularization
  - Data augmentation
  - Dropout
  - L2

Regularization: Data Augmentation

- Principle: generate virtual training examples
  - original image x_i
  - modified image $\hat{x}_i$
  - unchanged label y_i

$$y_i = \text{“cat”}$$

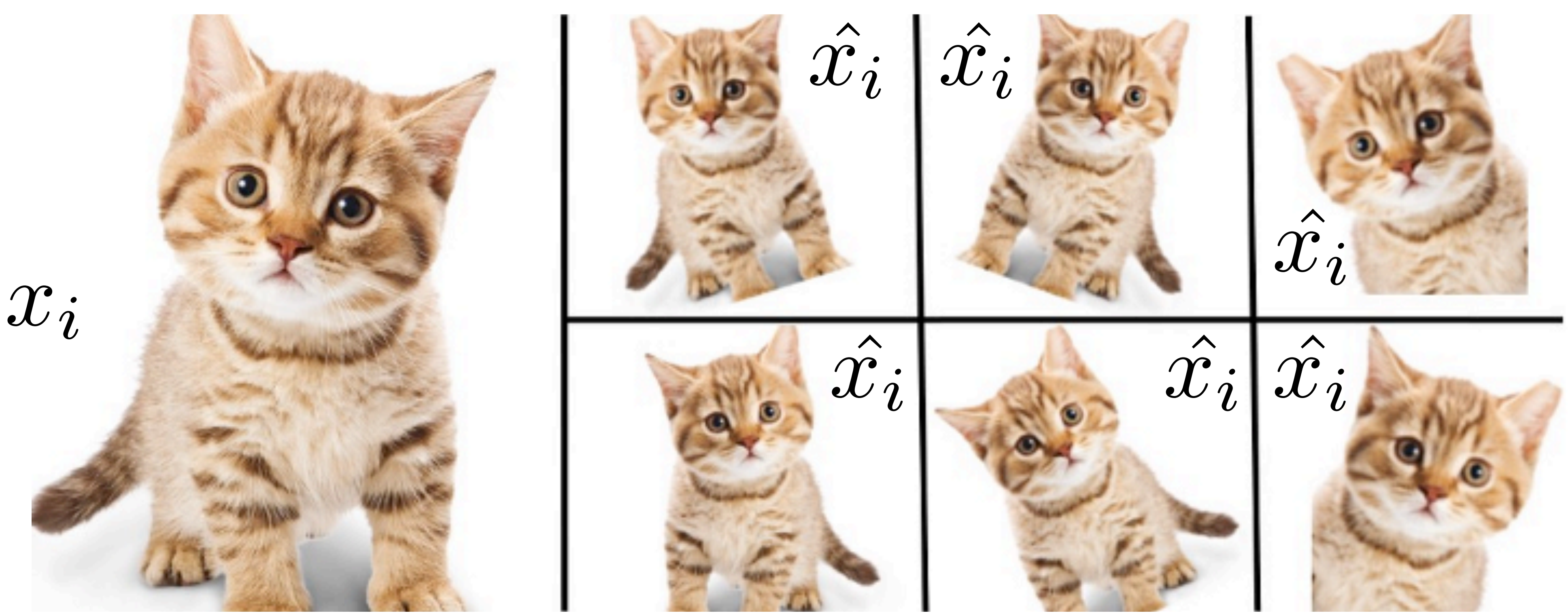
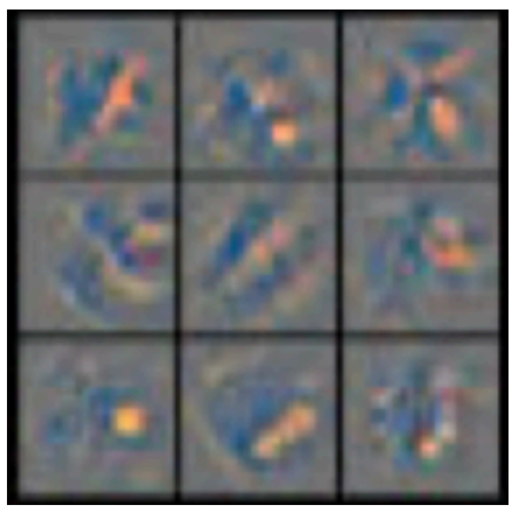


Image from nanonets.com

A history of Convolutional neural networks (CNN)
What does AlexNet learn?

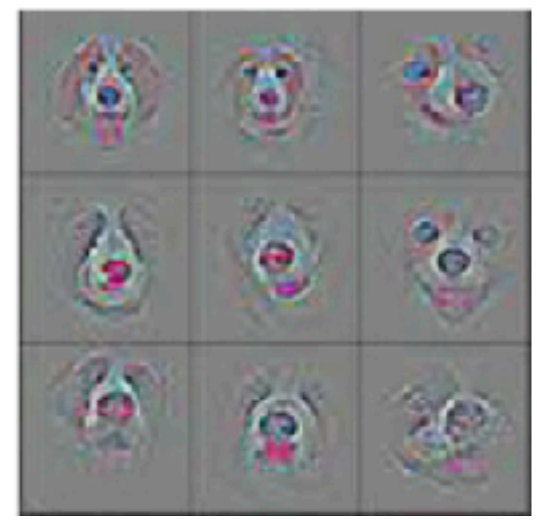
Sample convolution filters learned:

Layer 1

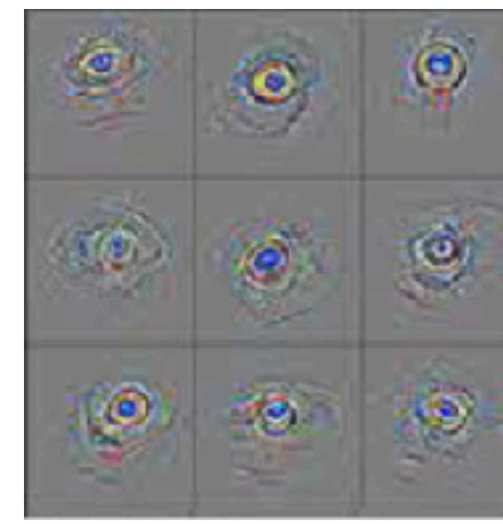


Layer 2

Layer 3



Layer 4

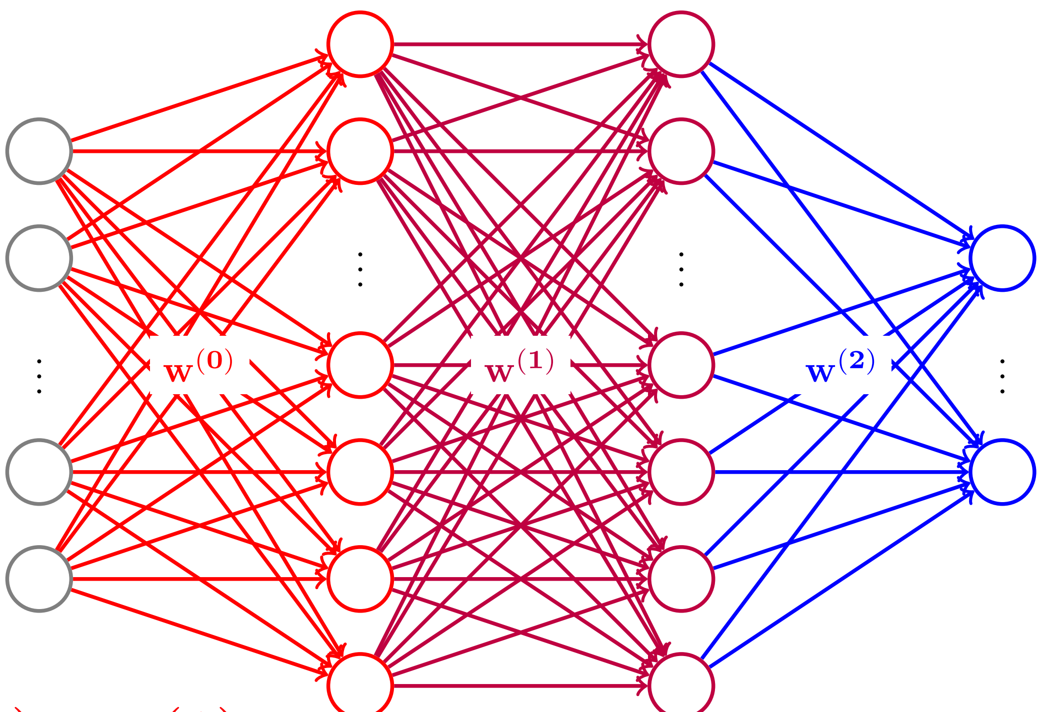


Layer 5

Source: [Zeiler & Fergus, 2013]

A history of Convolutional neural networks (CNN)
Deeper and deeper networks

- Deeper networks = higher-level understanding
- Main limitation: vanishing gradients



$$\frac{\partial \mathcal{L}}{\partial w^{(2)}} = \frac{\partial \mathcal{L}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial w^{(2)}}$$

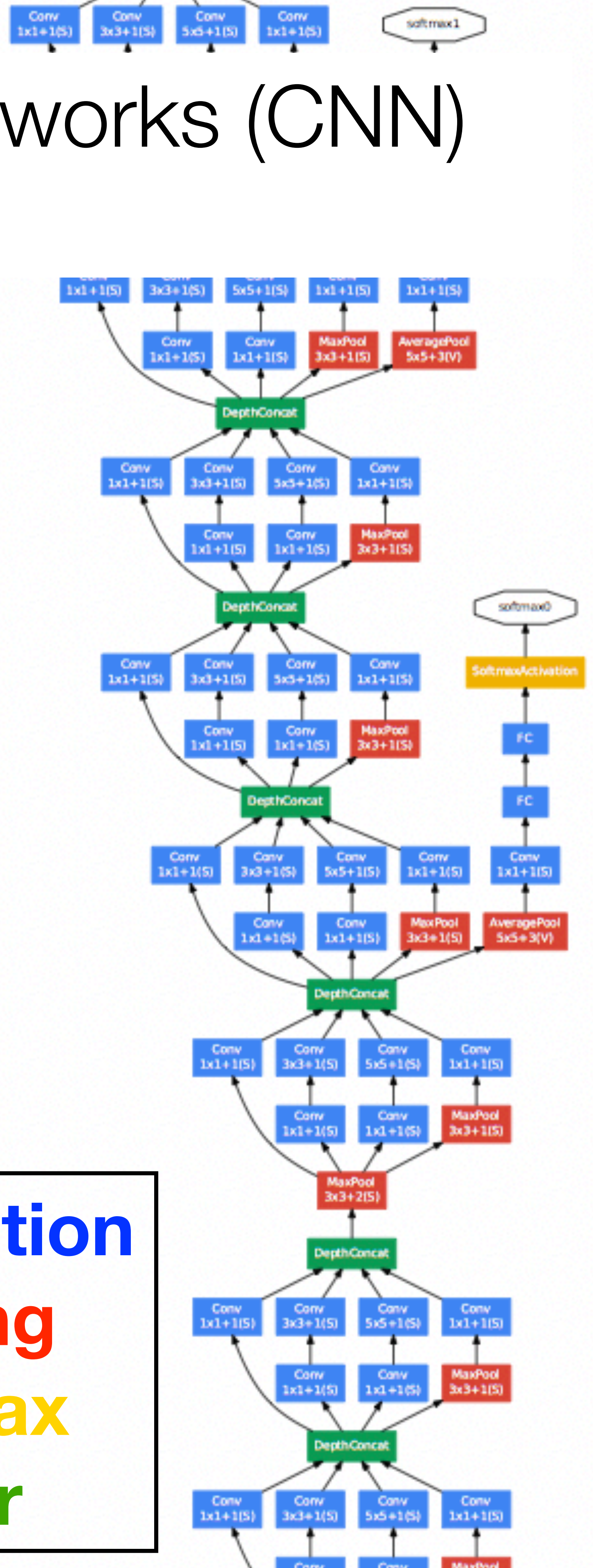
$$\frac{\partial \mathcal{L}}{\partial w^{(0)}} = \frac{\partial \mathcal{L}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial o^{(2)}} \frac{\partial o^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial o^{(1)}} \frac{\partial o^{(1)}}{\partial w^{(0)}}$$

$$\frac{\partial a^{(l)}}{\partial o^{(l)}} = \varphi'(o^{(l)})$$

ReLU
as default
activation function

A history of Convolutional neural networks (CNN)
Inception (2014)

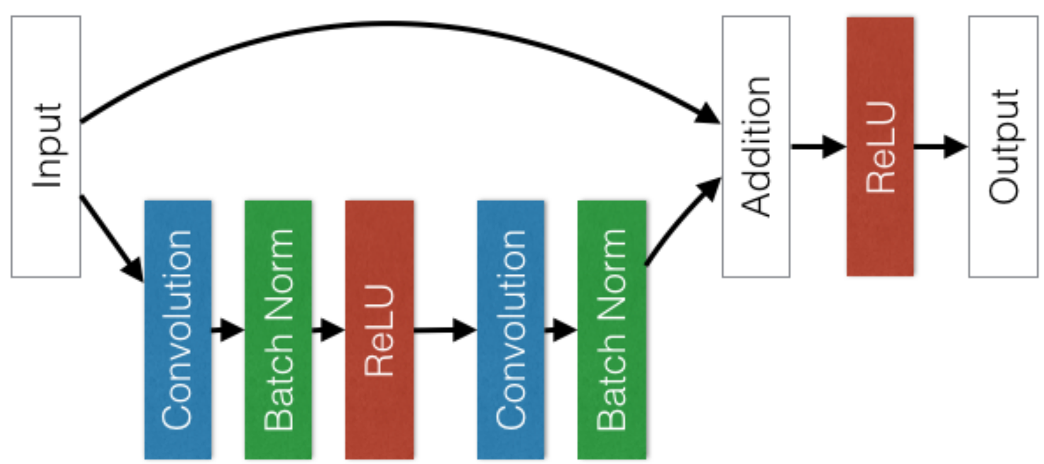
- *Network of networks*
- ~100 blocks, 22 layers
  - Several convolutions per layer
- 5 million parameters
- *Intermediate* classification outputs



Convolution
Pooling
Softmax
Other

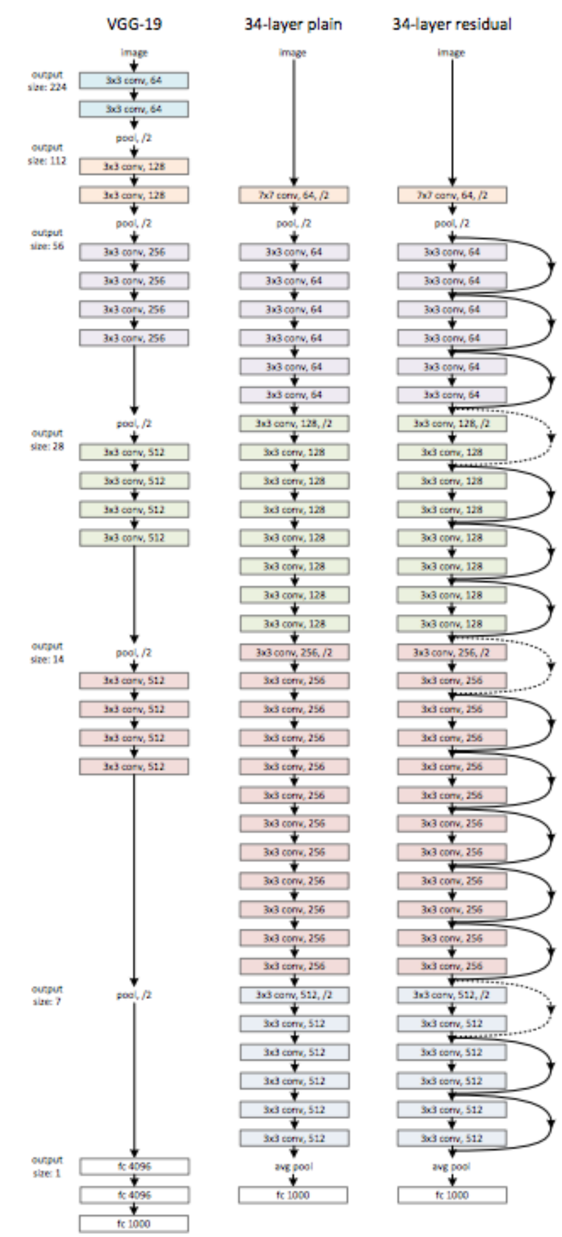
A history of Convolutional neural networks (CNN) Residual Networks (*aka* ResNets)

- Residual connections = Shortcuts in the computational graph



$$a^{(n+1)} = \varphi\left[a^{(n)} + F_{\text{conv}}(a^{(n)})\right]$$

$$\frac{\partial a^{(n+1)}}{\partial a^{(n)}} = \varphi'\left[a^{(n)} + F_{\text{conv}}(a^{(n)})\right] \times \varphi\left[1 + F'_{\text{conv}}(a^{(n)})\right]$$



[He *et al.*, 2016]

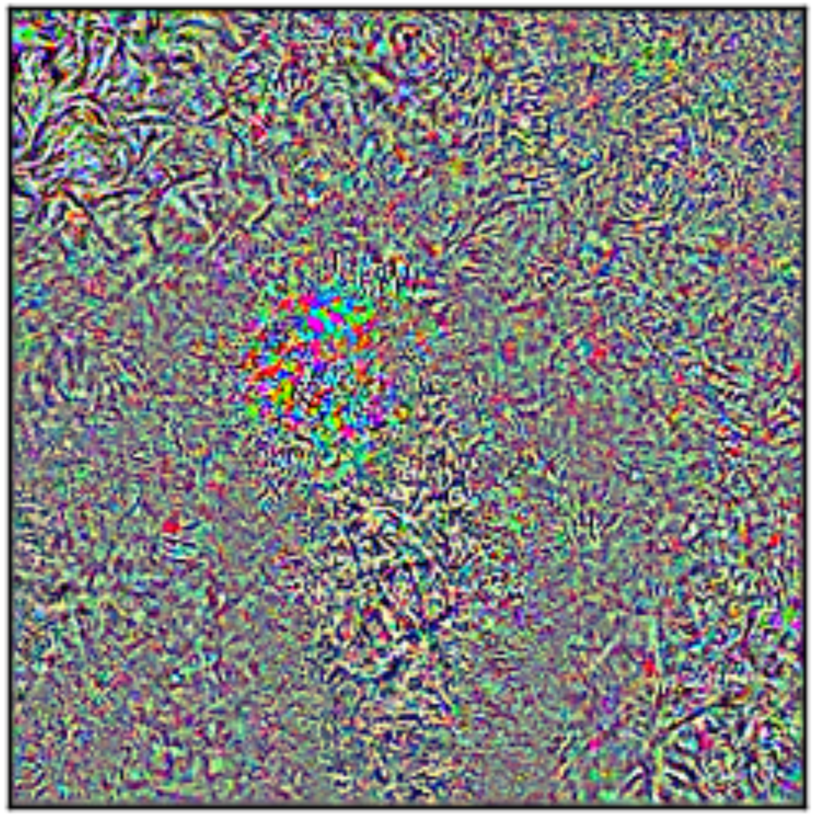
Why such sudden changes?

- Big data (ImageNet & co)
- Big infrastructures (GPU)
- Optimization
  - Algorithms
  - *Tricks* (initialization, regularization, fighting vanishing gradients)
- Automatic differentiation libraries (tensorflow, pytorch, …)

Limitations of current models
Adversarial examples

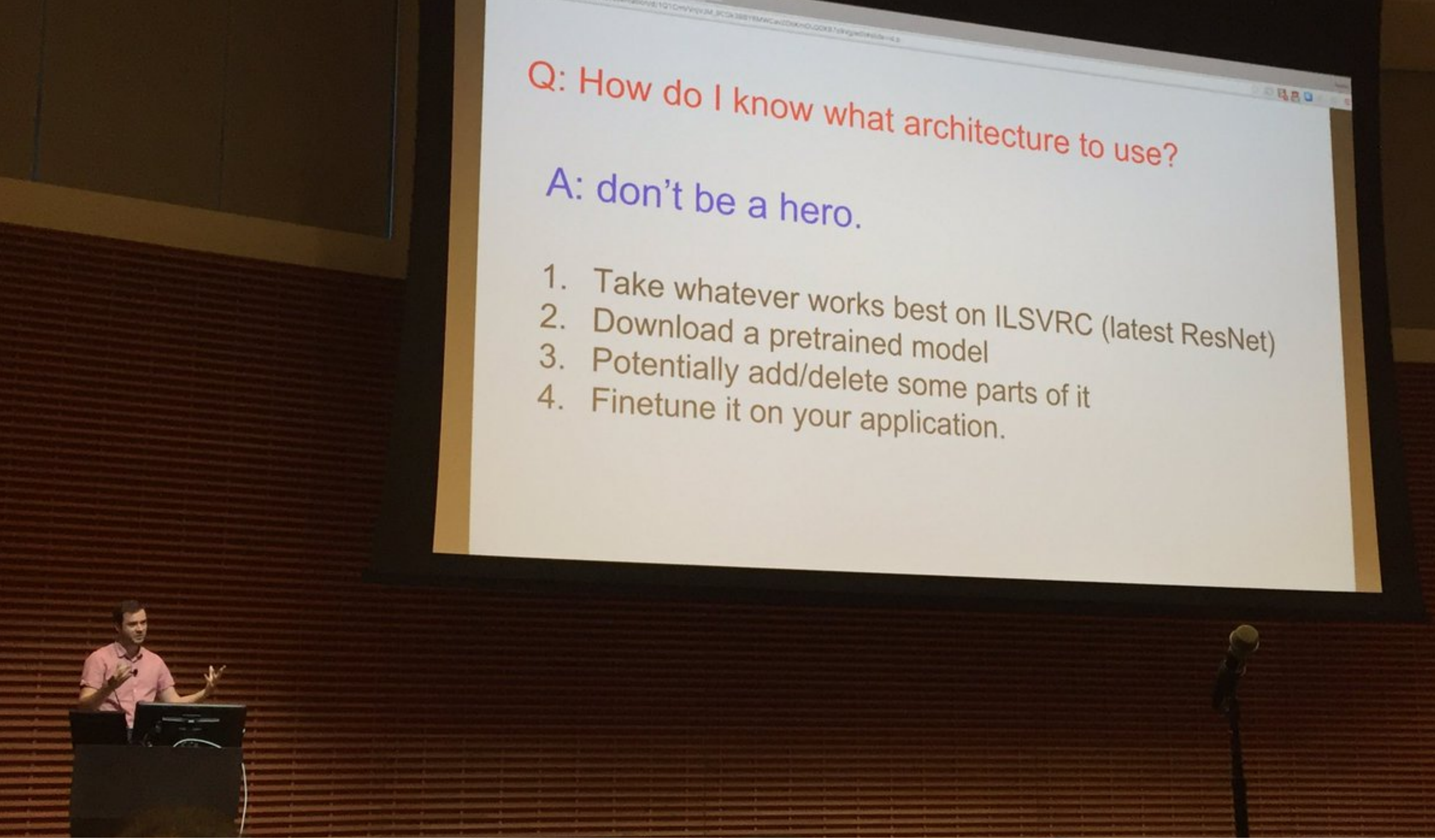
Original image:
Ara : 97%

Transformed image:
Ara : 0%
Bookshelf : 99%



Amplified noise

Source : github.com/Hvass-Labs/TensorFlow-Tutorials



Andrej Karpathy,
Deep Learning Summer School,
2016