Artificial Intelligence

Deep Learning and Convolutional Neural Networks

Ing. Adam Ligocki

Course Supervisor: doc. Ing. Václav Jirsík, CSc.

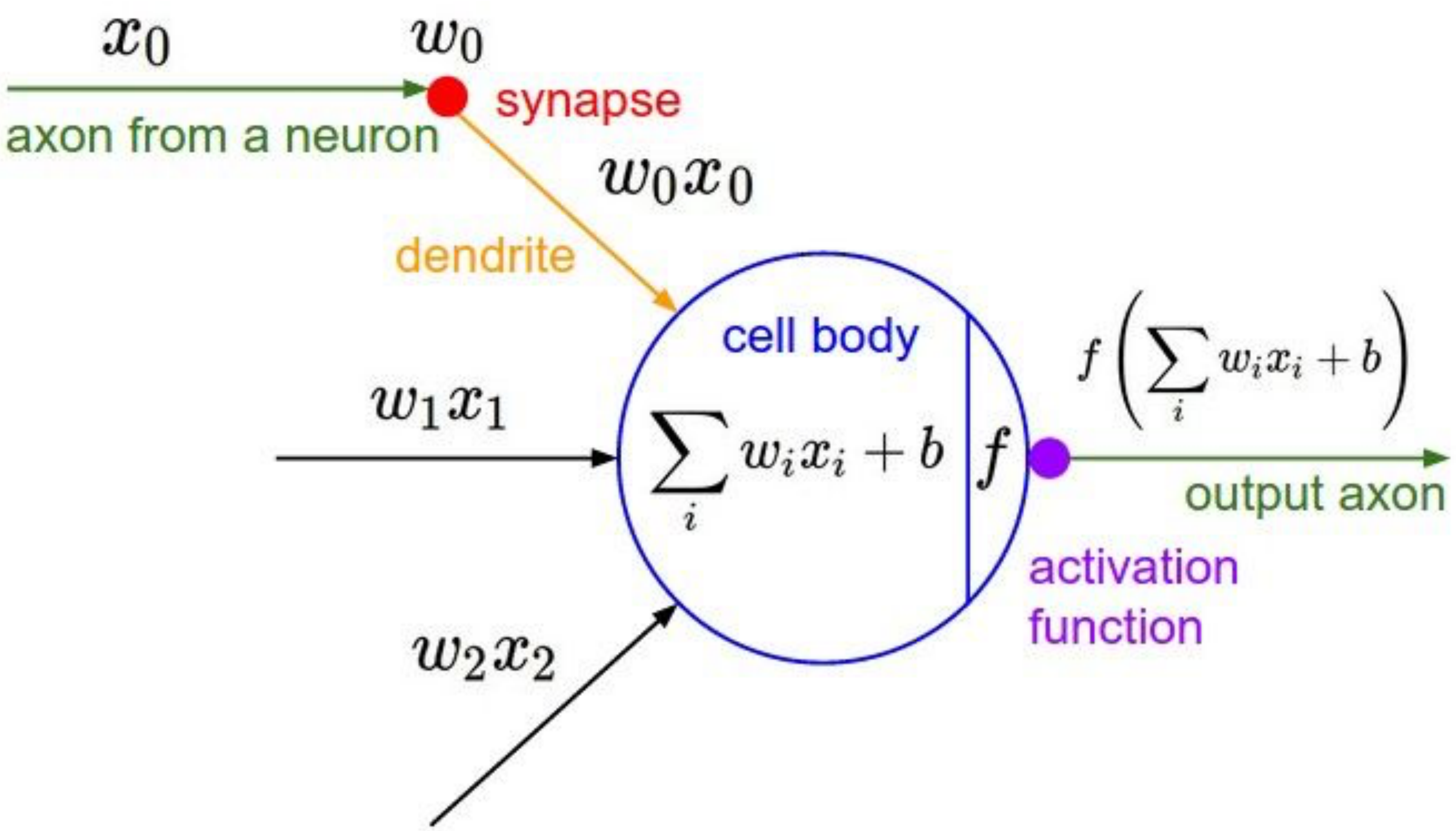
Fully Connected Neural Network

Fully Connected Neural Network

- Single neuron represents a simple linear equation

$$y = f(w_1 x_1 + \ldots + w_i x_i + b)$$

- The neural network is a complex combination of these simple mathematical models

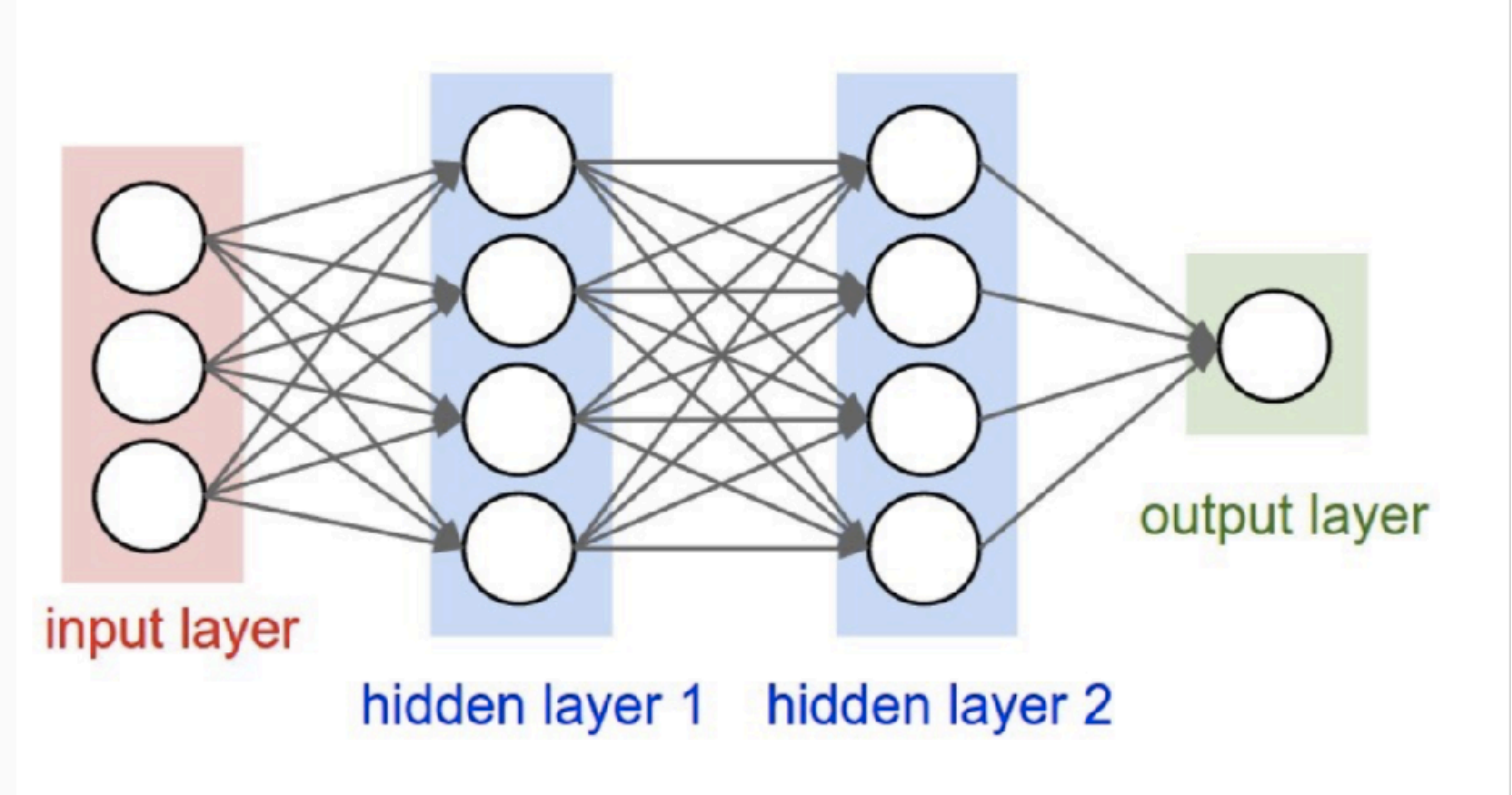


Fully Connected Neural Network

- Neural Networks are a kind of matrix multiplication representation

$$Y_{10x1} = w_{10x100} * X_{100x1}$$

- Generally in Machine learning this technique is called "Linear Classifier"



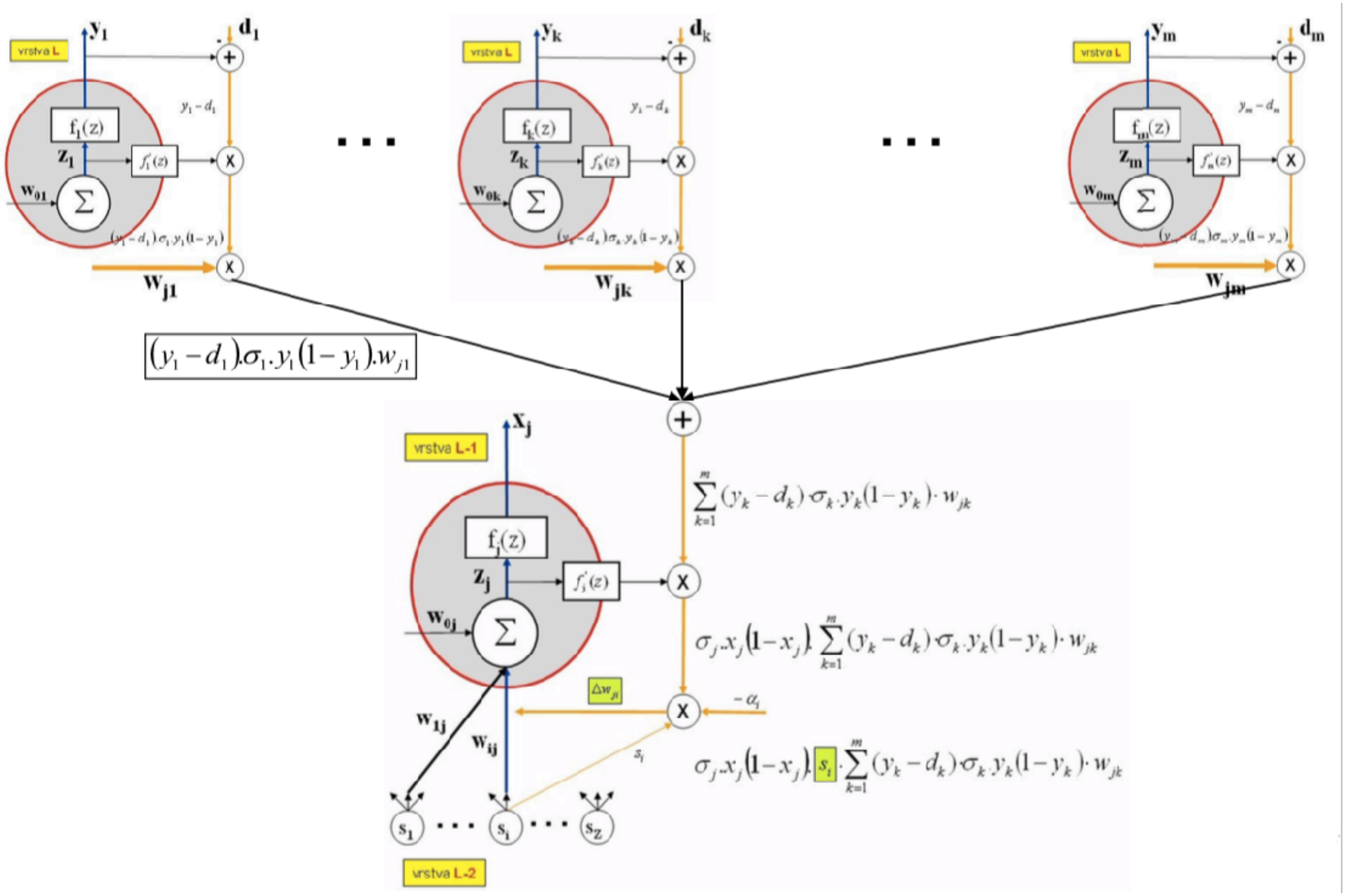
Back Propagation

Back Propagation

- Back propag. is an algorithm to iteratively adjust weights w.r.t. total error to get better result in the next step

$$\Delta W_{jk} = -\alpha \frac{\Delta E_c}{\Delta w_{jk}} = -\alpha \sum_{s=1}^{p} \frac{\Delta E_s}{\Delta w_{jk}}$$

- The better result in the next step does not mean better result at the end of the learning



Chain Rule

$$\frac{\Delta E_s}{\Delta w_{jk}} = \frac{\Delta E_s}{\Delta y_k}\frac{\Delta y_k}{\Delta z_k}\frac{\Delta z_k}{\Delta w_{jk}}$$

$$z_k = \sum_{j=1}^{N} x_j w_{jk} \qquad \frac{\Delta z_k}{\Delta w_{jk}} = x_j$$

$$E_s = \frac{1}{2}\sum_{j=1}^{m}(y_{sj} - d_{sj})^2 \qquad \frac{\Delta E_s}{\Delta y_k} = y_k - d_k$$

$$y_k = \frac{1}{1 + e^{-\sigma_k z_k}} \qquad \frac{\Delta y_k}{\Delta z_k} = \sigma_k * y_k(1 - y_k)$$

$$\frac{\Delta E_s}{\Delta w_{jk}} = (y_k - d_k)\sigma_k y_k(1 - y_k)x_j$$

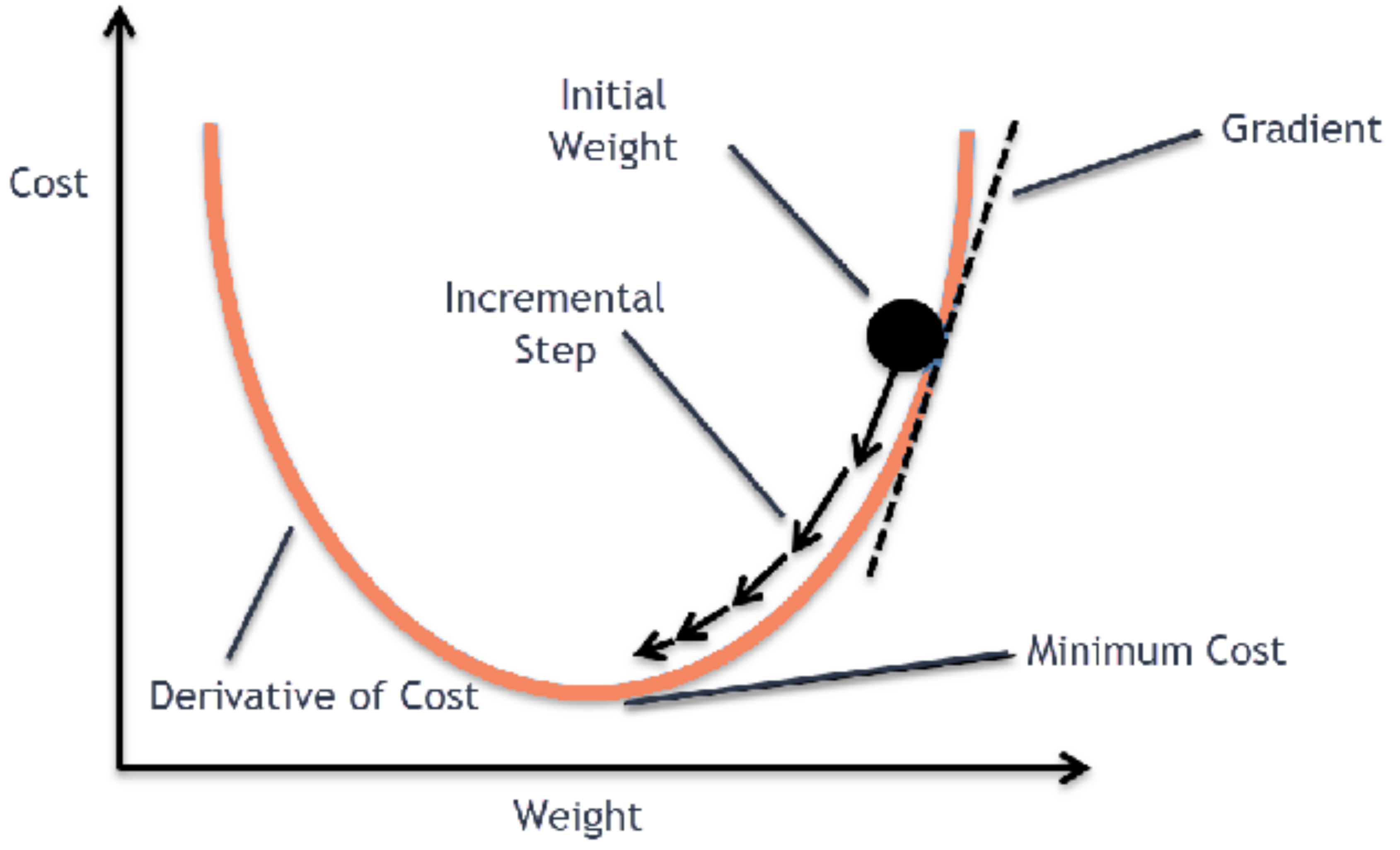
$$\frac{\Delta \sigma(x)}{\Delta x} = \sigma(x) * (1 - \sigma(x))$$

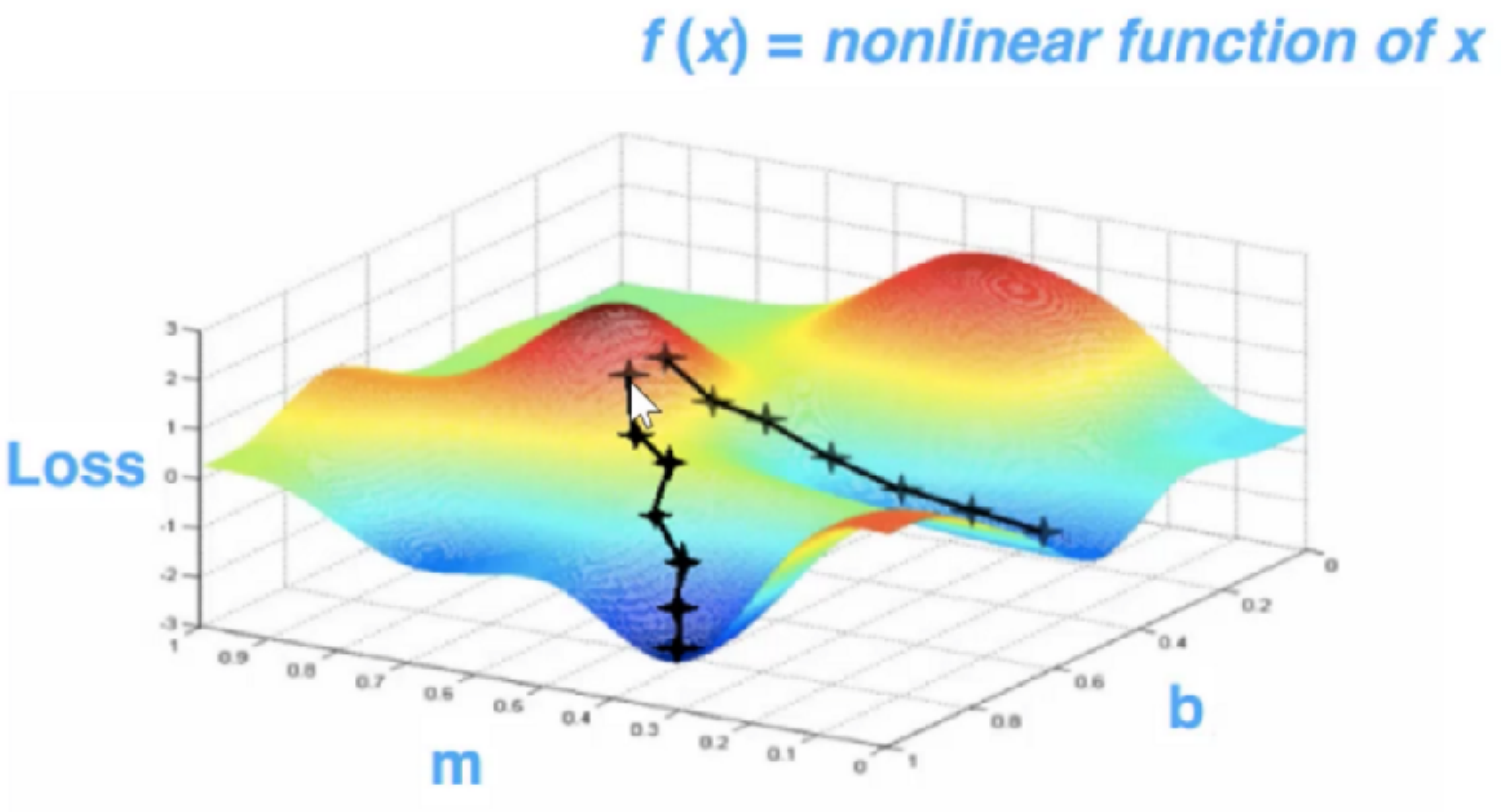
Stochastic Gradient Descent

- Common FCNN has ~1000 weights
  -> calculate partial derivation of matrix of size 1000x1000

- Common CNN has ~10 000 000 weights
  -> Mission Impossible

- SGD is an idea of learning neurone by using only small number of randomly chosen partial derivatives of the Total Error w.r.t. weights

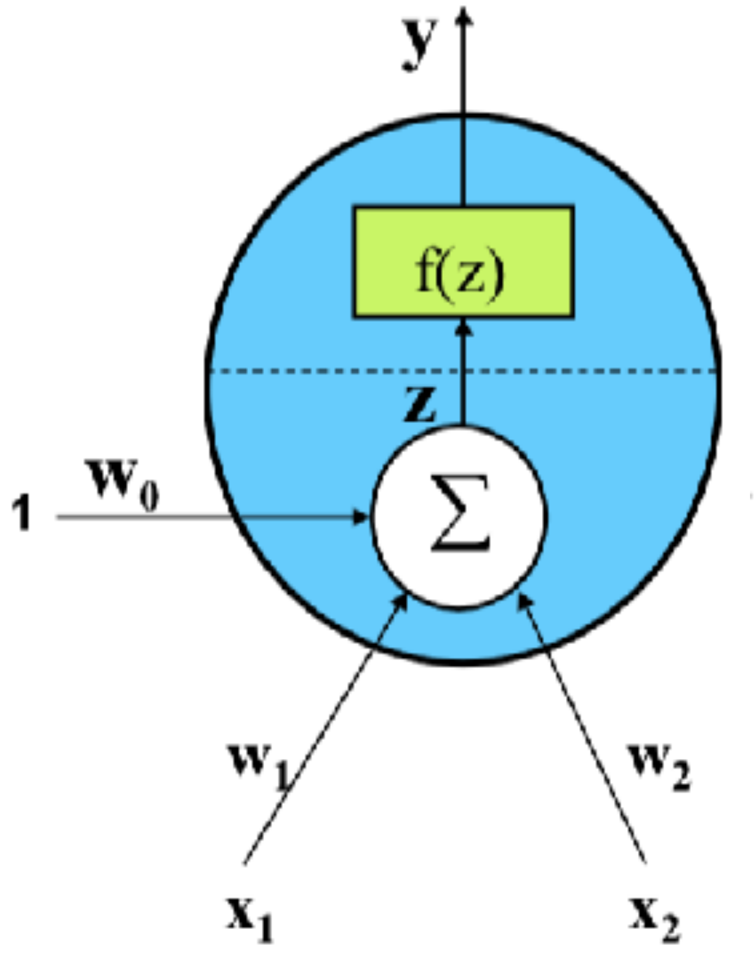


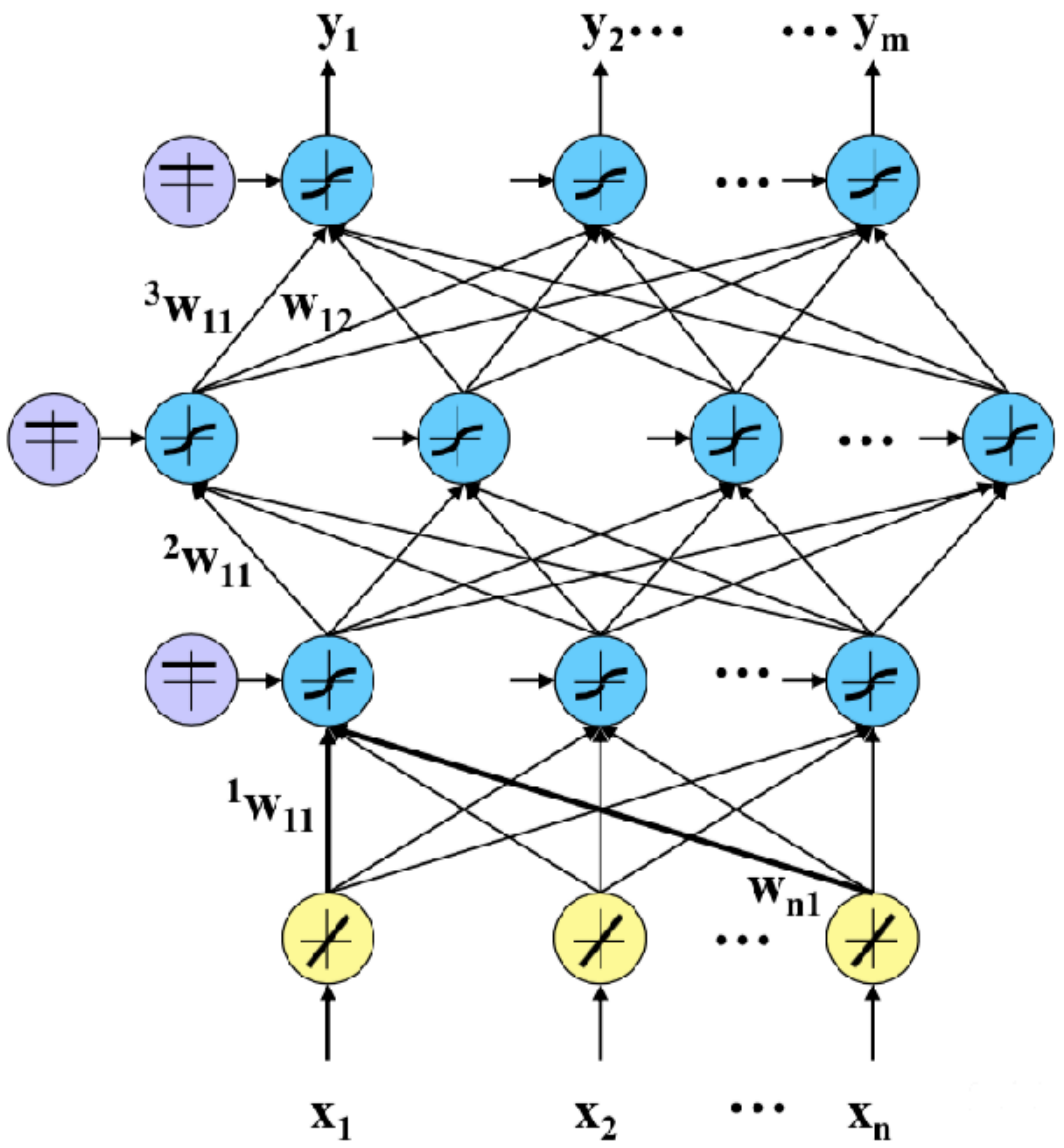


History

Perceptron and Fully Connected NN

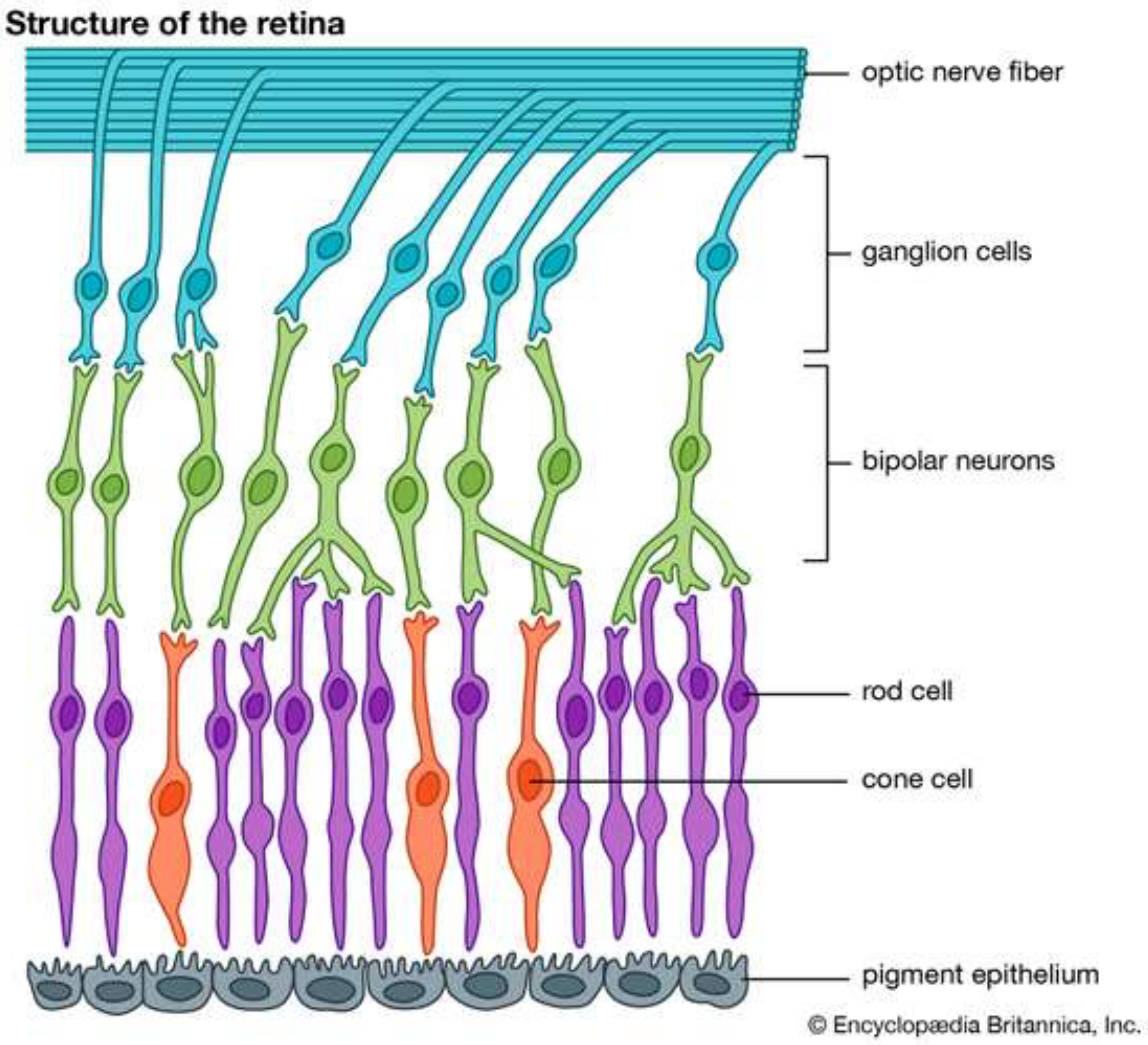
- 1943 - Mathematical model of neuron (Warren McCulloch and Walter Pitts)
- 1958 - Perceptron discovery (Frank Rosenblatt)
- 1986 - Back Propag discovery





Eye as a CNN

- Mammal eye consists out of several types of light-sensitive cells and signal processing neutrons.
- These neurones performs spacial differential functions over incoming signal from rods and cones
- For more informations search for “Retina structure” on Google



https://www.britannica.com/science/retinitis-pigmentosa

LeNet (1998)

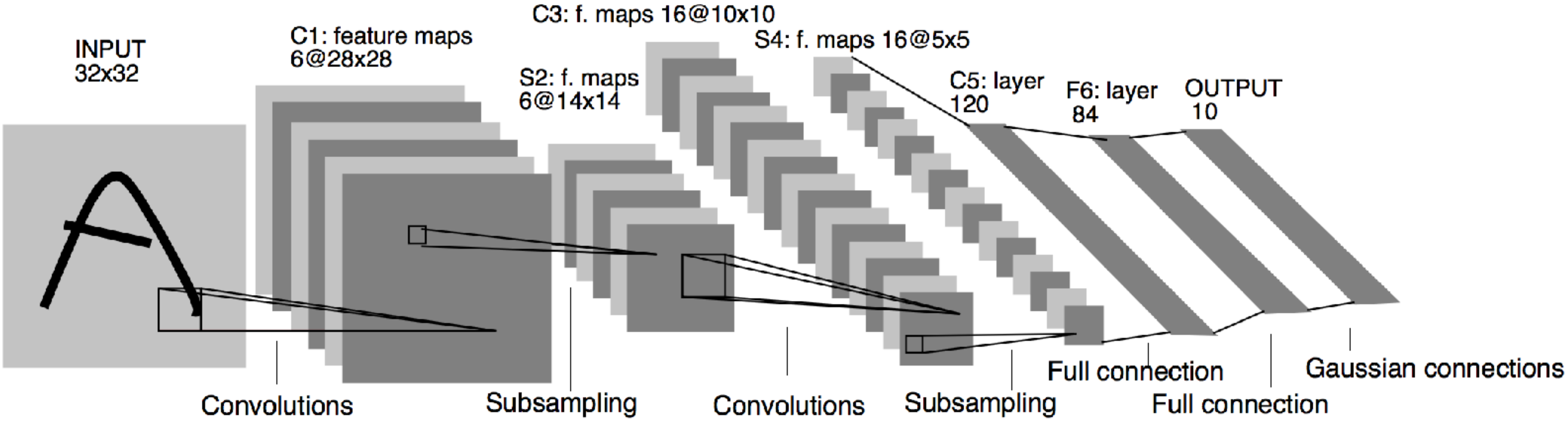
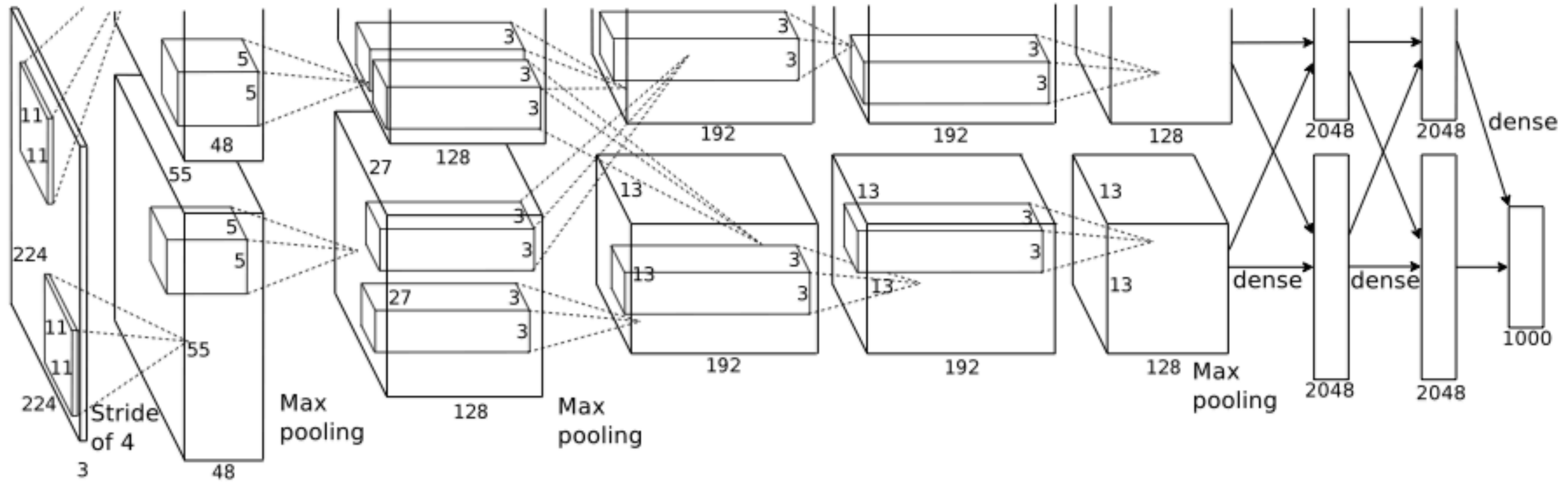


Image from original paper Y. LeCun 1998

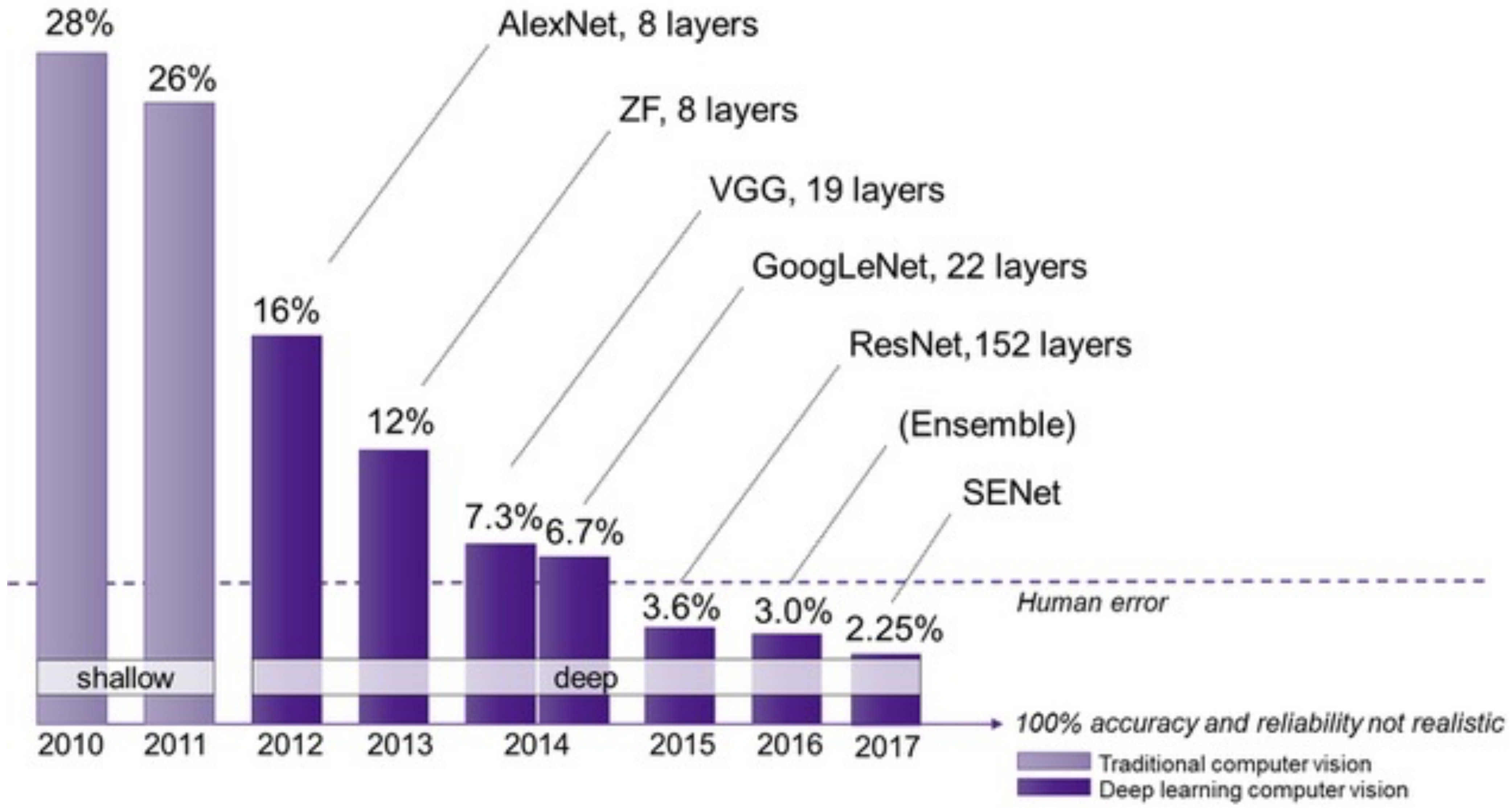
- Simple characters and numbers classification
- First ever “True Convolutional NN”

AlexNet (2012)

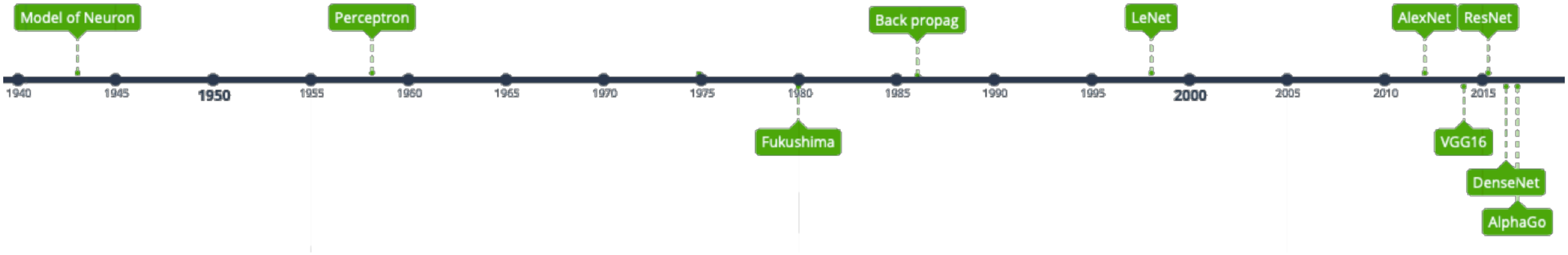


- First time ever the CNN outperformed other CV techniques

- The beginning of intensive research of the CNN topic



Timeline

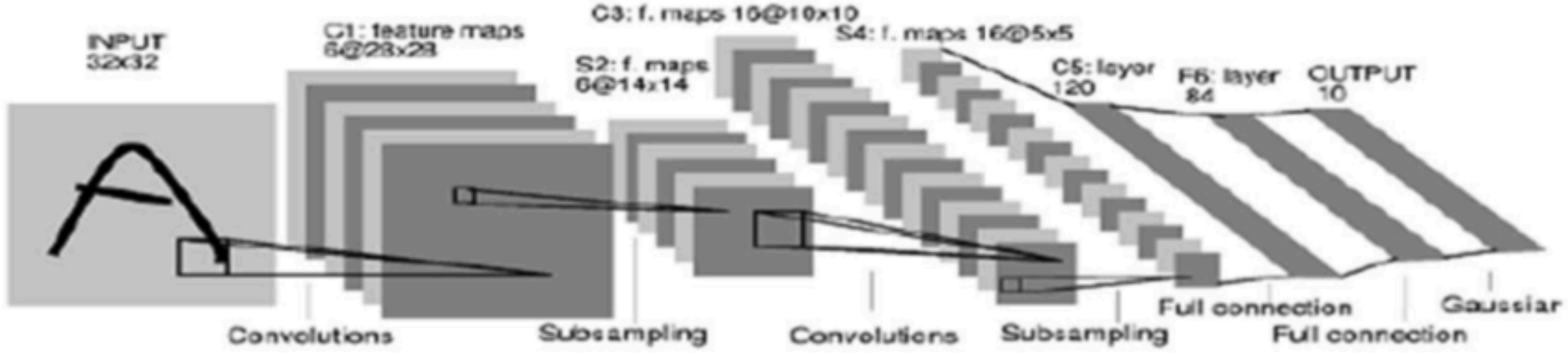


Model of Neuron
Perceptron
Back propag
LeNet
AlexNet
ResNet
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
Fukushima
VGG16
DenseNet
AlphaGo

Why the AlexNet started it all?

1998

LeCun et al.



# of transistors

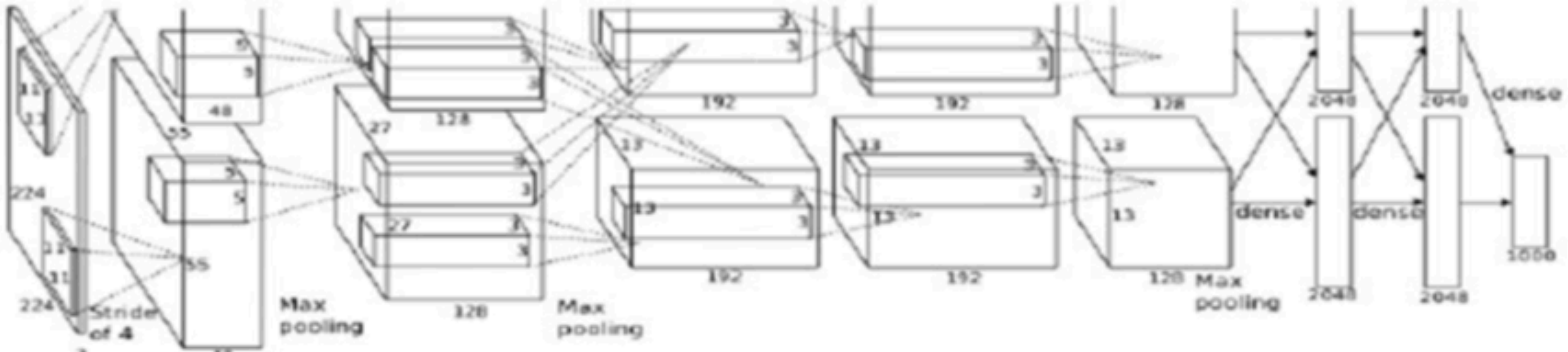
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



# of transistors

10^9

GPUs

# of pixels used in training

10^{14} IMAGENET

Convolutional Neural Network

Convolution in Computer Vision

- Convolution is a mathematical operation which combines features of two input signals into single result

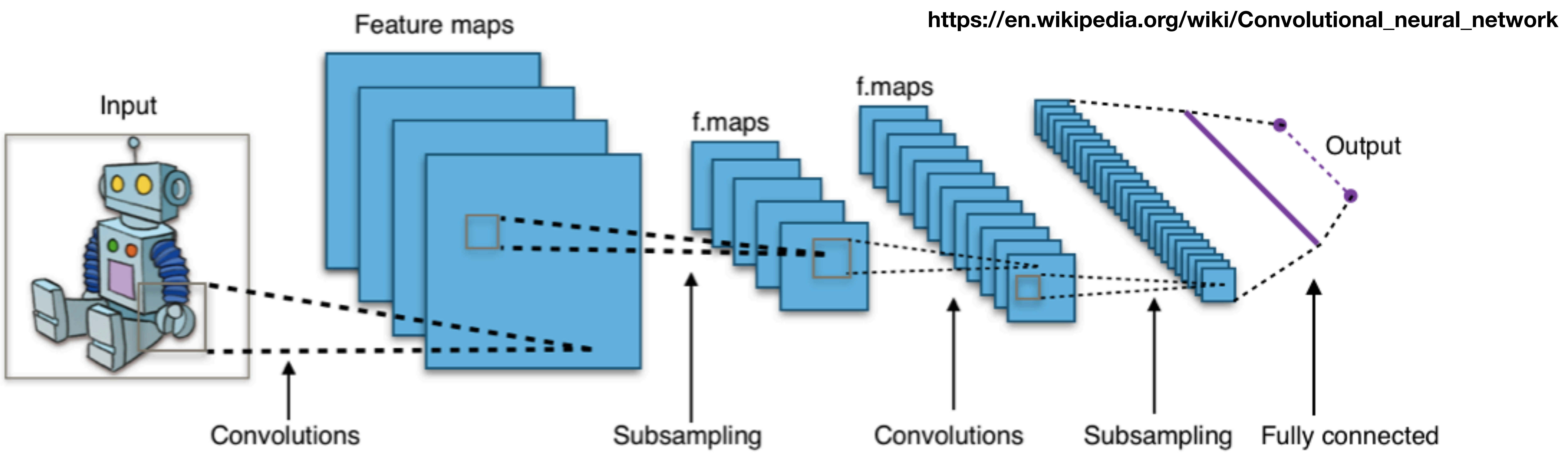
- In CV the 2D convolution is tho mostly used as a blurring (suppressing high freq) and edge (suppressing low freq) filters

- Convolution kernel size is usually of size 3x3 up to 11x11

$$\begin{bmatrix} 7 & 2 & 3 & 3 & 8 \\ 4 & 5 & 3 & 8 & 4 \\ 3 & 3 & 2 & 8 & 4 \\ 2 & 8 & 7 & 2 & 7 \\ 5 & 4 & 4 & 5 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 6 & -9 & -8 \\ -3 & -2 & -3 \\ -3 & 0 & -2 \end{bmatrix}$$

https://www.learnopencv.com/image-classification-using-convolutional-neural-networks-in-keras/

Principle

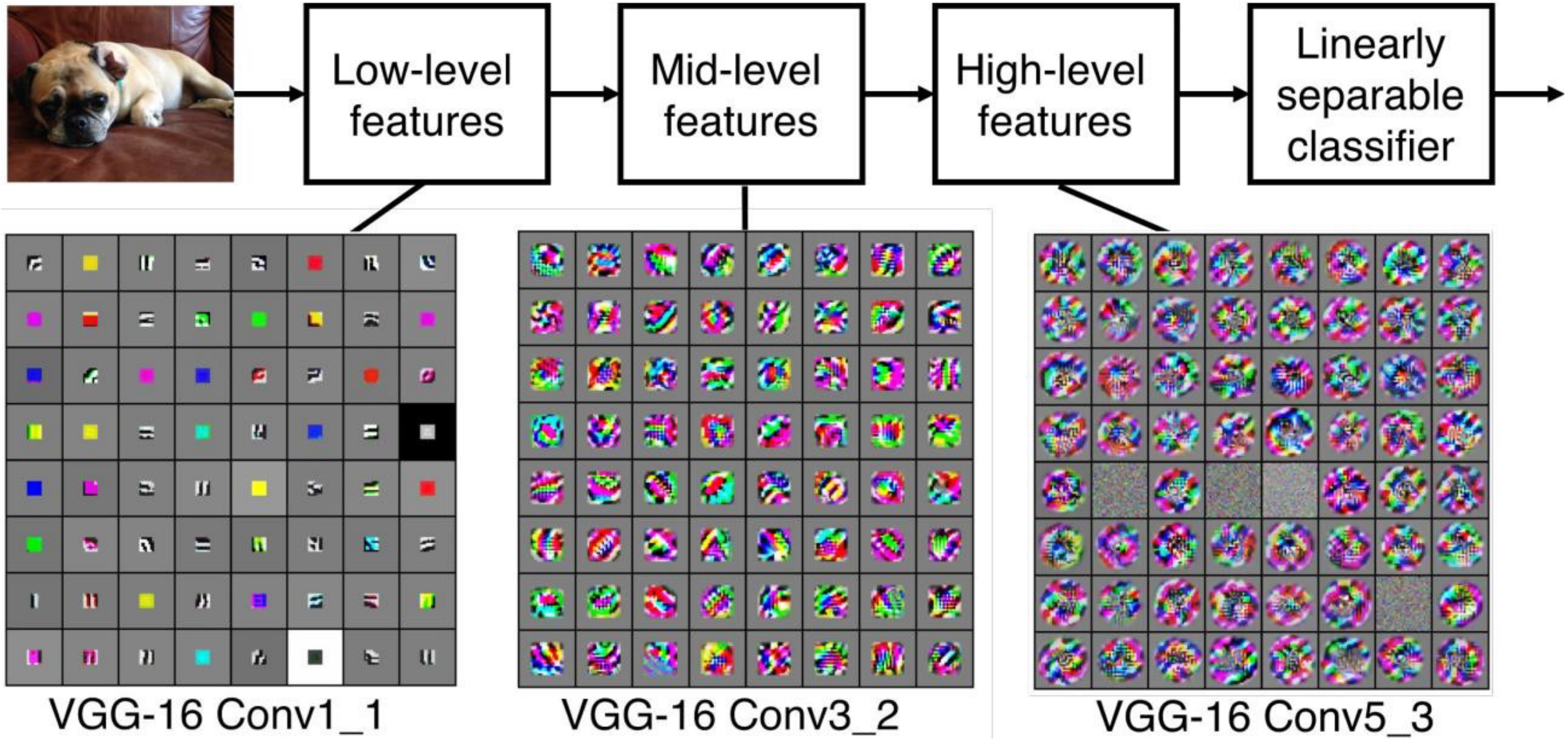


- CNN is based on the idea of the feature extraction from the input image by using large number of small convolution kernels

- Low layers extracts simple geometrical shapes (usually edge detectors), higher layers detects more complex structures

- The output of the highest layer is processed by FC NN classifier

Kernel Visualizations



Convolutional Neural Network

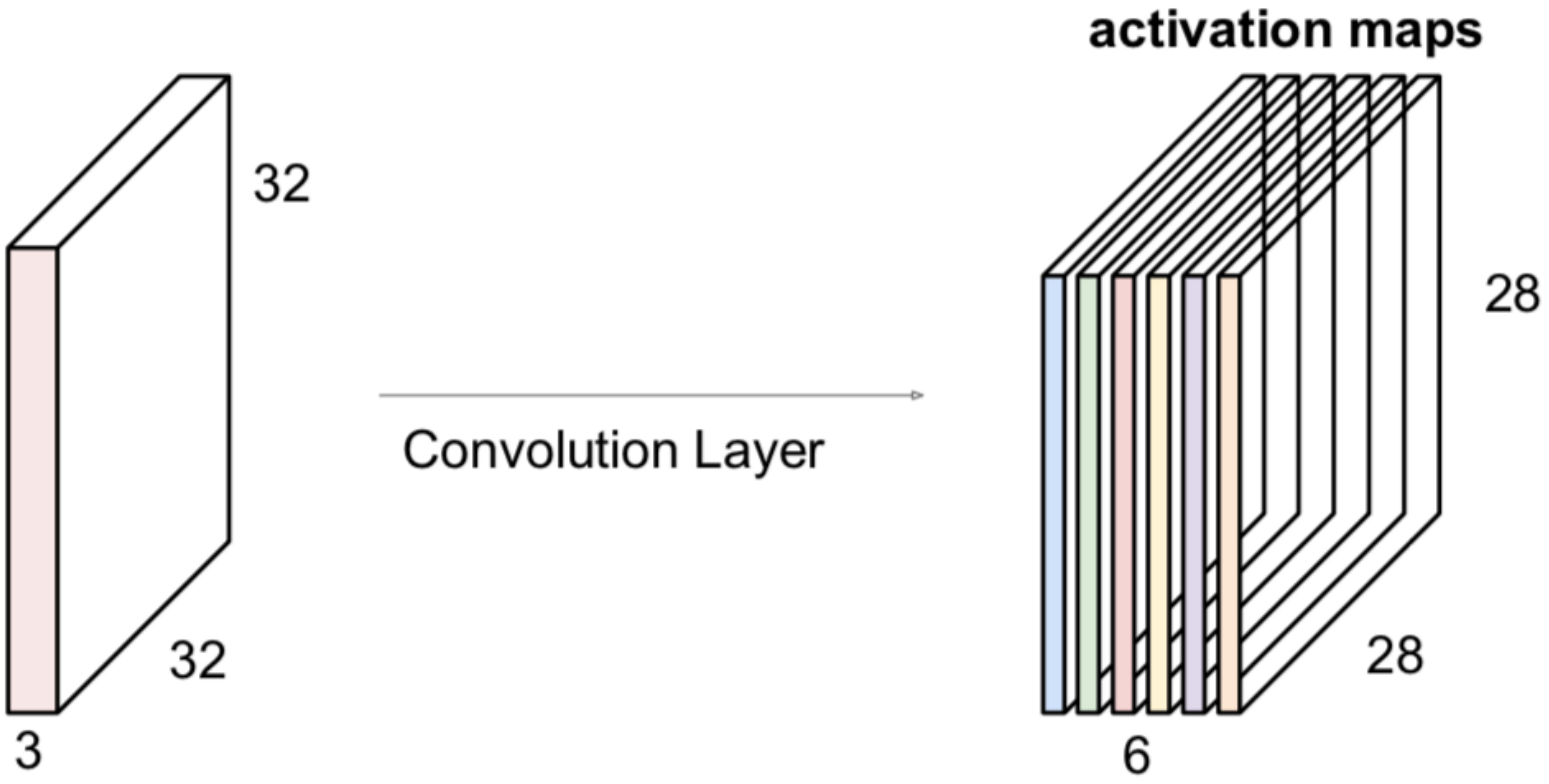
Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

Convolutional Neural Network

consider a second, green filter

Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

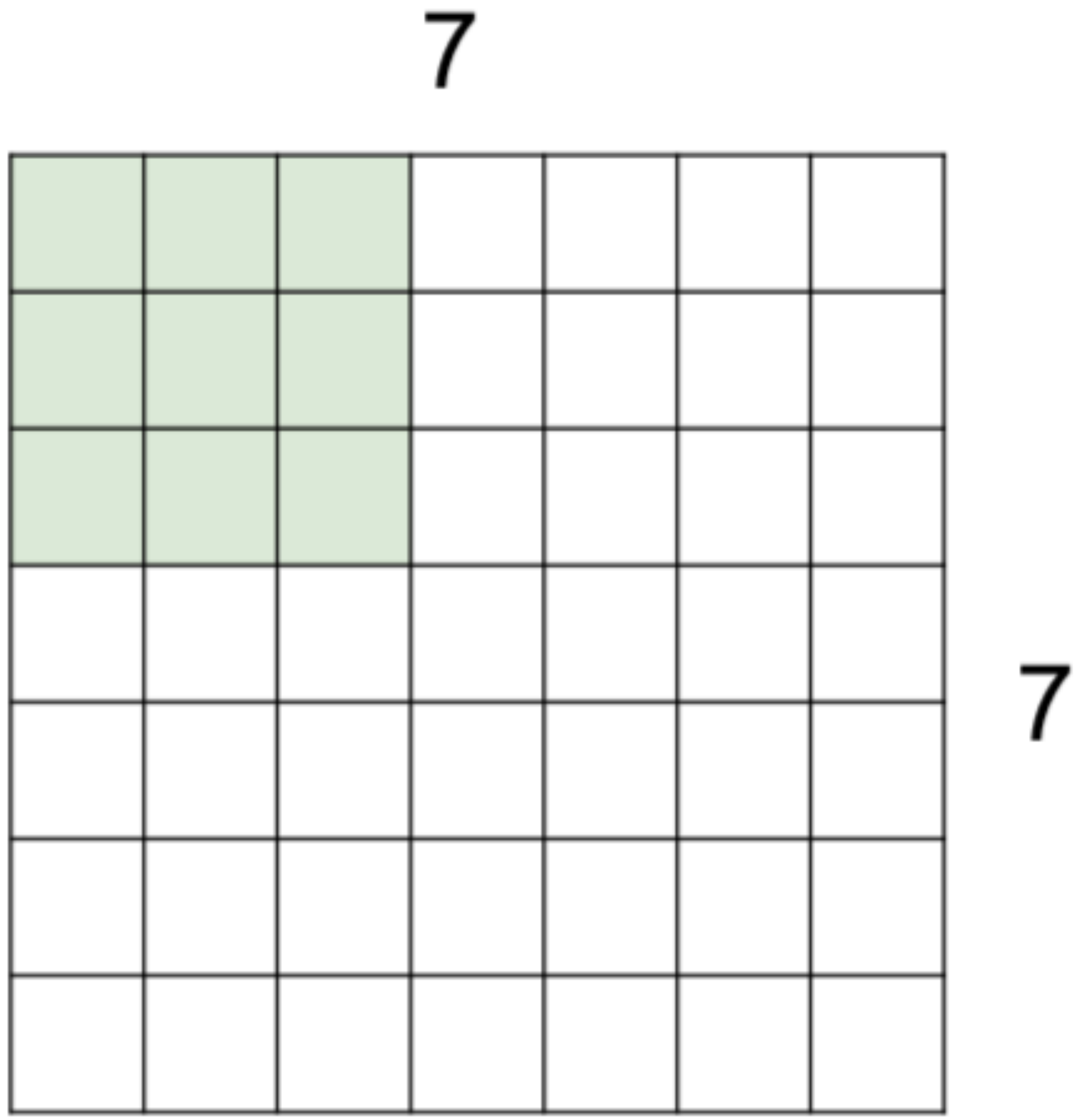
Convolutional Neural Network



We stack these up to get a “new image” of size 28x28x6!

Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

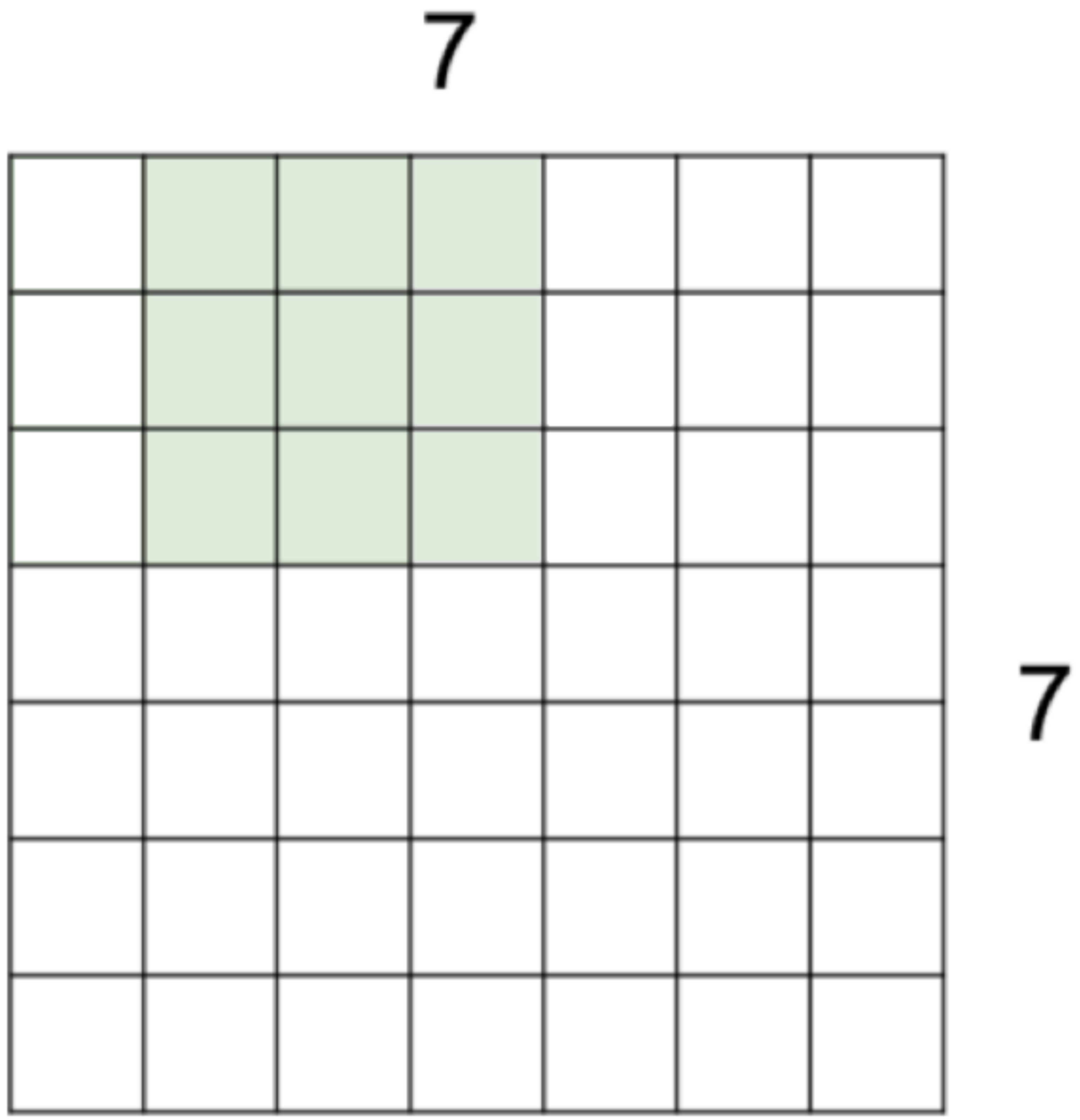
Convolutional Neural Network



7x7 input (spatially)
assume 3x3 filter

Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

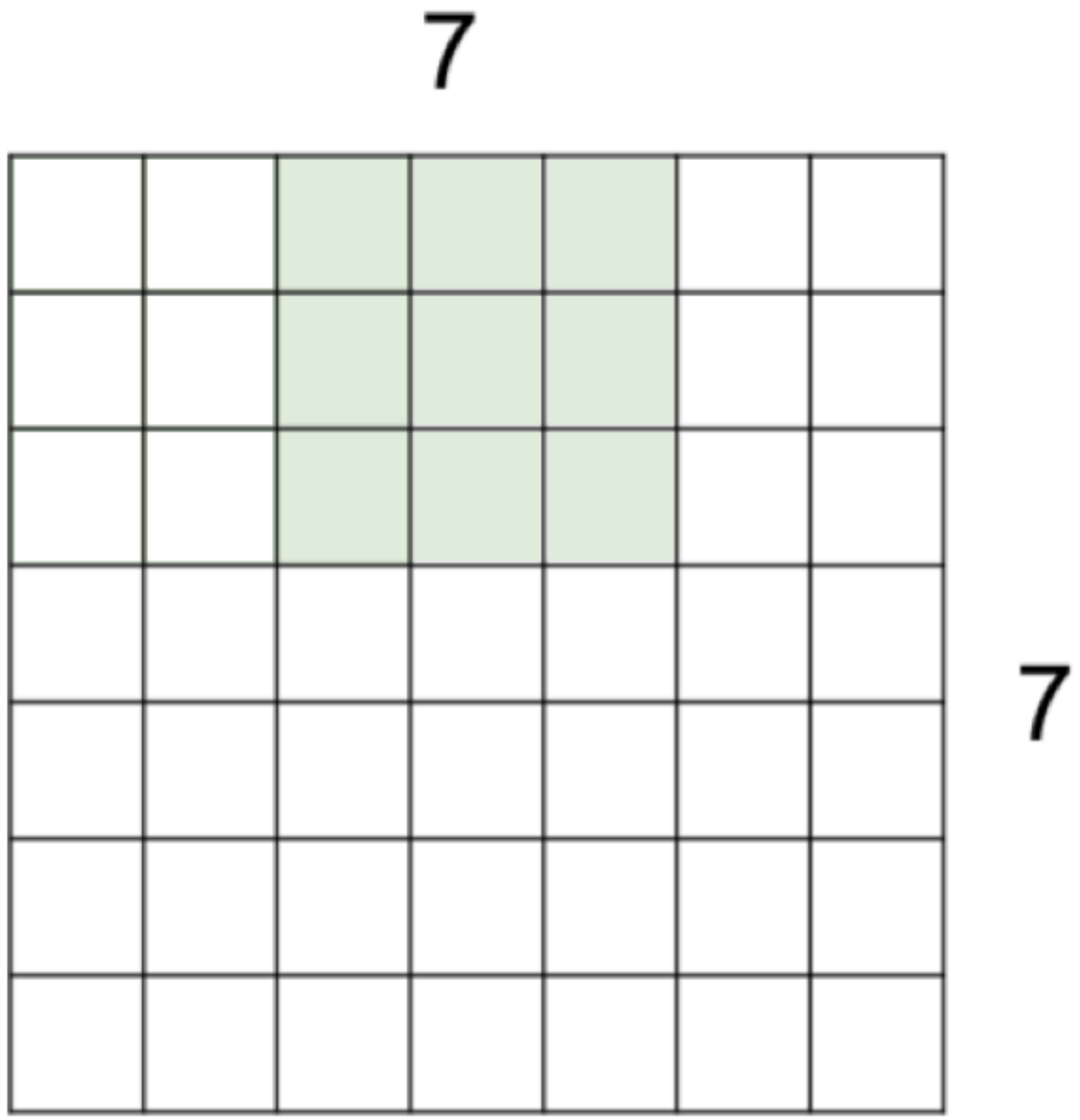
Convolutional Neural Network



7x7 input (spatially)
assume 3x3 filter

Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

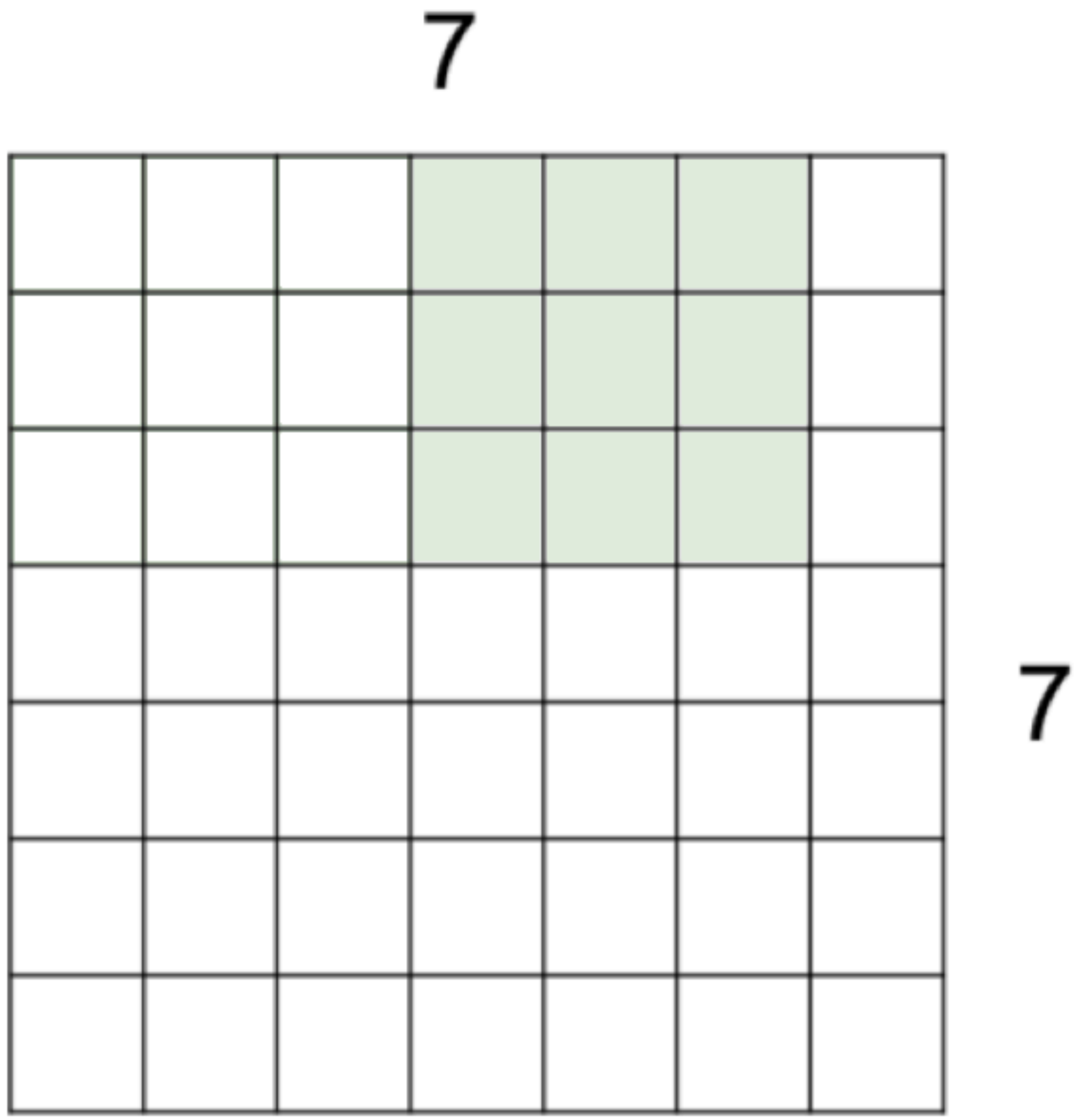
Convolutional Neural Network



7x7 input (spatially)
assume 3x3 filter

Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

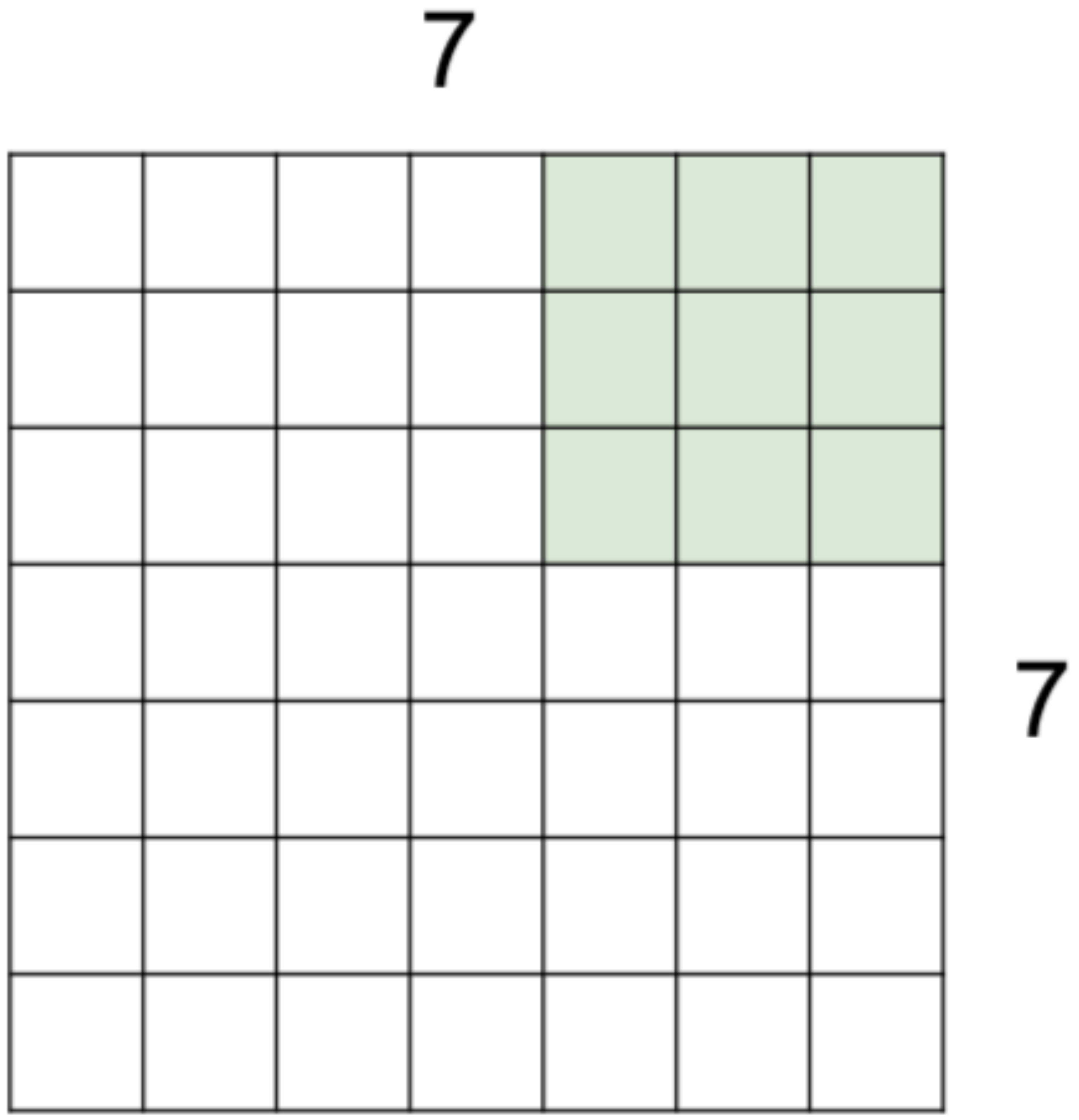
Convolutional Neural Network



7x7 input (spatially)
assume 3x3 filter

Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

Convolutional Neural Network

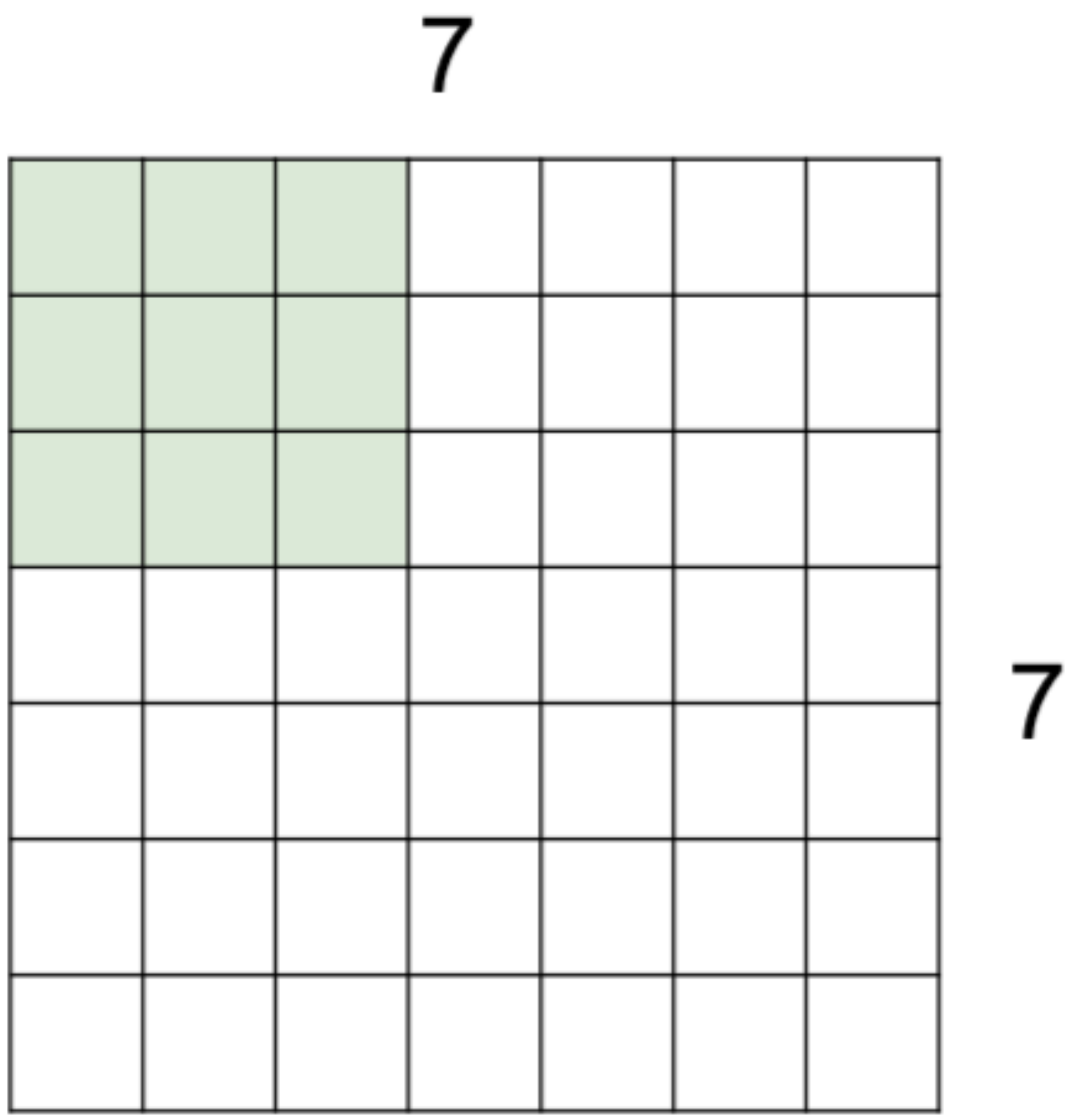


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

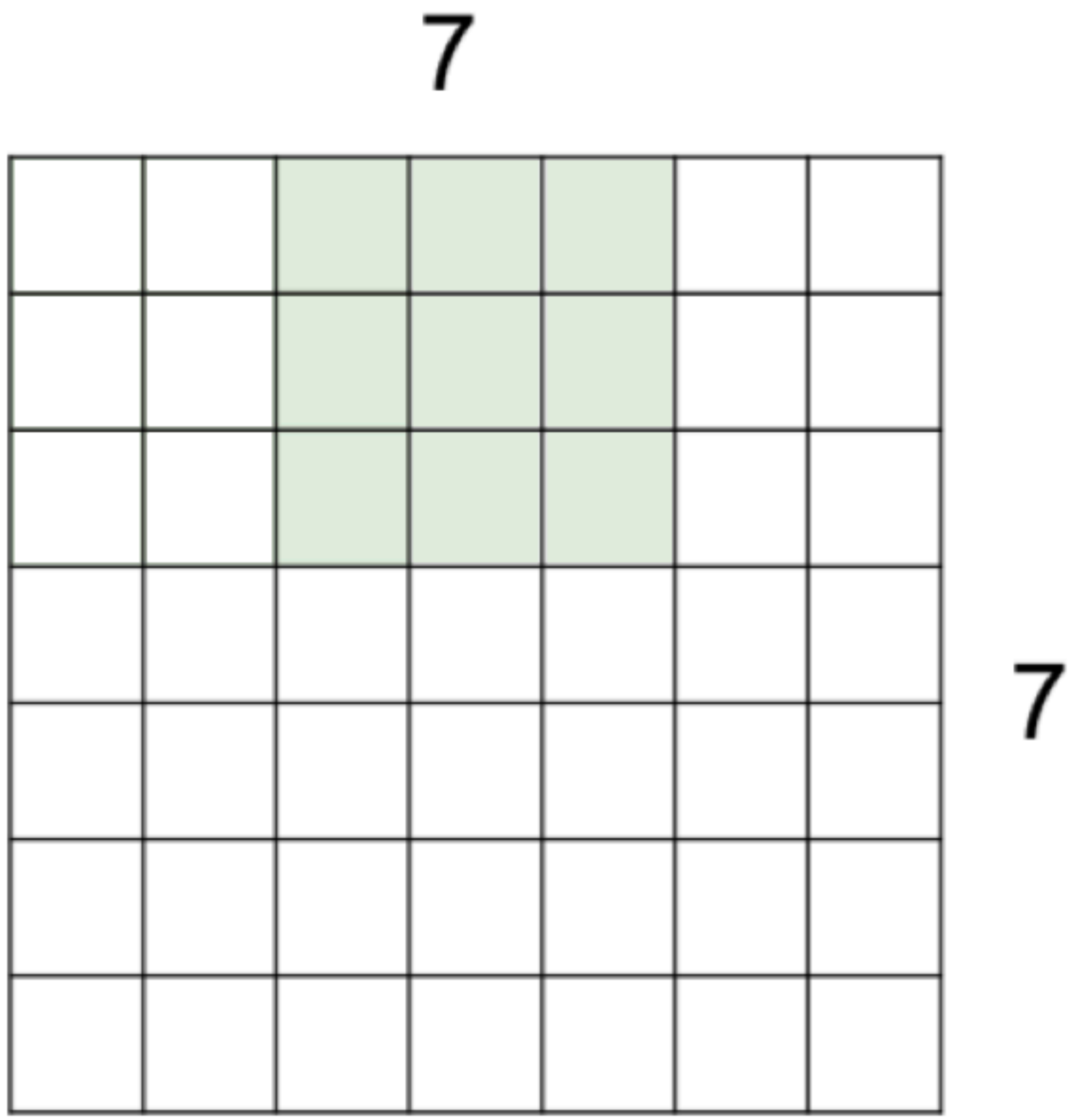
Convolutional Neural Network



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

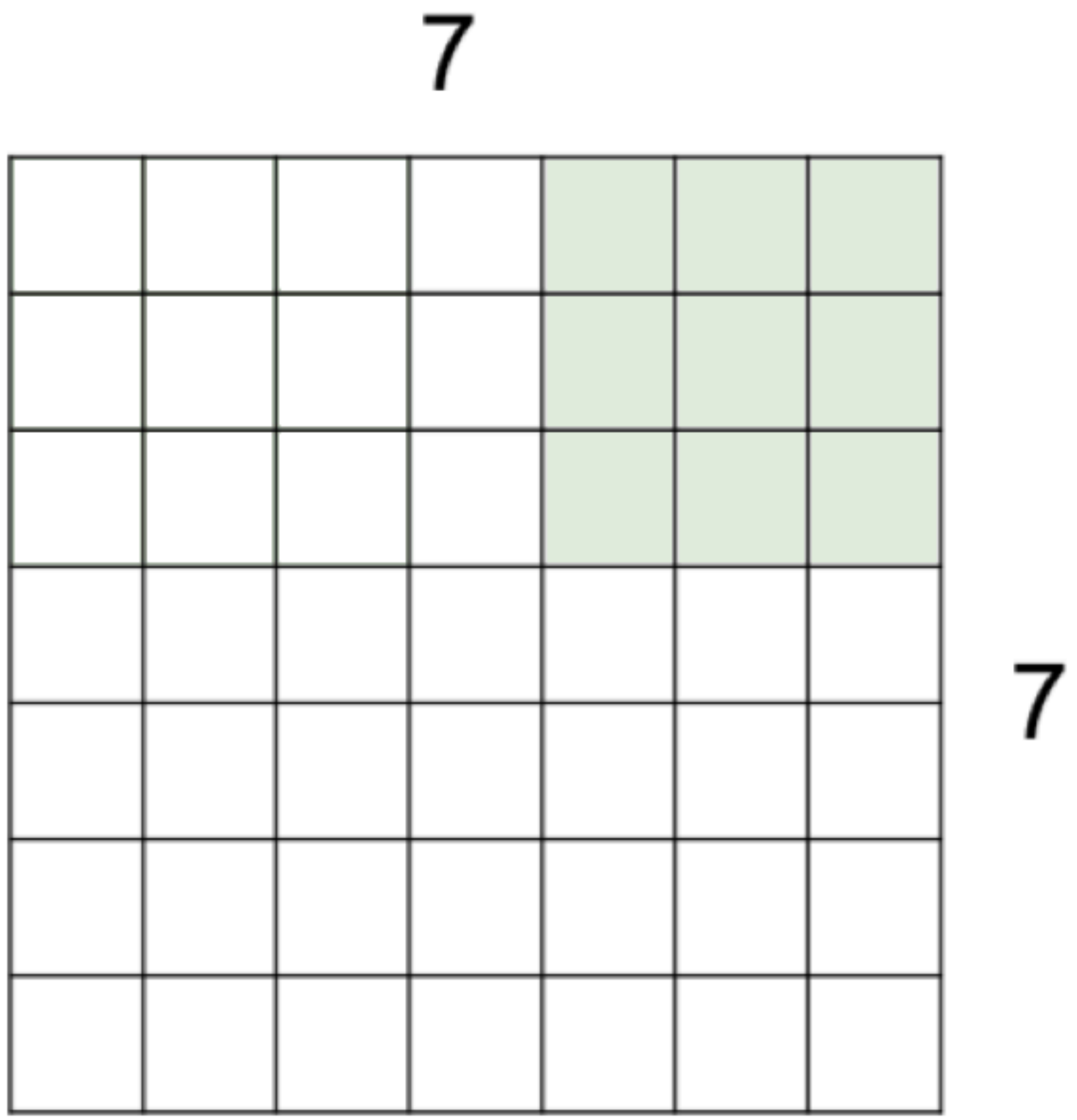
Convolutional Neural Network



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

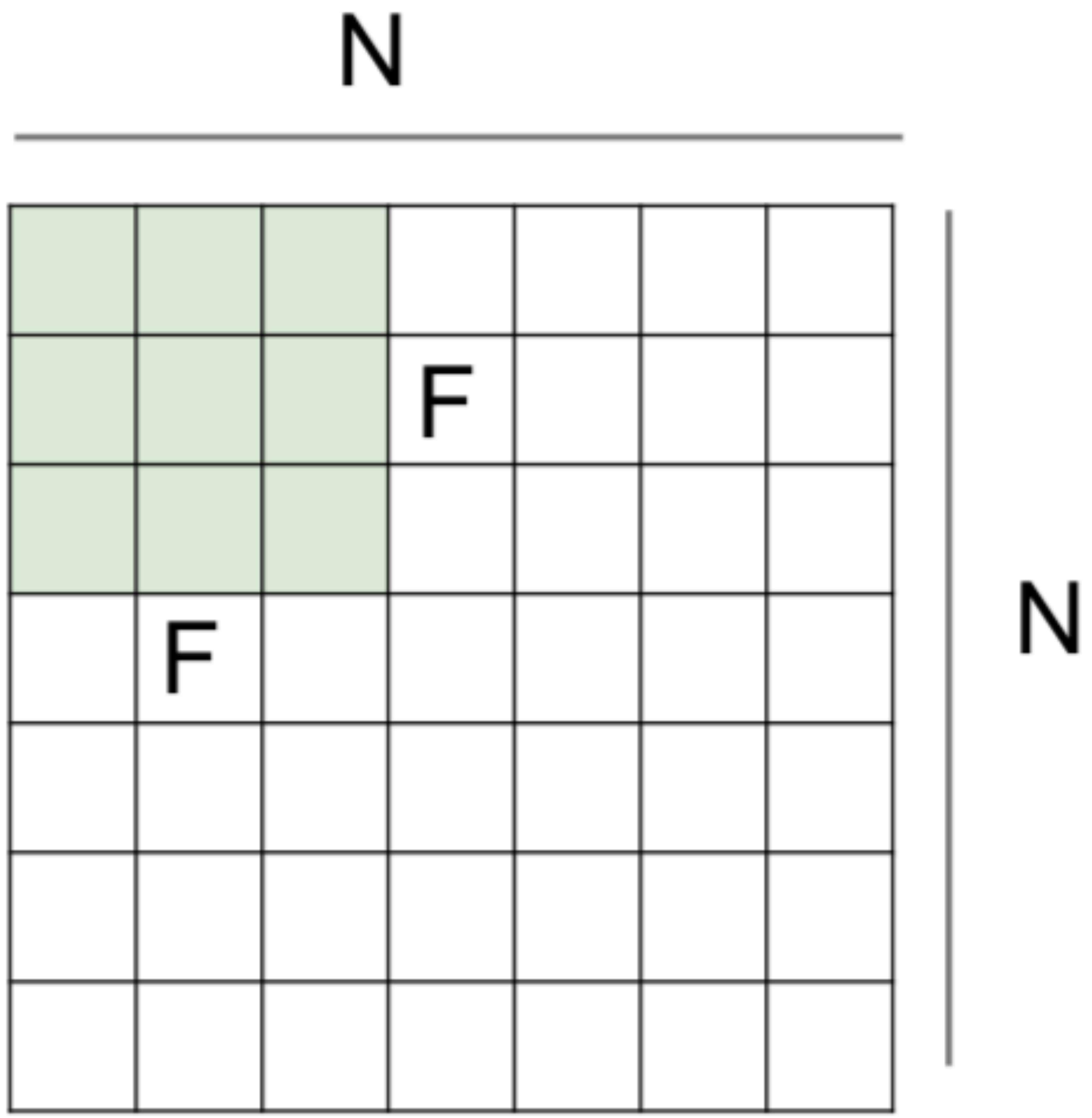
Convolutional Neural Network



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

Convolutional Neural Network

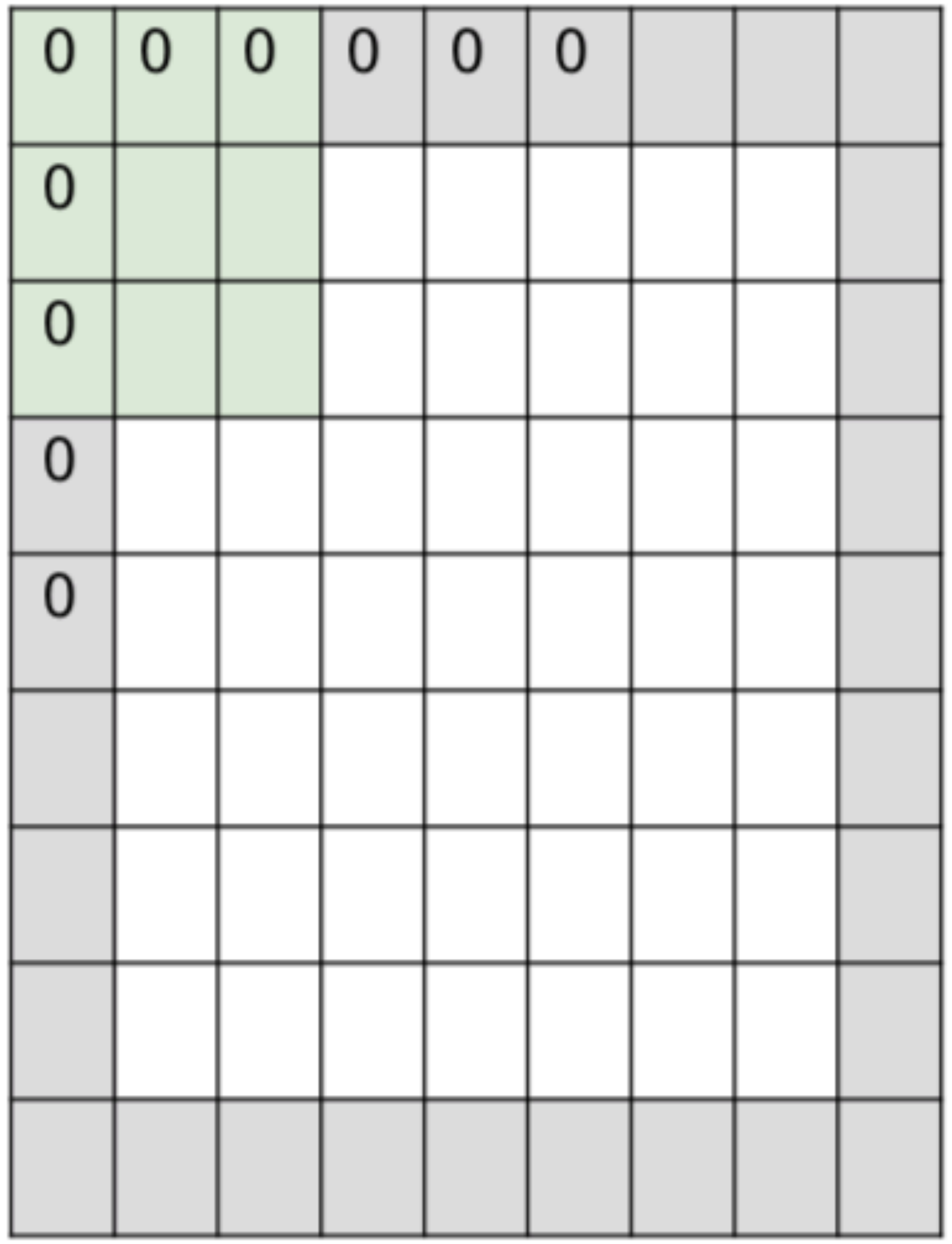


Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

Convolutional Neural Network



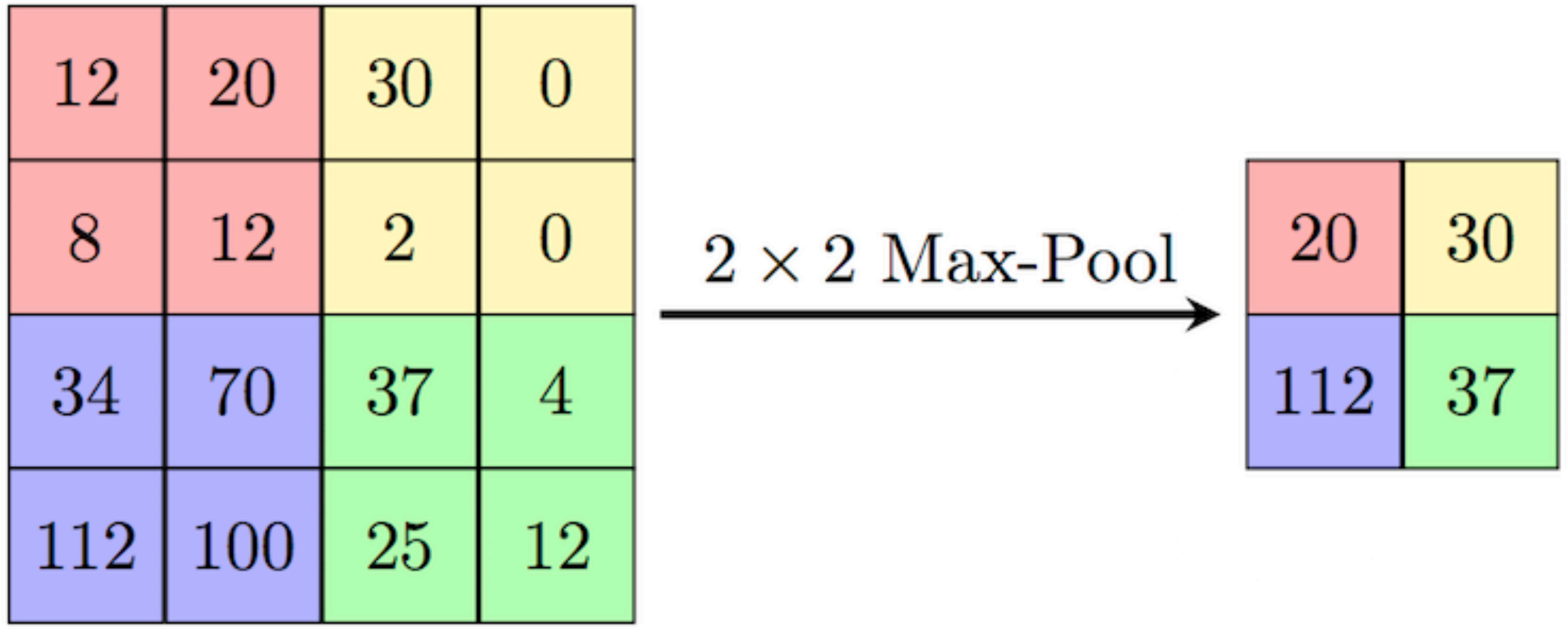
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

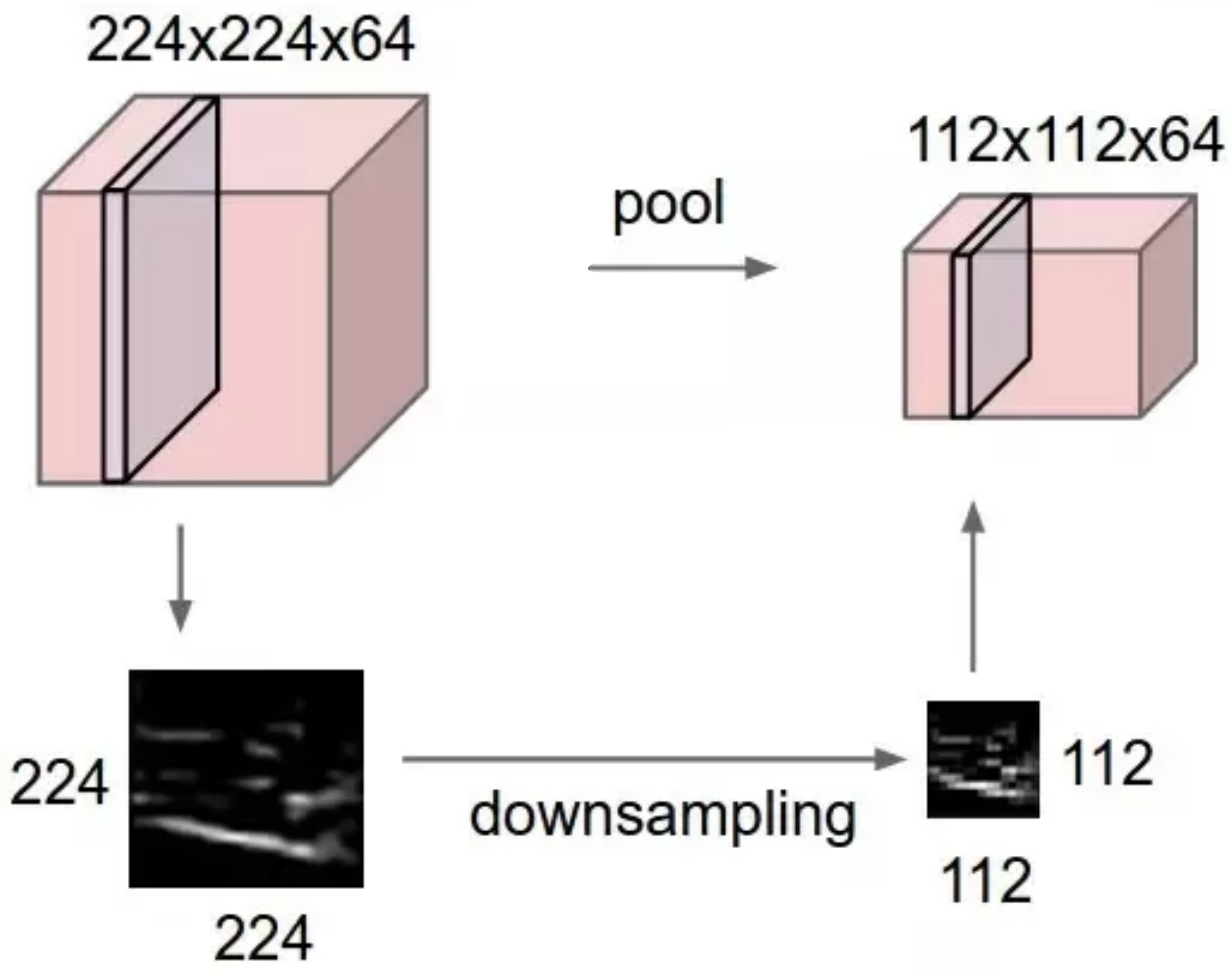
7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Slide from: Fei-Fei Li & Justin Johanson & Serena Yeung

Map Pooling

- Map Pooling layer reduces the dimension of input layer while the operation does not reduces information in the signal



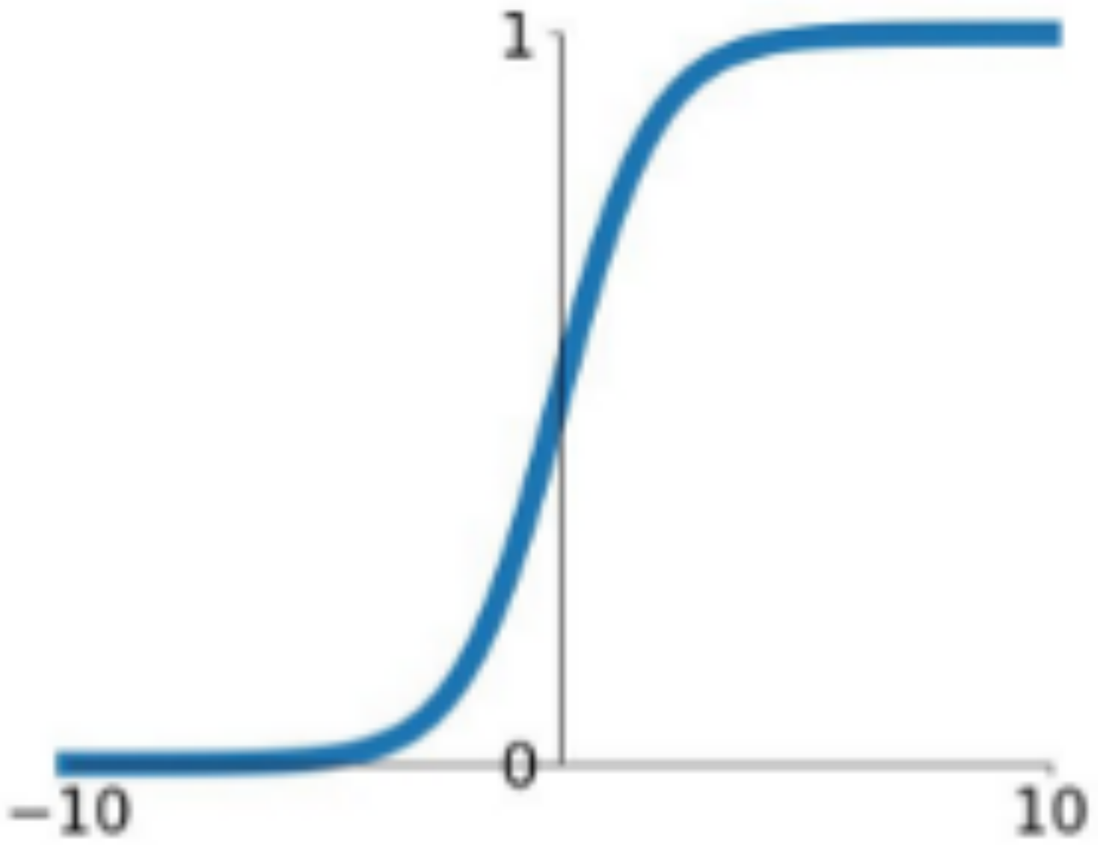


https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

Activation Functions

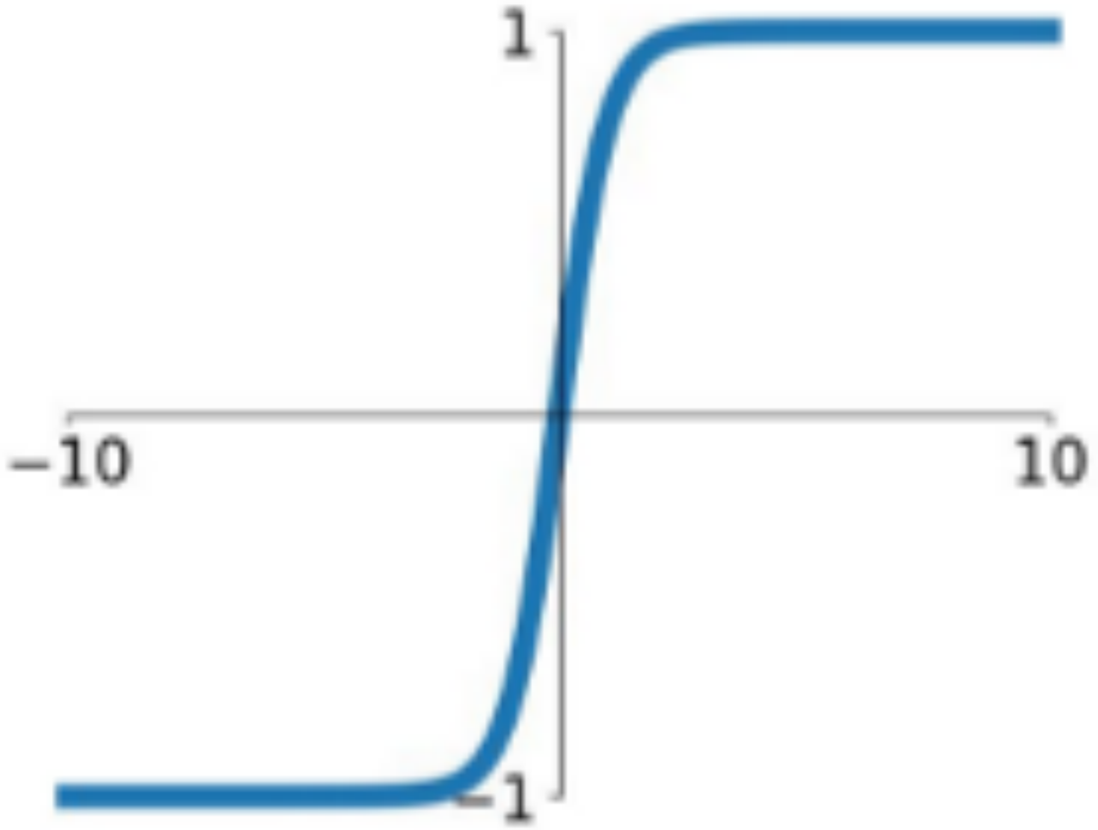
Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Tanh

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

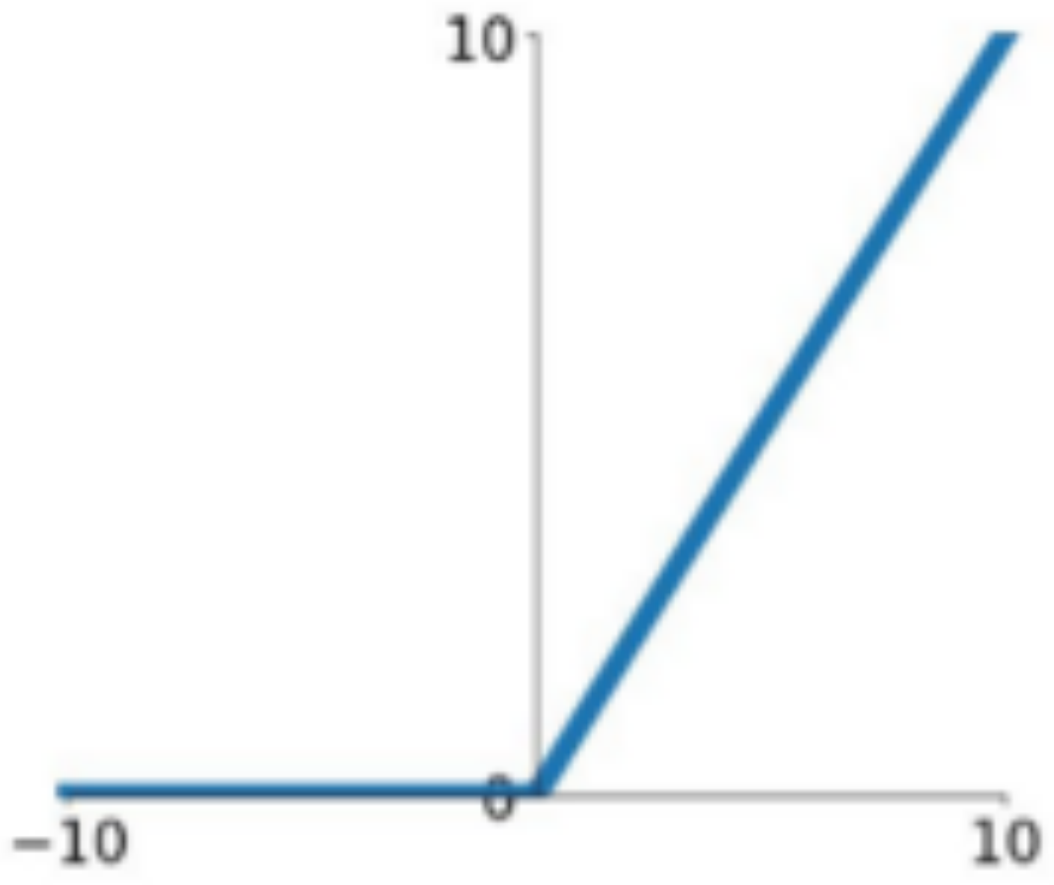


- Not zero centered
- Saturation kills gradient
- exp( ) is expensive to calculate

https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044

Activation Functions

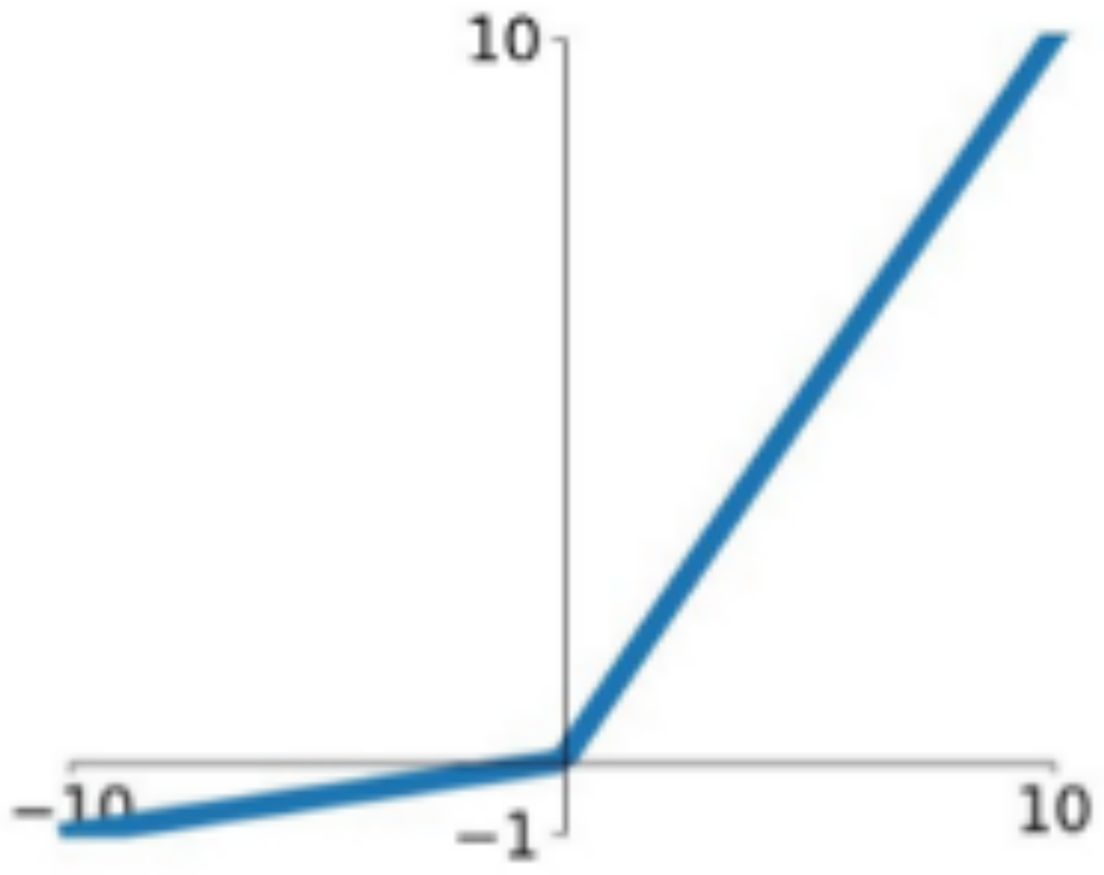
ReLU
(Rectified Linear Unit)



+ Simple to derivate

- Input value below 0 kills neurone

Leaky ReLU



+ Does not have the dead zone

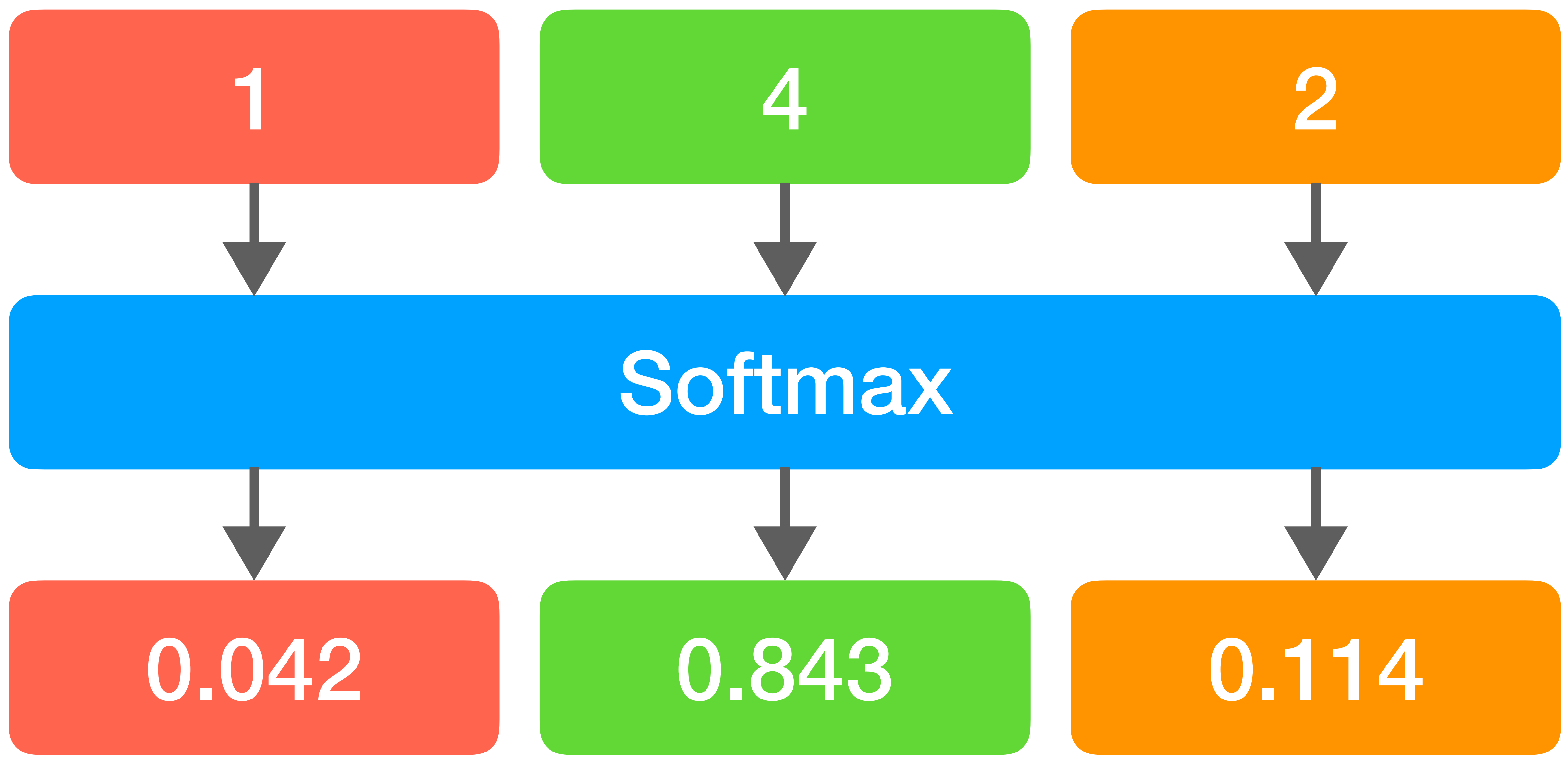
+ 10/10 machine learning engineers like Leaky Relu

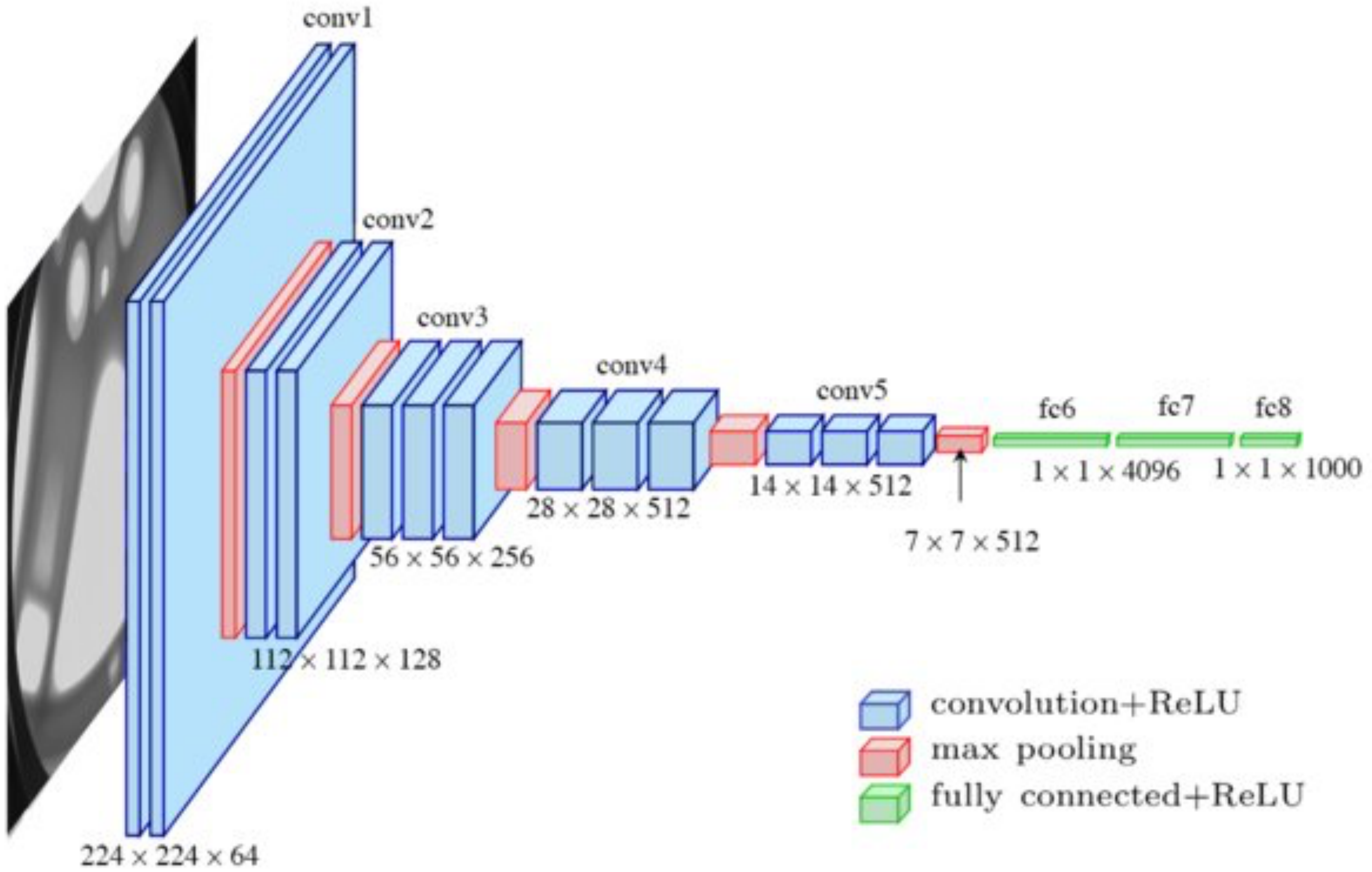
Activation Functions

- Softmax

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_k}} \text{ for } i = 1,\ldots,K$$

- Replaces non-linear max operation on the output of NN
- Softmax is differentiable -> can be used in back propag
- Normalize the output vector

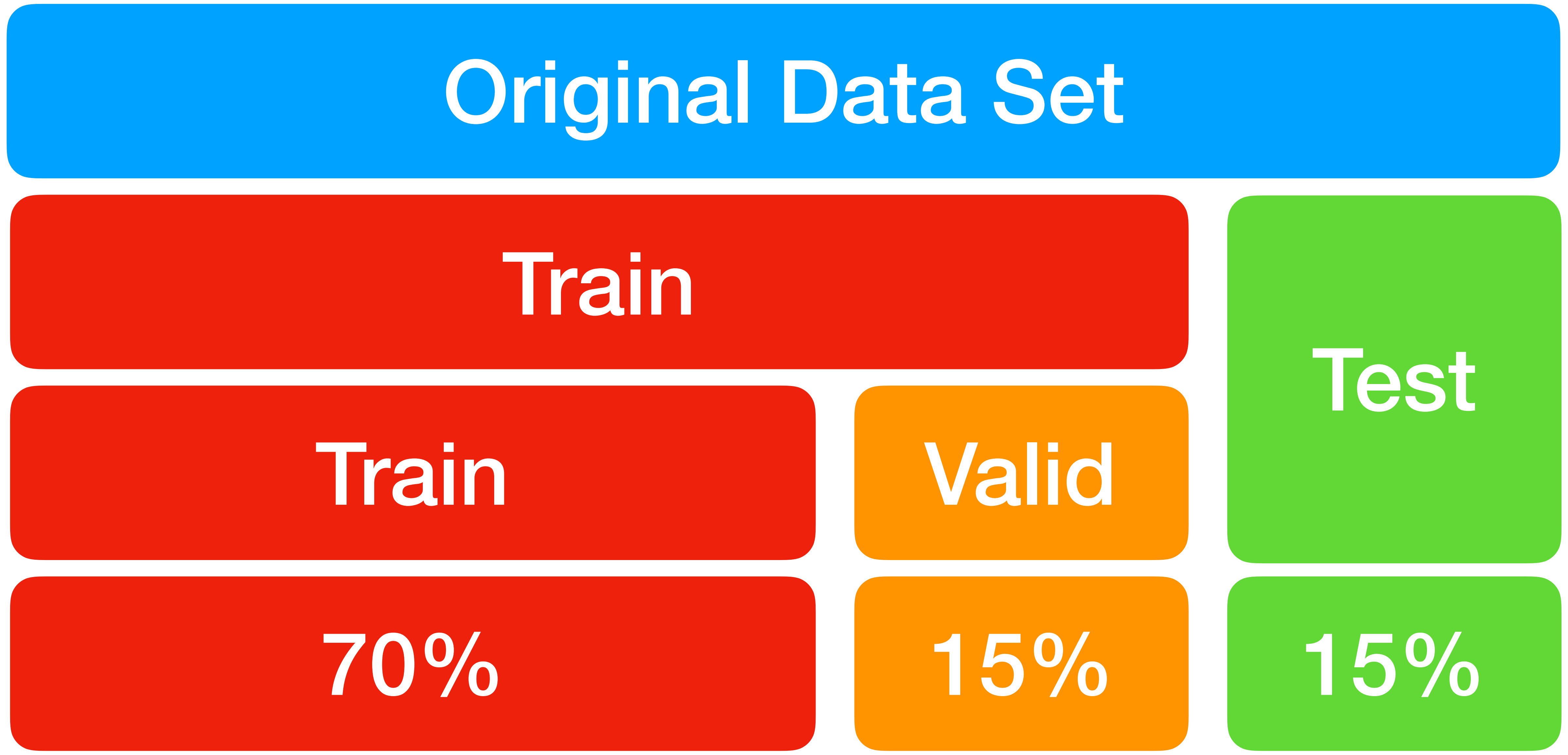



conv1
conv2
conv3
conv4
conv5
fc6
fc7
fc8
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU

Learning CNN

Training Dataset

- Splitting the dataset into the Train, Validation and the Test sets
- For small number of learning data see the “Cross Validation”
- Model is trained on the “Train data”
- Score on validation data helps to monitor learning progress
- Score on test data gives the final performance for the trained model



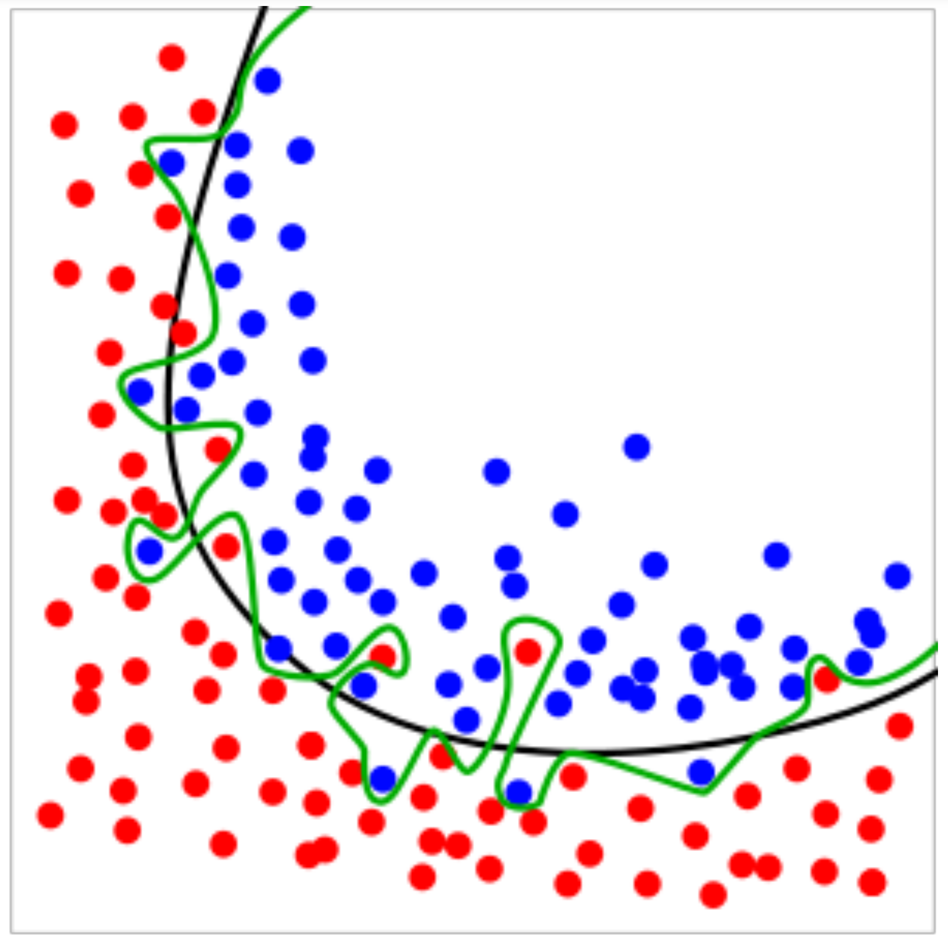
Overfitting

- When NN learns for long enough on “small” dataset, it starts to memorise features specific for training data

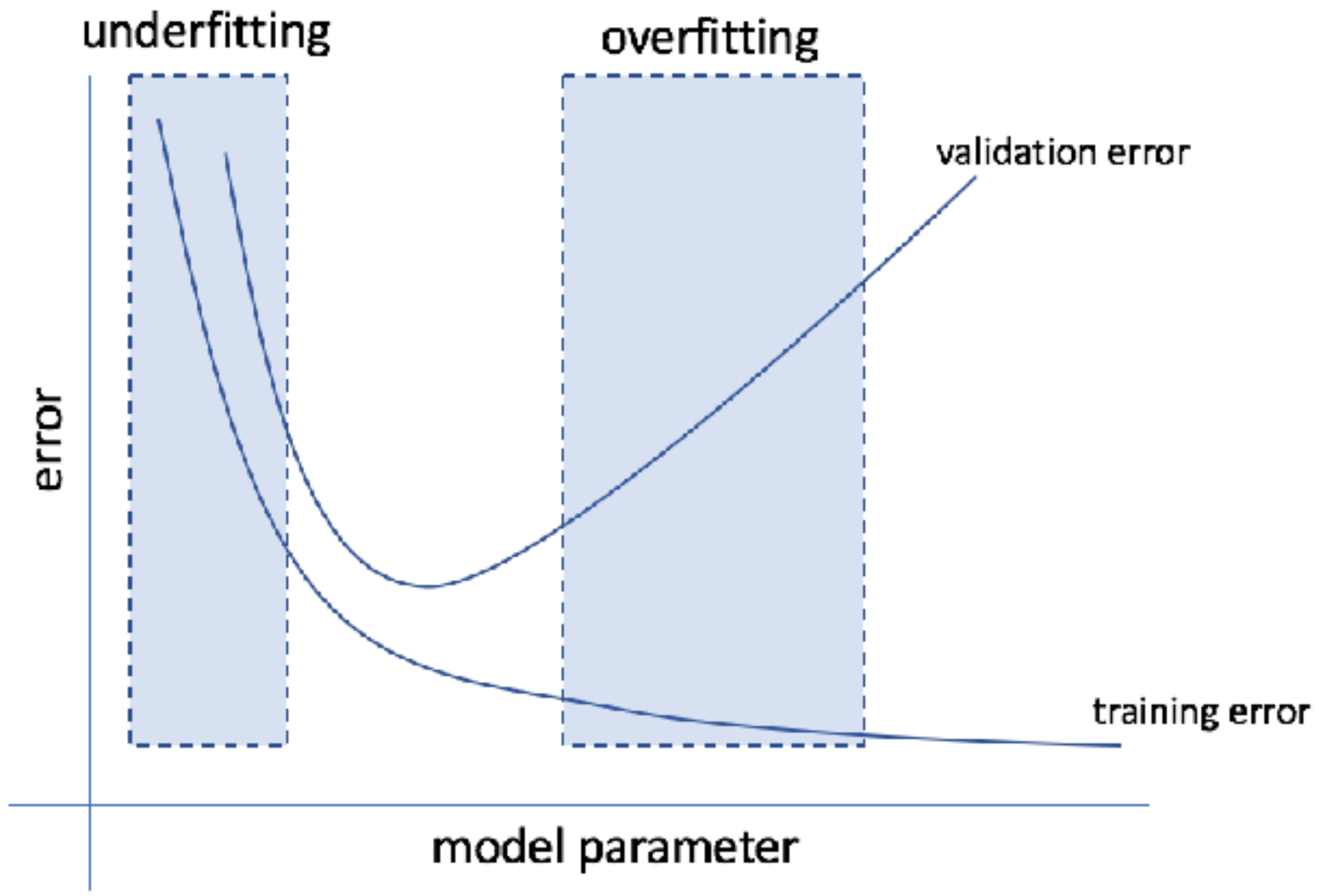
- Overfitting effect grows with the complexity of the model

- See: Occam's razor

- To prevent overfitting use the model with the lowest validation error, not the training one



https://en.wikipedia.org/wiki/Overfitting



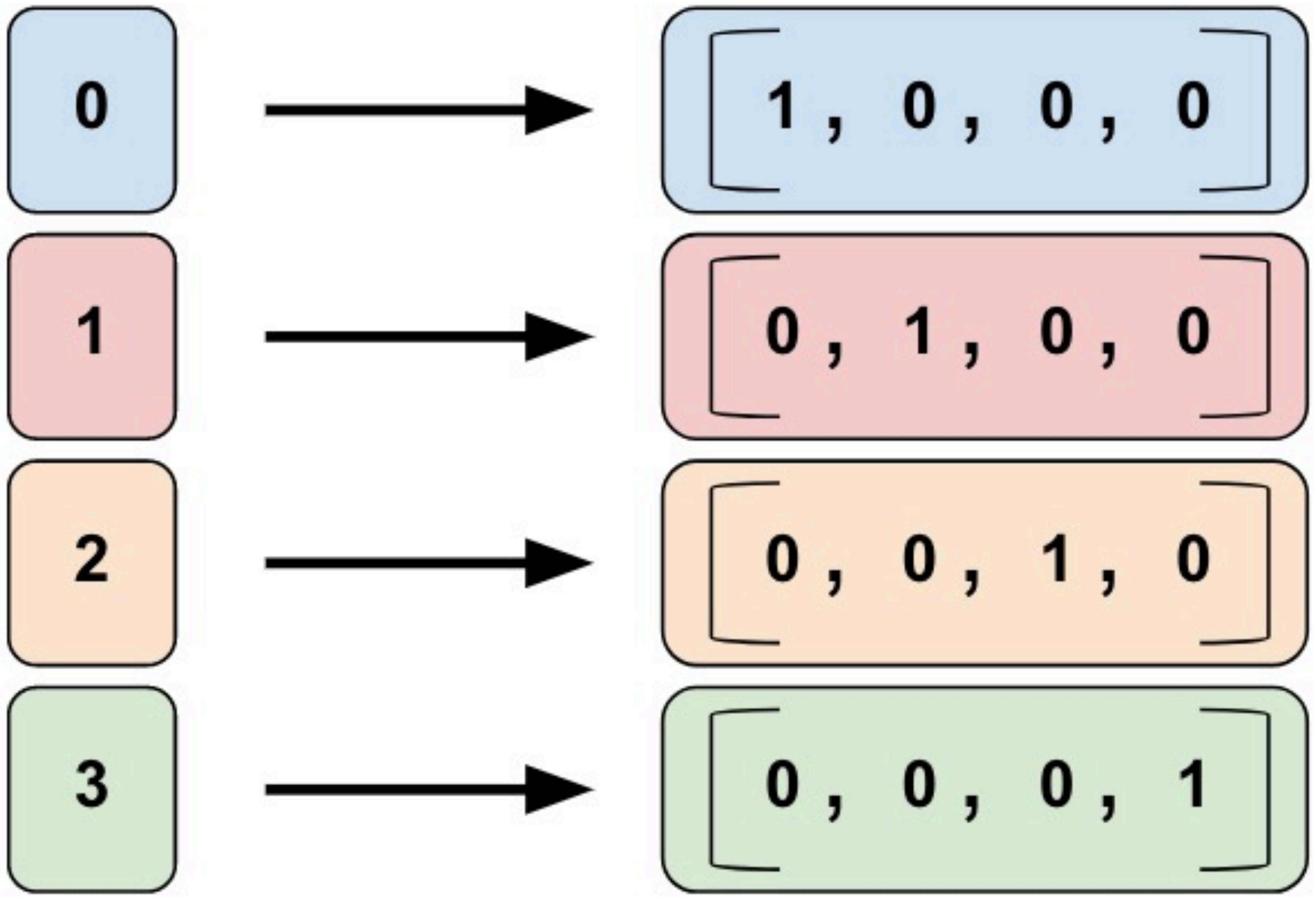
https://www.jeremyjordan.me/evaluating-a-machine-learning-model/

One HotEnd Encoding

- For every single class that NN should be able to classify there is one dedicate neuron in the output layer

- The output value ‘1’ of the neuron says “This is my class”, the output value ‘0’ says “Definitely not my class”.

- Representing many classes by single output value creates undesirable relation between neighbour values



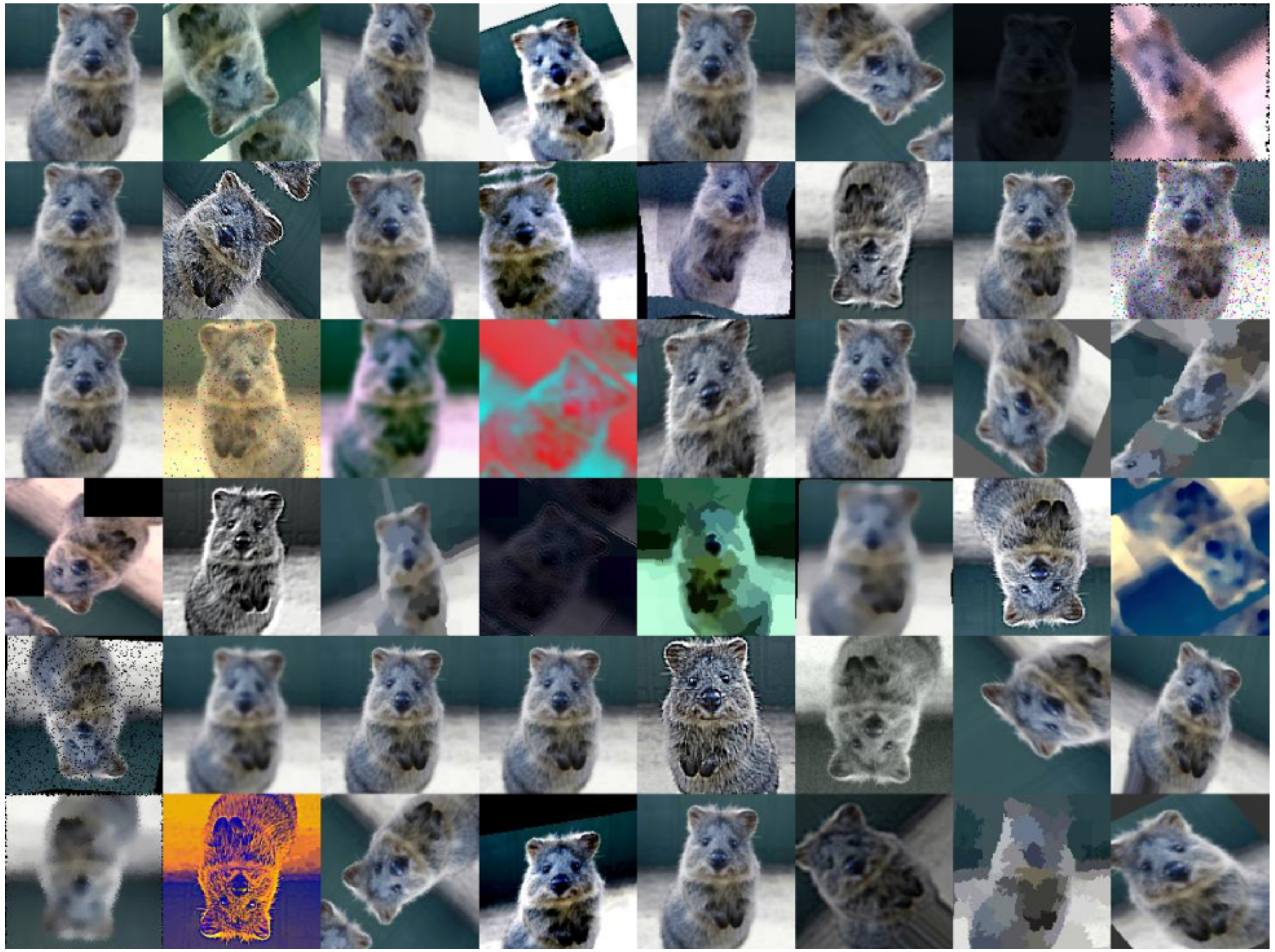
https://blog.e-kursy.it/deeplearning4j-workshop

Train Data Shuffling

- The classes in the input data should be distributed through the entire dataset
- Clustering classes next to each other causes temporary overfitting for specific class

Augmentations

- Used for small number of training data
- Artificially generate synthetical data for training
- Artificial data helps to improve the training process and the model robustness
- Augmentation techniques
  - Flip
  - Rotation
  - Scaling
  - Crop
  - Translation
  - Noising



https://github.com/aleju/imgaug

Batch Learning

- On-line learning technique updates model weights by using back propagation algorithm after every single error estimation for input data sample
- Batch learning cumulates and averages $\Delta\mathbf{w}$ over several input data samples
- Batch learning prevents to overfit the model and learning process is more stable compared to on-line learning

Regularisation

- Extending Cost function by the term that summarise weights over entire neural network model

- Most used are the L1 and L2 regularisation terms

Cost Function = Loss + Regularisation term

$$L_1 = \alpha * \sum ||\mathbf{w}|| \qquad L_2 = \alpha * \sum ||\mathbf{w}||^2$$

- Regularisation penalised large weights coefficients which causes overfitting

- Usually we choose regularisation parameter about ~0.001

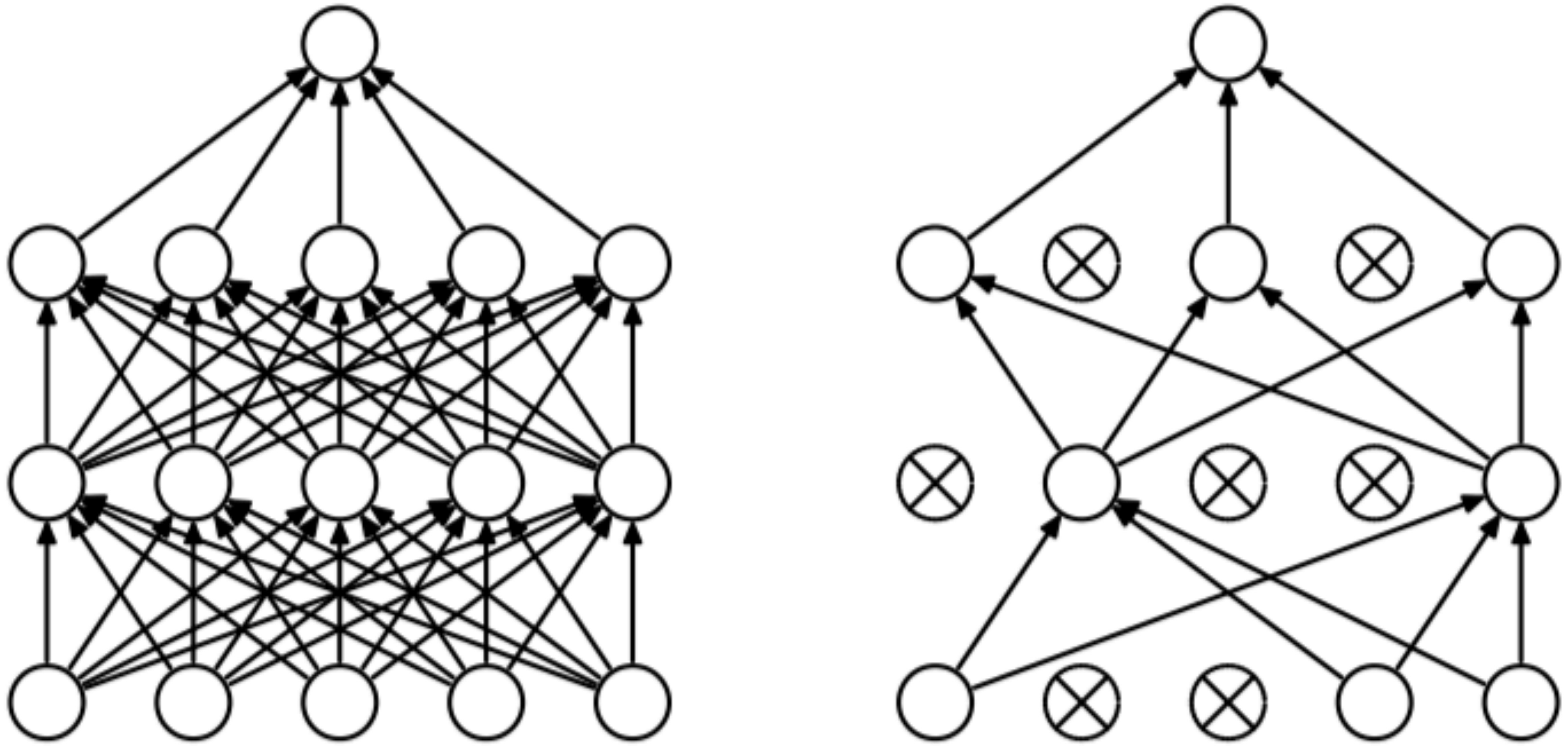
Dropout

- Idea of dropout is about to randomly “switch off” some neurones

- Dropout ration usually up to 50% of all neurones in the NN

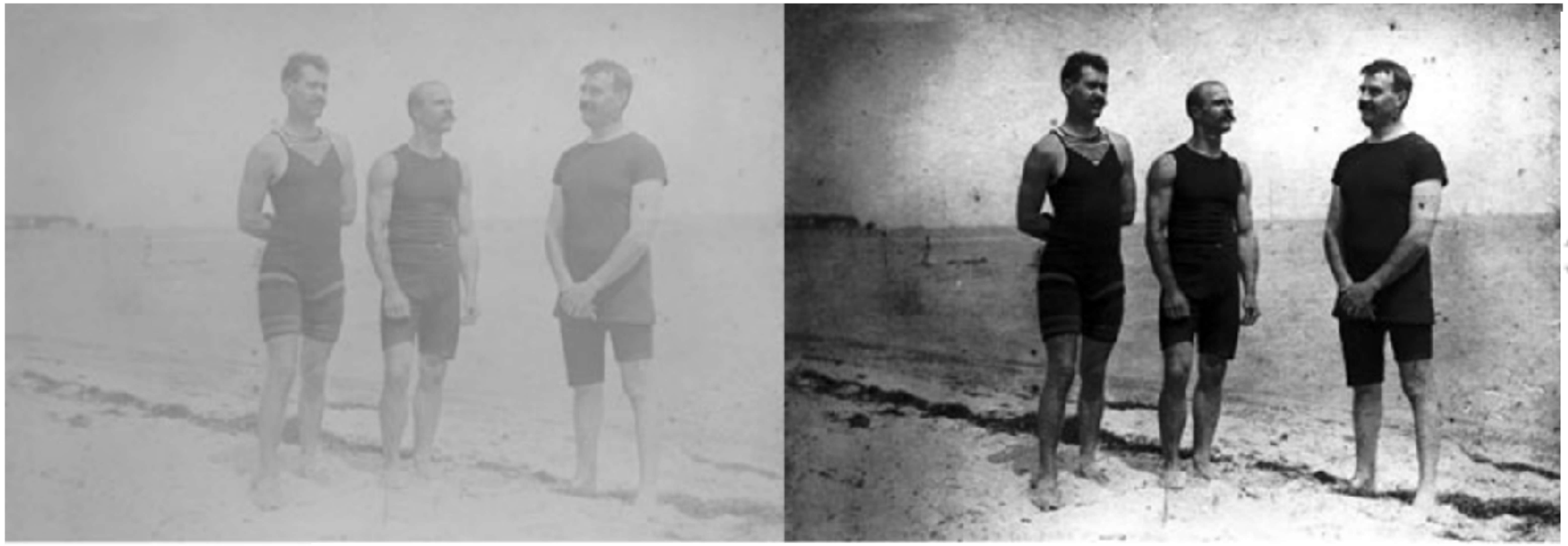
- Technique prevents model to overfit training dataset by using low number of neurones

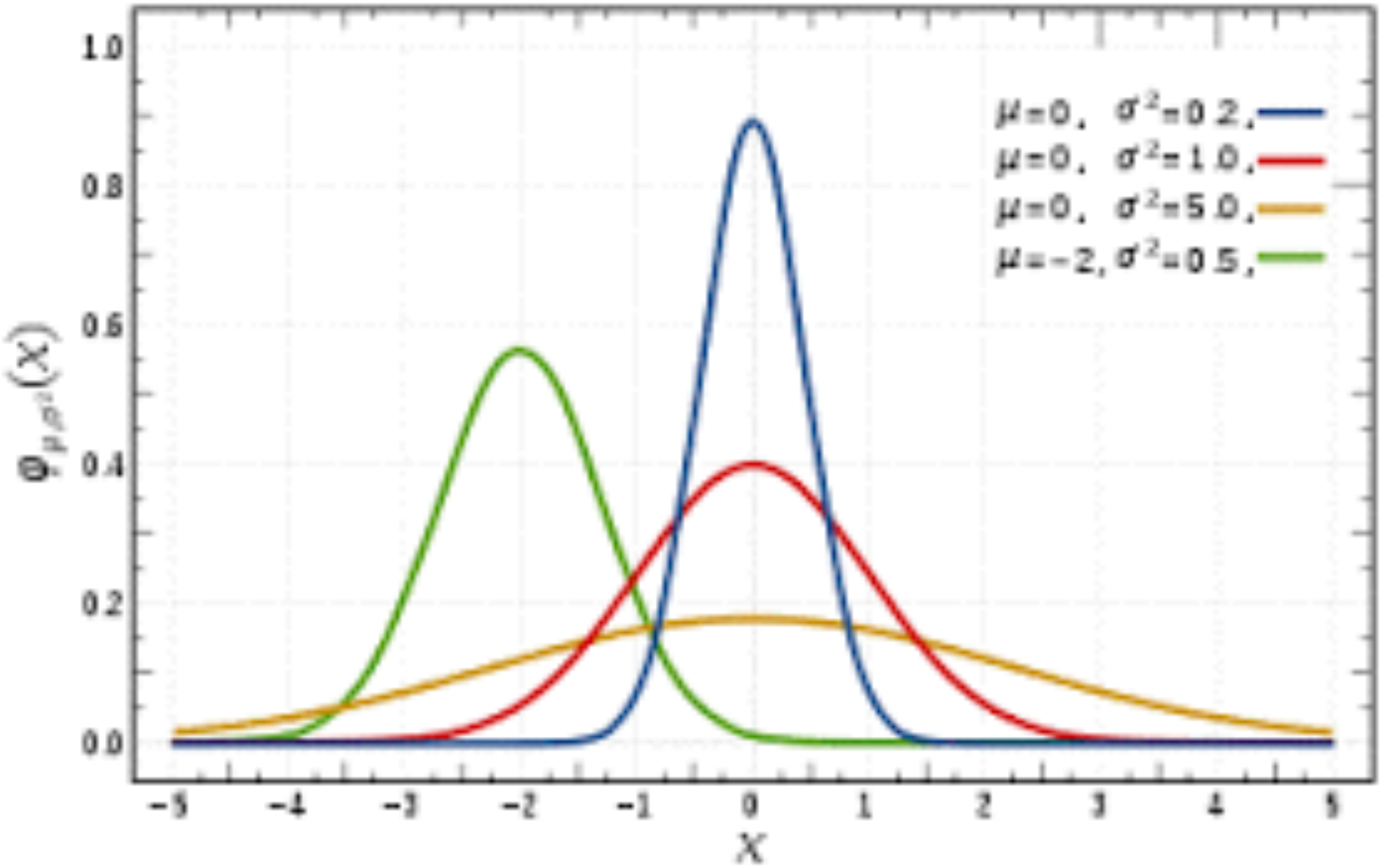
- Dropout is used only during the learning phase. For testing phase the dropout is turned off



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

Batch Normalisation

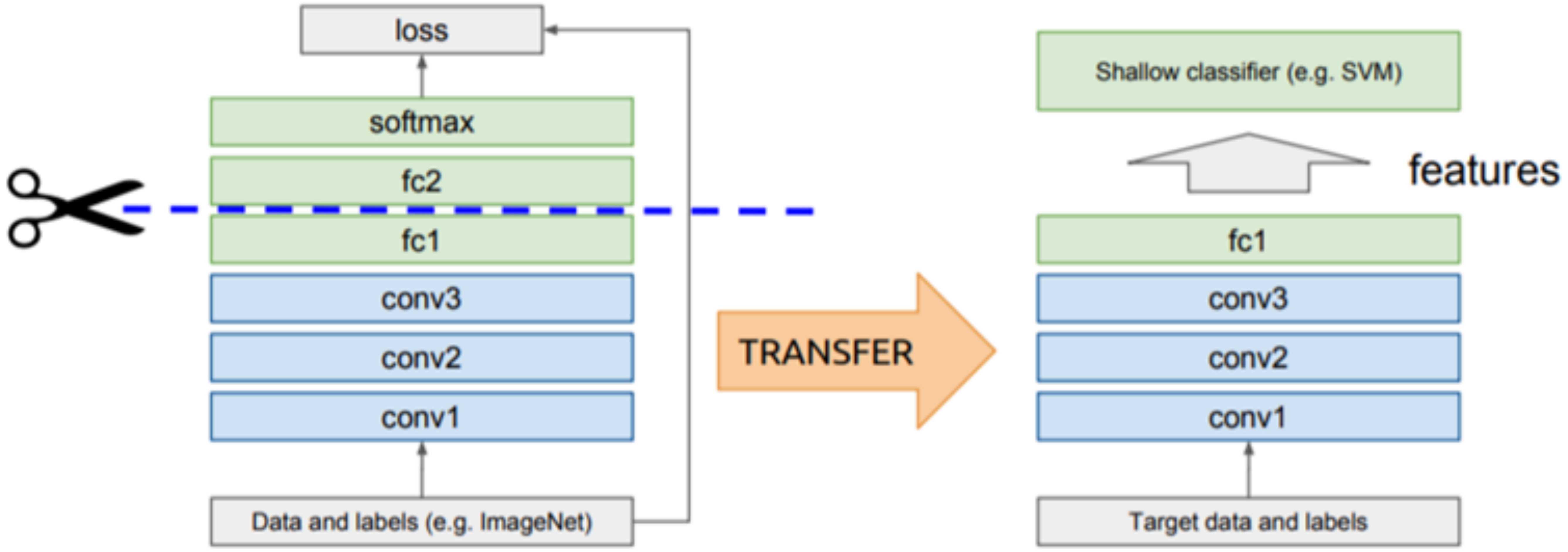

https://www.bruzed.com/2009/10/contrast-stretching-and-histogram-equalization/



- Various input data have very different value distribution which leads to the effect, that some data have larger impact on NN decision making
- Normalising all the input channels for the same mean and std dev makes all input channels “same important”

Transfer Learning

- Training the feature extractors in the middle of the DNN is the hardest job
- TF works with the idea that models which are designed for the similar set of problems are more or less the same
- Why to learn NN from scratch? Use something that already exists!



BottleNeck

- Is reducing computational complexity, while it keeps similar quality of the feature extractions capability

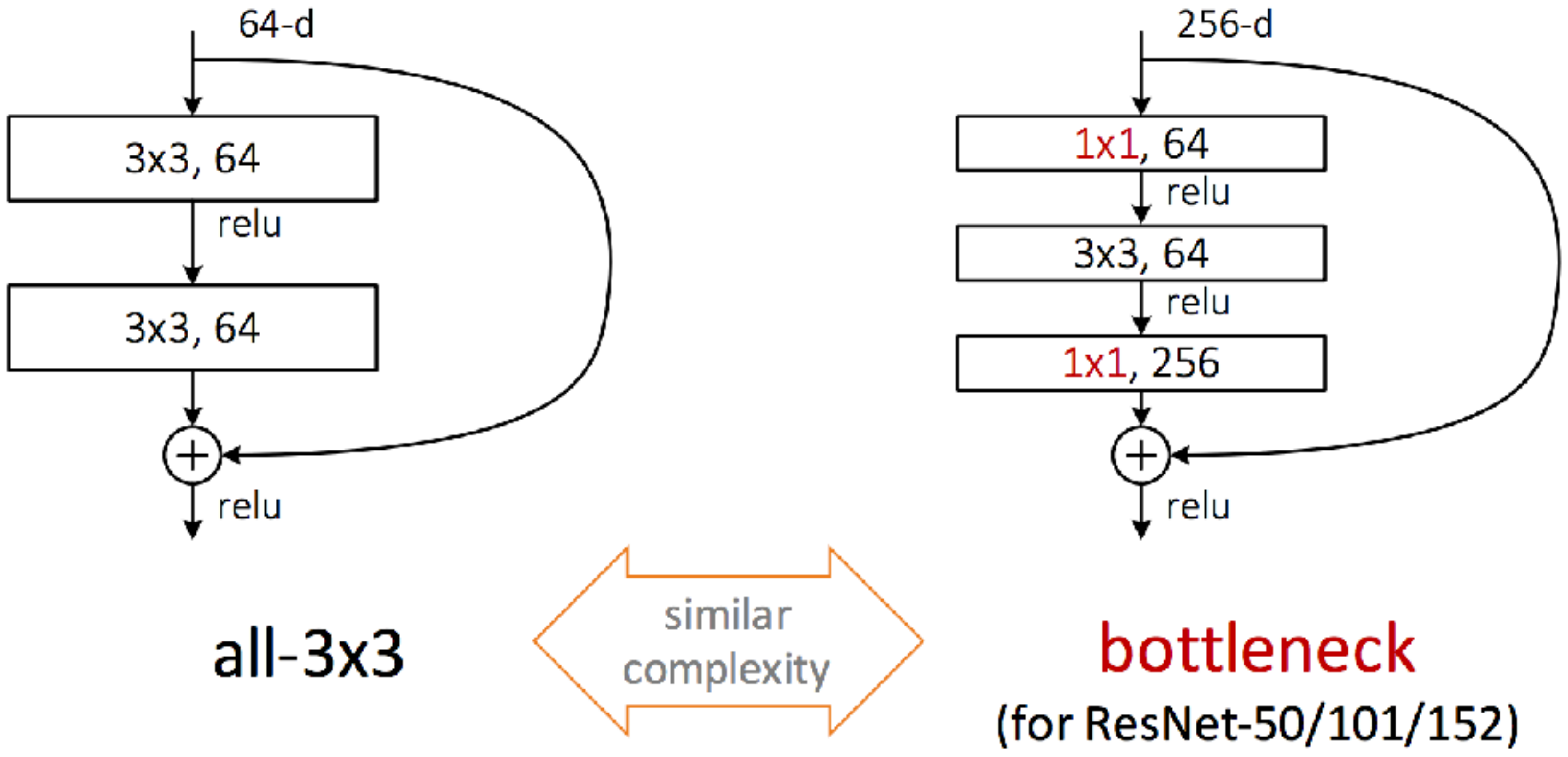
- Layer 256 channels in,
  256 channels out,

  — — — — — — — — — — — — — — —

  conv 3x3x256:
  256x256x3x3 ops ~= 0.6mil ops

  — — — — — — — — — — — — — — —

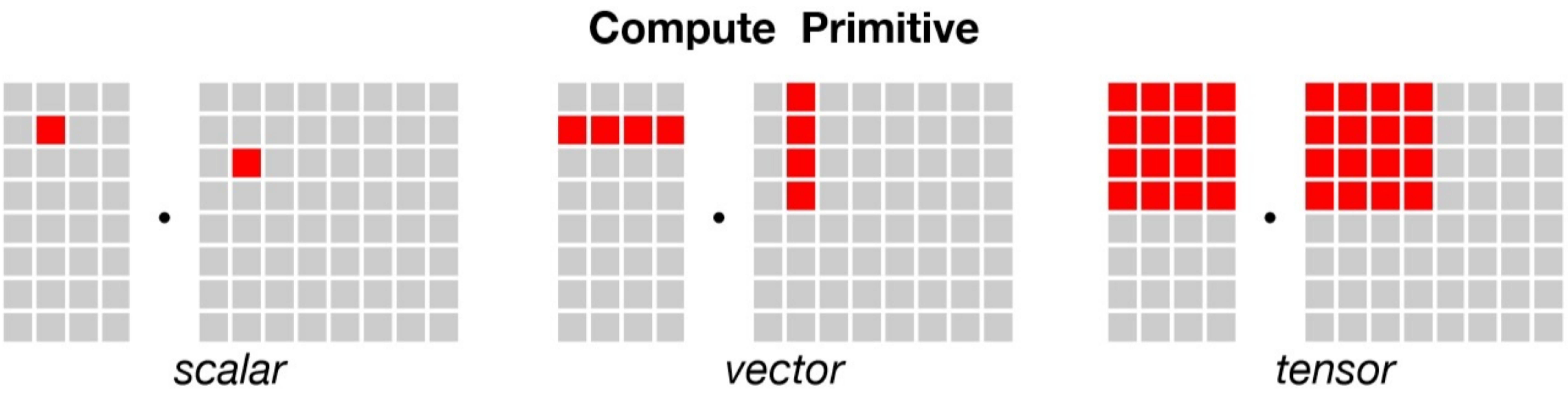
  conv 1x1x64: 256x64x1x1 = 16k
  conv 3x3x64: 64x64x3x3 = 36k
  conv 1x1x256: 64x256x1x1 = 16k



https://stephan-osterburg.gitbook.io

CPU vs GPU vs TPU

- Central PU - general design to solve every mathematical problem
- Graphics PU - specialised design for parallelisation simple rendering tasks
- Tensor PU - matrix multiplication only dedicated HW with fast memory management

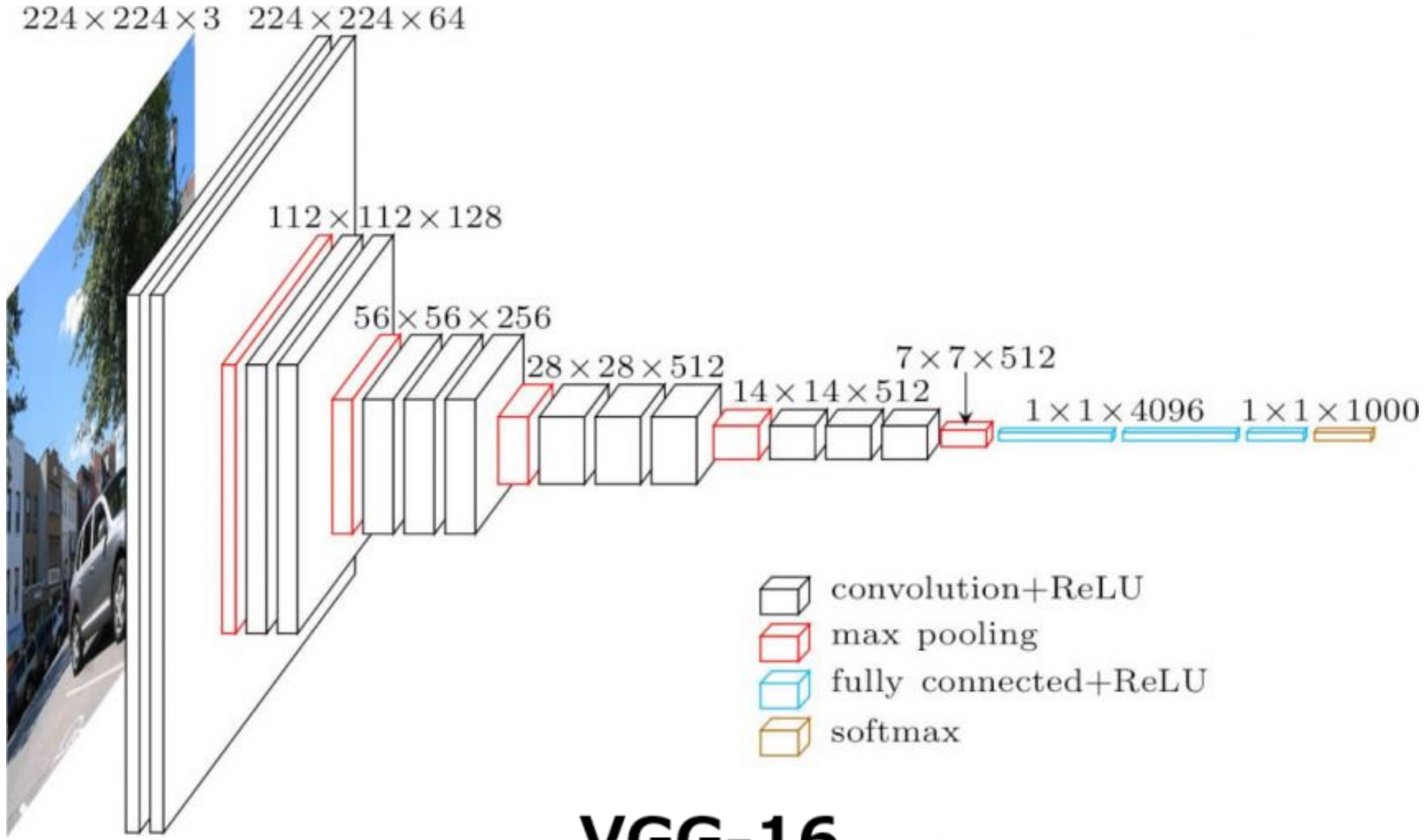


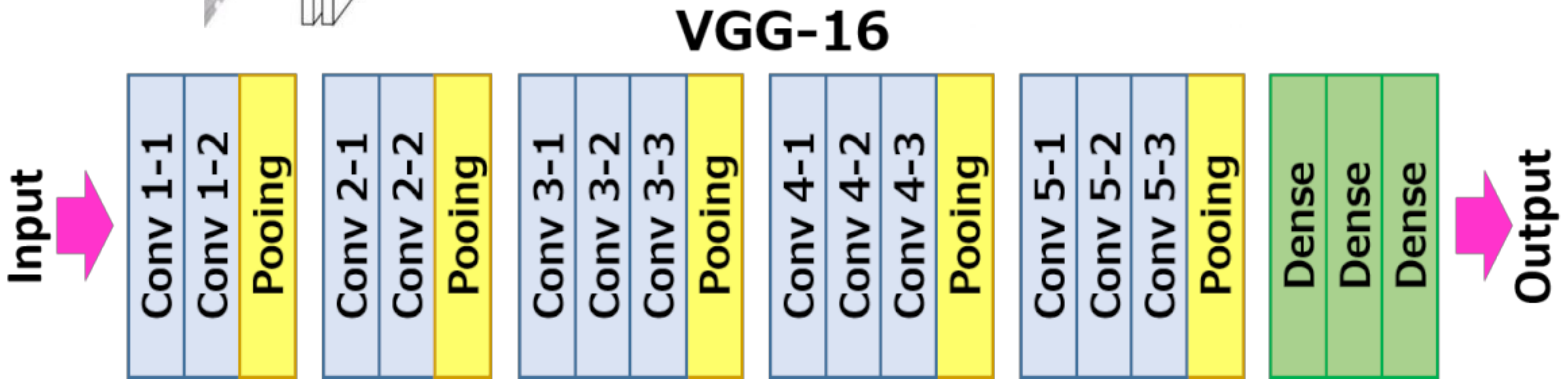
https://iq.opengenus.org/cpu-vs-gpu-vs-tpu/

Other Architectures

VGG16

- First widely used CNN
- Compared to AlexNex, VGG uses only 3x3 kernels (11x11 and 5x5 in AlexNet)
- 92.7% top 5 accuracy in ImageNet Challenge





ResNet (Residual NN)

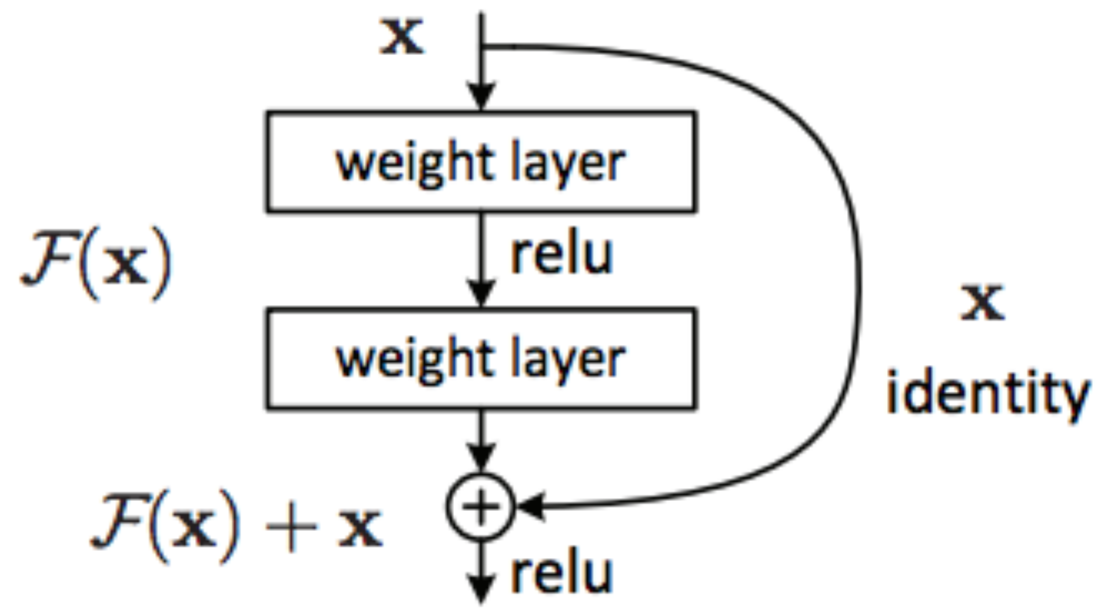
- Introducing bypass over neural network's layers

- Bypass helps to model identity over the layer

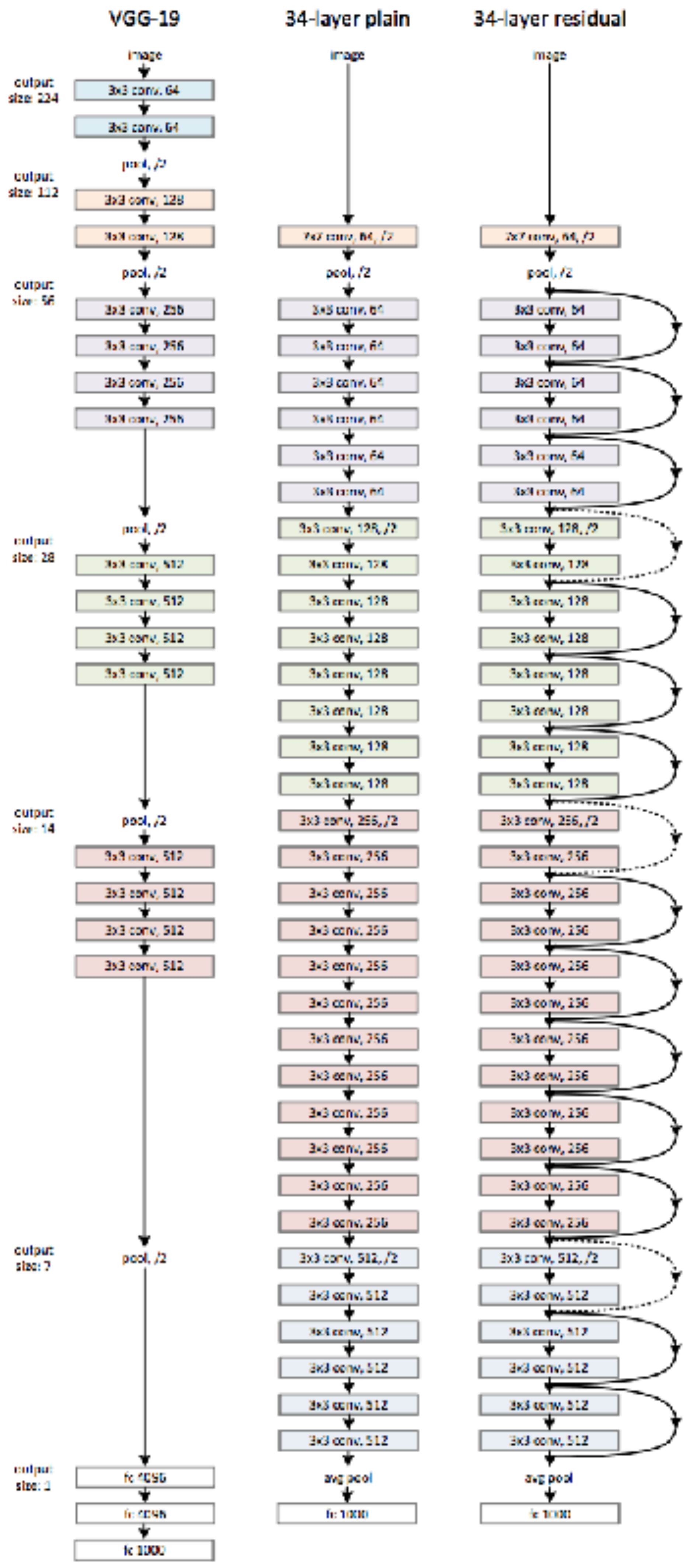
- Identity allows to bypass the "vanishing gradient" problem for very deep CNN

- Best known: ResNet52, ResNet152

- First ever trained DCNN with more than 1000 layers



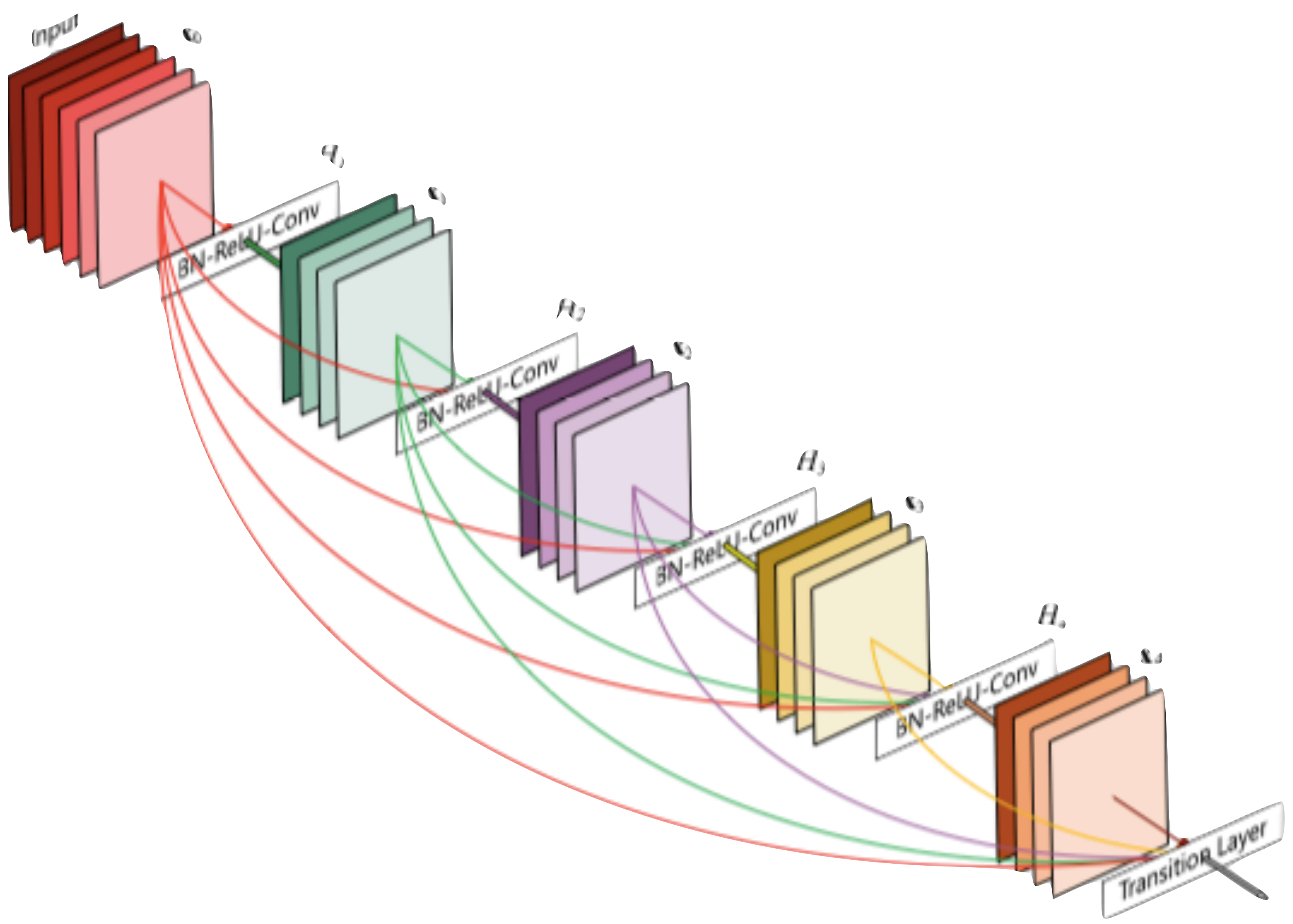
$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$



DenseNet, PyramidNet

- Introducing bypasses between layer groups through entire architecture

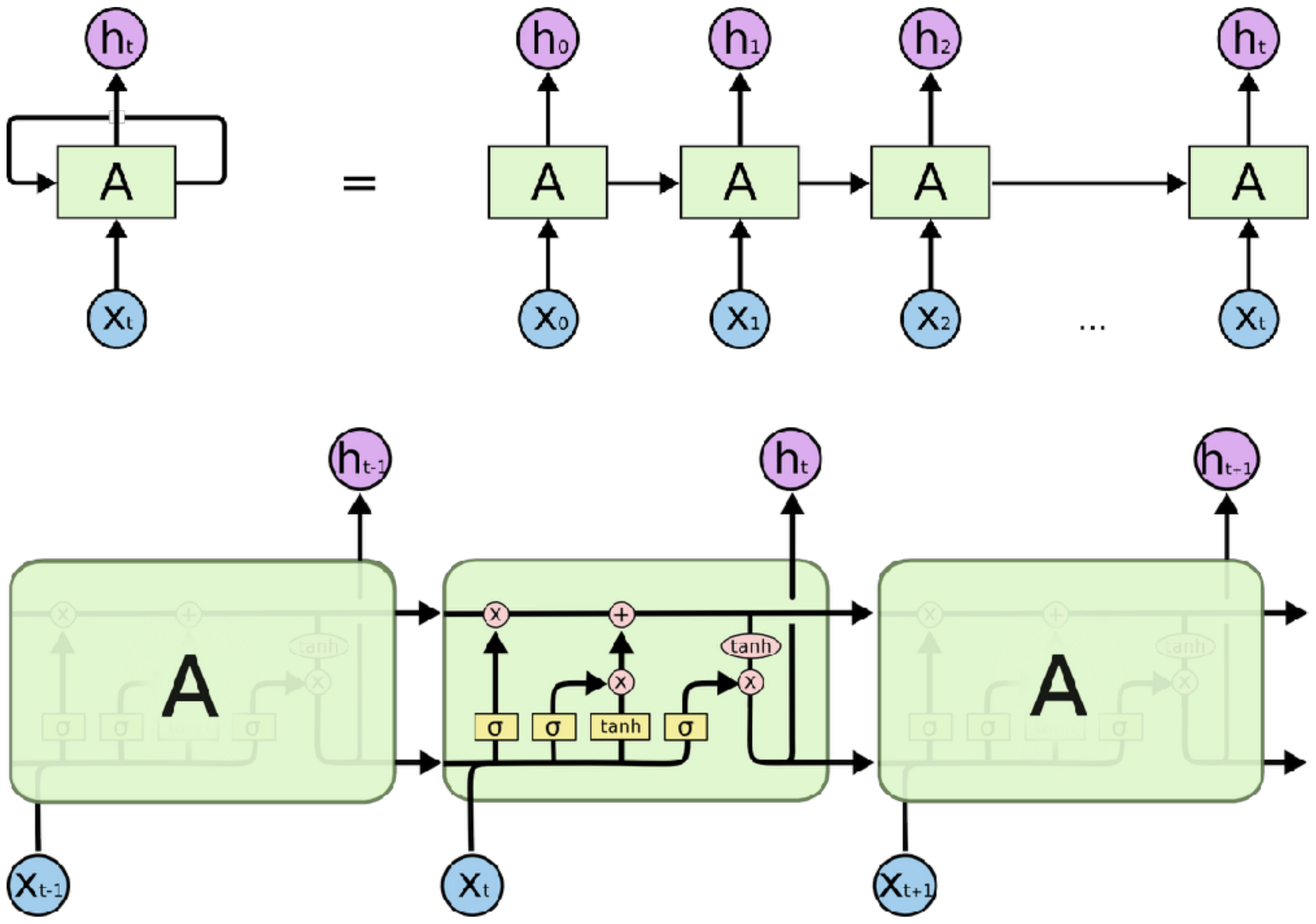
- Currently best results on image recognition challenges



Network	# of Params	Output Feat. Dim.	Depth	Training Mem.	CIFAR-10	CIFAR-100
NiN [18]	-	-	-	-	8.81	35.68
All-CNN [27]	-	-	-	-	7.25	33.71
DSN [17]	-	-	-	-	7.97	34.57
FitNet [21]	-	-	-	-	8.39	35.04
Highway [29]	-	-	-	-	7.72	32.39
Fractional Max-pooling [4]	-	-	-	-	4.50	27.62
ELU [29]	-	-	-	-	6.55	24.28
ResNet [7]	1.7M	64	110	547MB	6.43	25.16
ResNet [7]	10.2M	64	1001	2,921MB	-	27.82
ResNet [7]	19.4M	64	1202	2,069MB	7.93	-
Pre-activation ResNet [8]	1.7M	64	164	841MB	5.46	24.33
Pre-activation ResNet [8]	10.2M	64	1001	2,921MB	4.62	22.71
Stochastic Depth [10]	1.7M	64	110	547MB	5.23	24.58
Stochastic Depth [10]	10.2M	64	1202	2,069MB	4.91	-
FractalNet [14]	38.6M	1,024	21	-	4.60	23.73
SwapOut v2 (width$\times$4) [26]	7.4M	256	32	-	4.76	22.72
Wide ResNet (width$\times$4) [34]	8.7M	256	40	775MB	4.97	22.89
Wide ResNet (width$\times$10) [34]	36.5M	640	28	1,383MB	4.17	20.50
Weighted ResNet [24]	19.1M	64	1192	-	5.10	-
DenseNet ($k = 24$) [9]	27.2M	2,352	100	4,381MB	3.74	19.25
DenseNet-BC ($k = 40$) [9]	25.6M	2,190	190	7,247MB	3.46	17.18
PyramidNet ($\alpha = 48$)	1.7M	64	110	655MB	4.58$\pm$0.06	23.12$\pm$0.04
PyramidNet ($\alpha = 84$)	3.8M	100	110	781MB	4.26$\pm$0.23	20.66$\pm$0.40
PyramidNet ($\alpha = 270$)	28.3M	286	110	1,437MB	3.73$\pm$0.04	18.25$\pm$0.10
PyramidNet (bottleneck, $\alpha = 270$)	27.0M	1,144	164	4,169MB	3.48$\pm$0.20	17.01$\pm$0.39
PyramidNet (bottleneck, $\alpha = 240$)	26.6M	1,024	200	4,451MB	3.44$\pm$0.11	16.51$\pm$0.13
PyramidNet (bottleneck, $\alpha = 220$)	26.8M	944	236	4,767MB	3.40$\pm$0.07	16.37$\pm$0.29
PyramidNet (bottleneck, $\alpha = 200$)	26.0M	864	272	5,005MB	**3.31**$\pm$0.08	**16.35**$\pm$0.24

Recurrent Neural Nets

- Hopfield Neural Net
- Long Short Term Memory Neural Network
- Usage:
  signal and text processing and generation
- RNN are extremely demanding on memory amount to remember partial derivations of entire input series

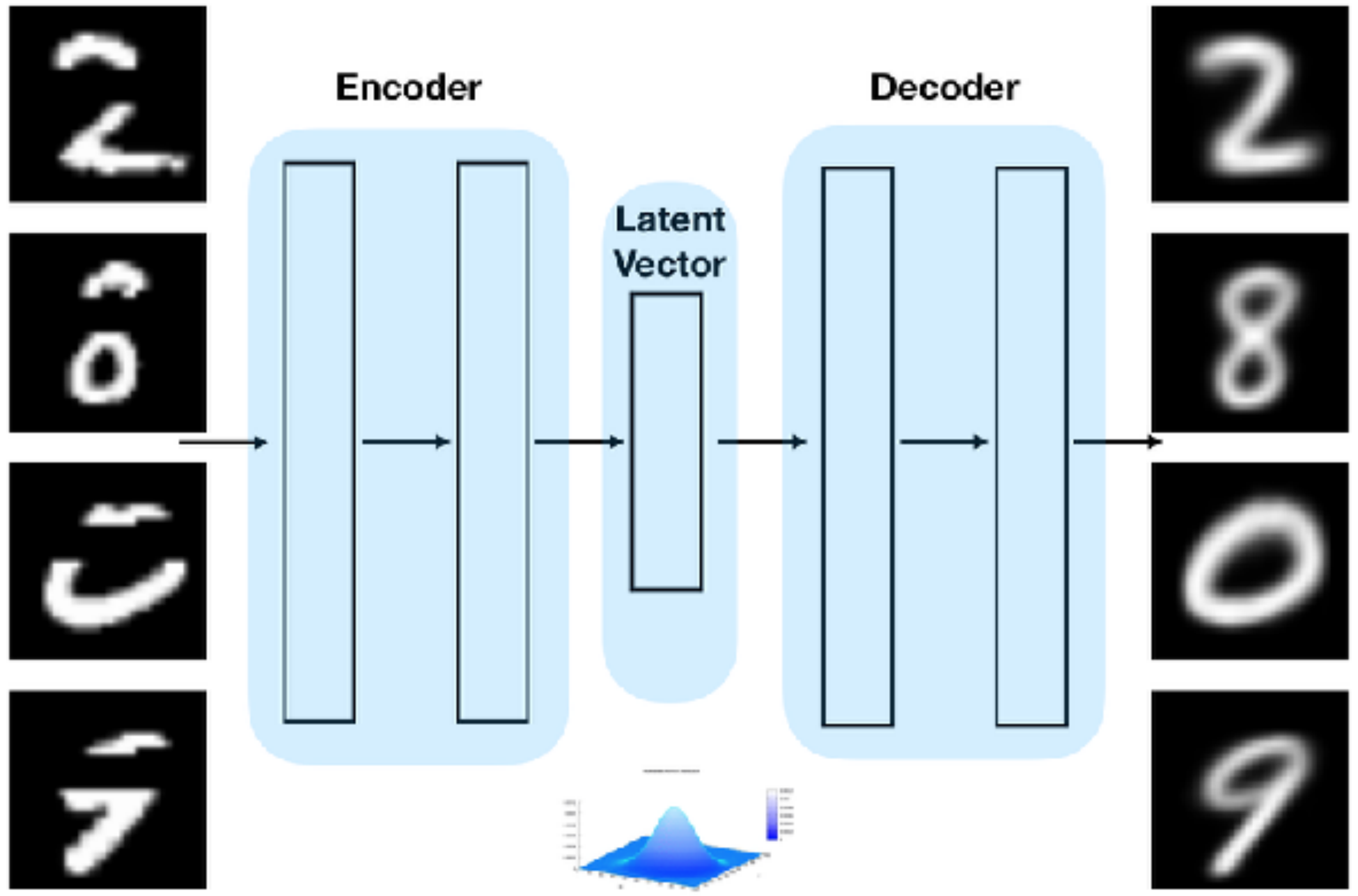


Encoders

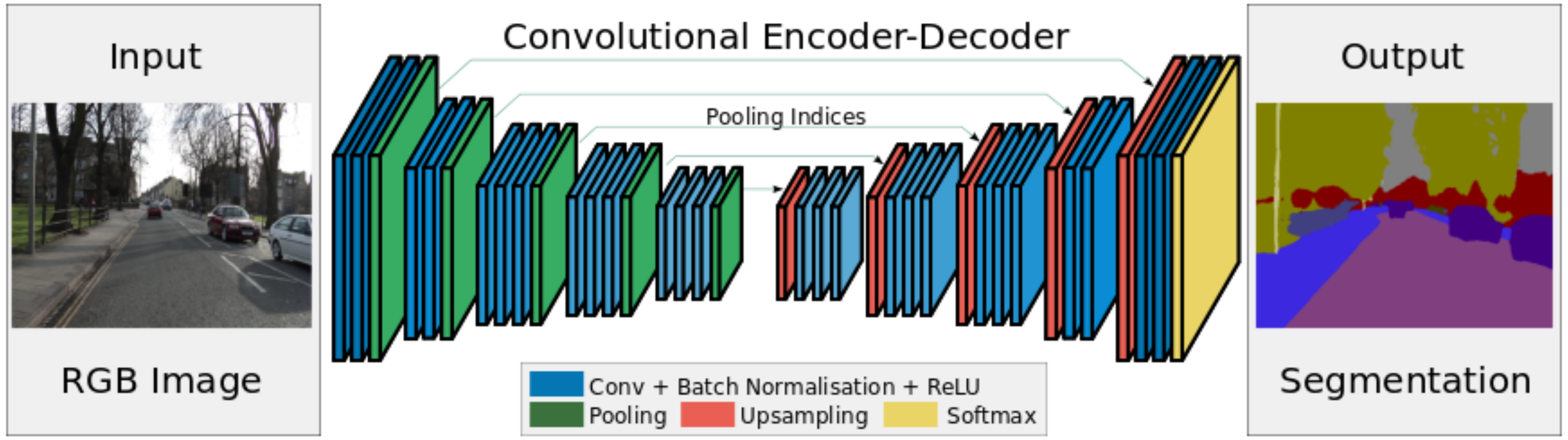
- Architecture reduces input image into low dimensional vector that extracts (stores) information

- Upsampling section expand stored information into output image

- Applications:
  - Denoising
  - Data Compression
  - Semantic Segmentation

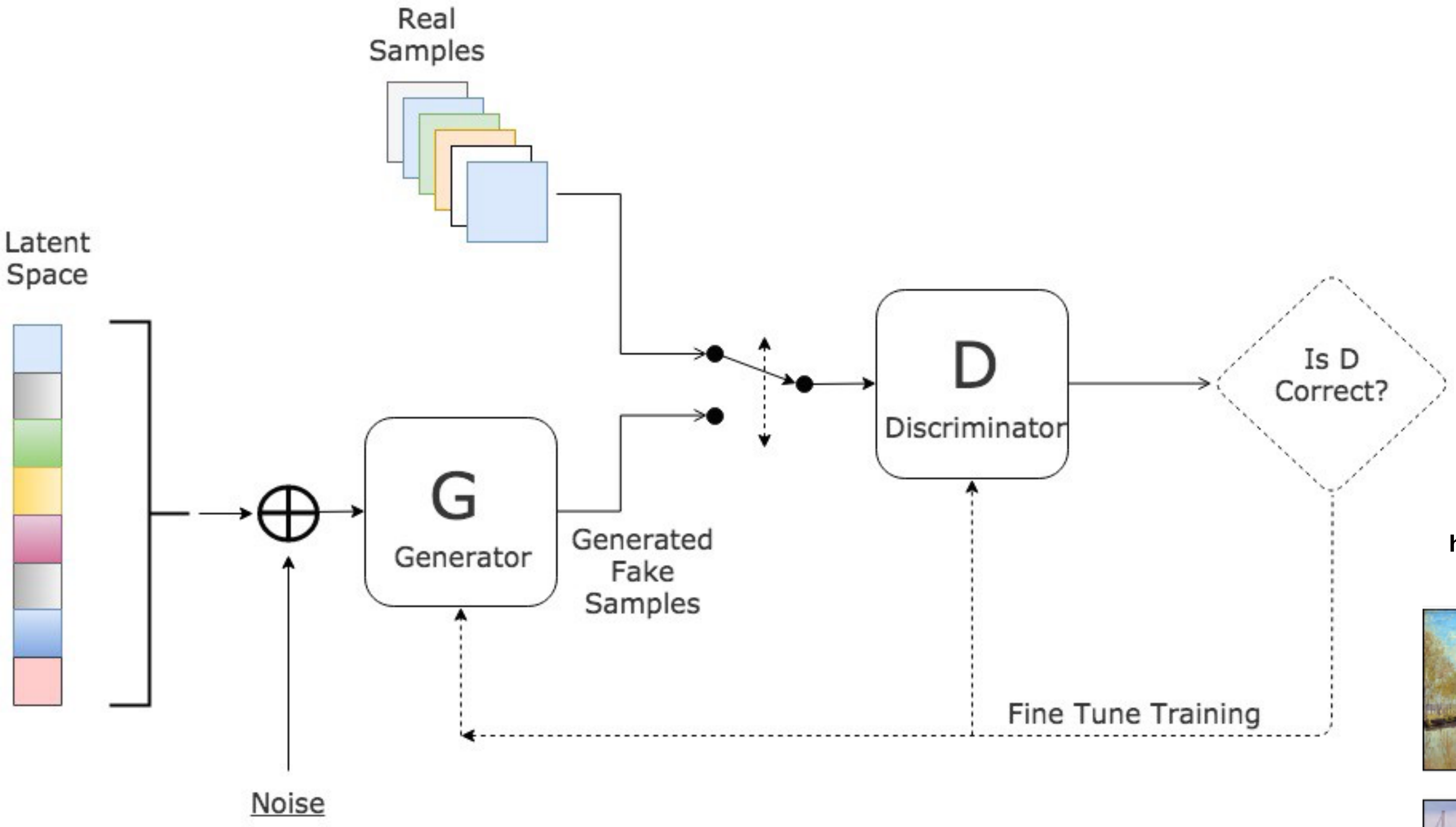


https://towardsdatascience.com/generating-digits-and-sounds-with-artificial-neural-nets-ca1270d8445f



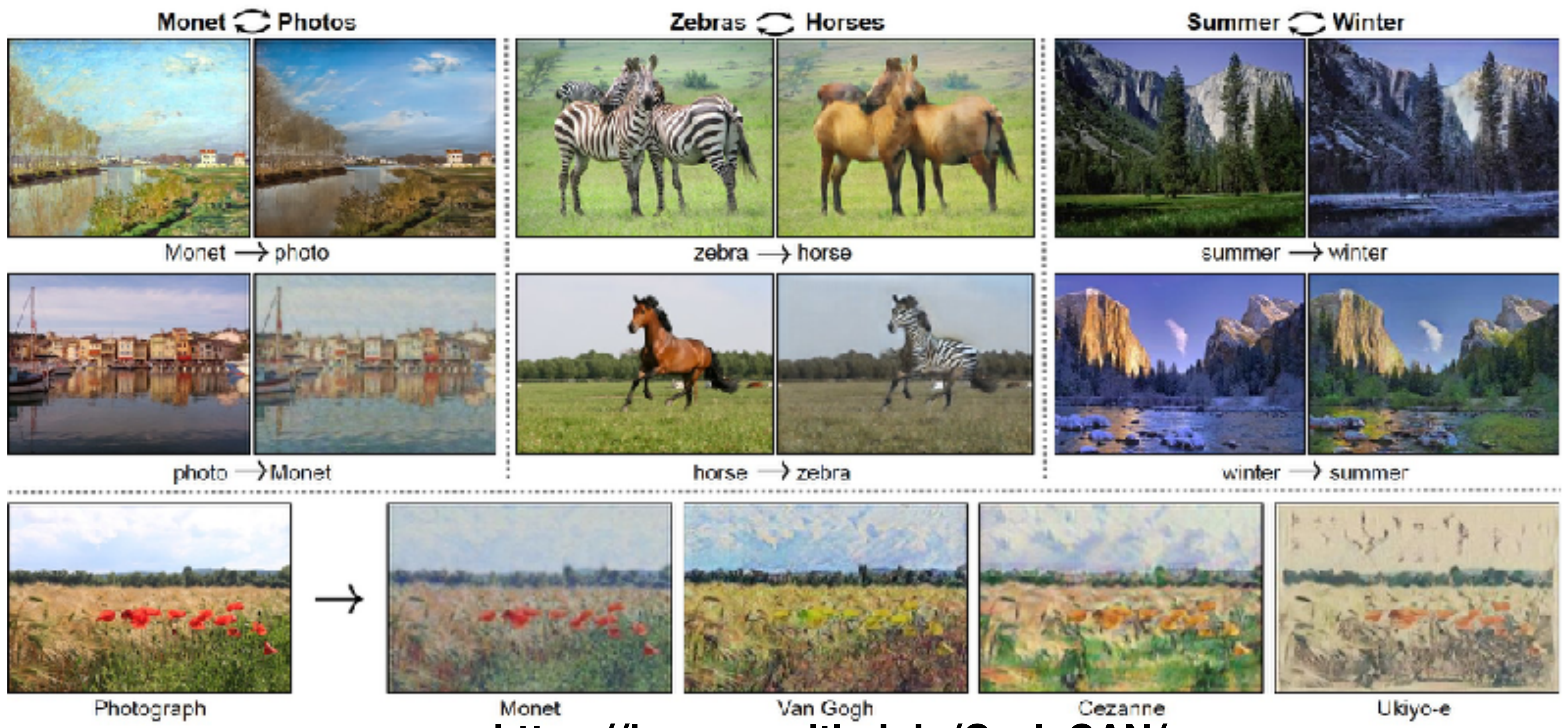
https://mi.eng.cam.ac.uk/projects/segnet/#publication

Generative Adversarial Network



https://medium.com/@sh.tsang/review-gan-generative-adversarial-nets-gan-e12793e1fb75

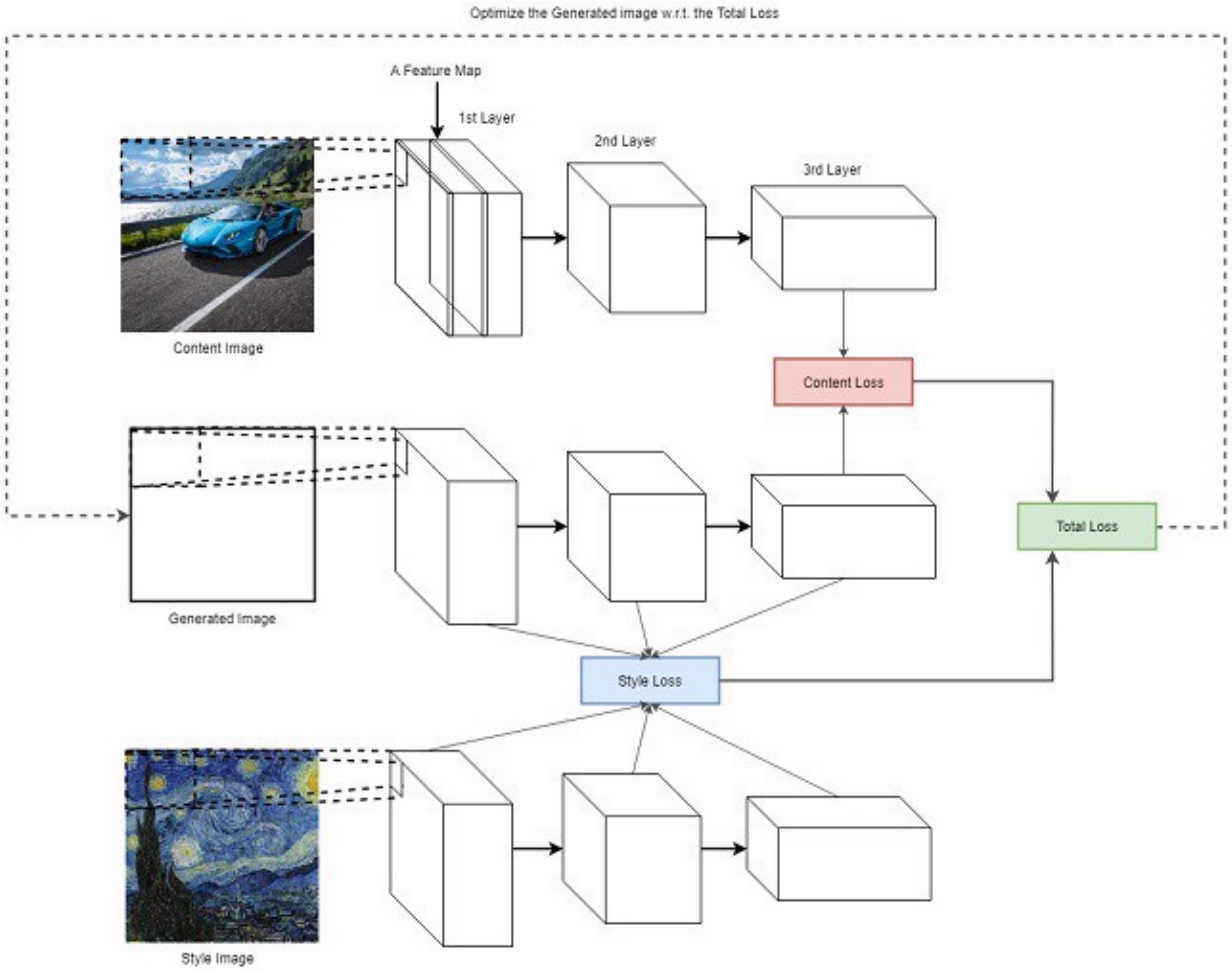
https://www.lyrn.ai/2018/12/26/a-style-based-generator-architecture-for-generative-adversarial-networks/



https://junyanz.github.io/CycleGAN/

Style Transfer

- Generates new image w.r.t. the combination of the Content Loss and the Style Loss



https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-neural-style-transfer-ef88e46697ee

Neural Networks Tools

TensorFlow

- General mathematical graph framework usually used for machine learning and neural networks applications

- Fully open-source

- Supports Python and C++

- See https://www.tensorflow.org

PyTorch

- Framework for GPU matrix multiplication acceleration and parallelisation.
- Open source code
- Supports Python and C++
- See https://pytorch.org

Keras

- A high-level API wrapper framework over existing machine learning toolboxes
- Runs on top of TensorFlow, CNTK and Theano
- Simplifies complex DL frameworks into lightweight and easy to use API
- See https://keras.io

TensorRT

- Platform for accelerating deep learning interference on GPU
- Contains the CUDA compiler that optimise neural network interference on a currently installed graphic card
- ONNX - standard for NN models
- Not used for NN learning
- See https://developer.nvidia.com/tensorrt

There are more …

Sonnet

Microsoft Cognitive Toolkit

Microsoft

Chainer

ONNX

GLUON

theano

Caffe

mxnet

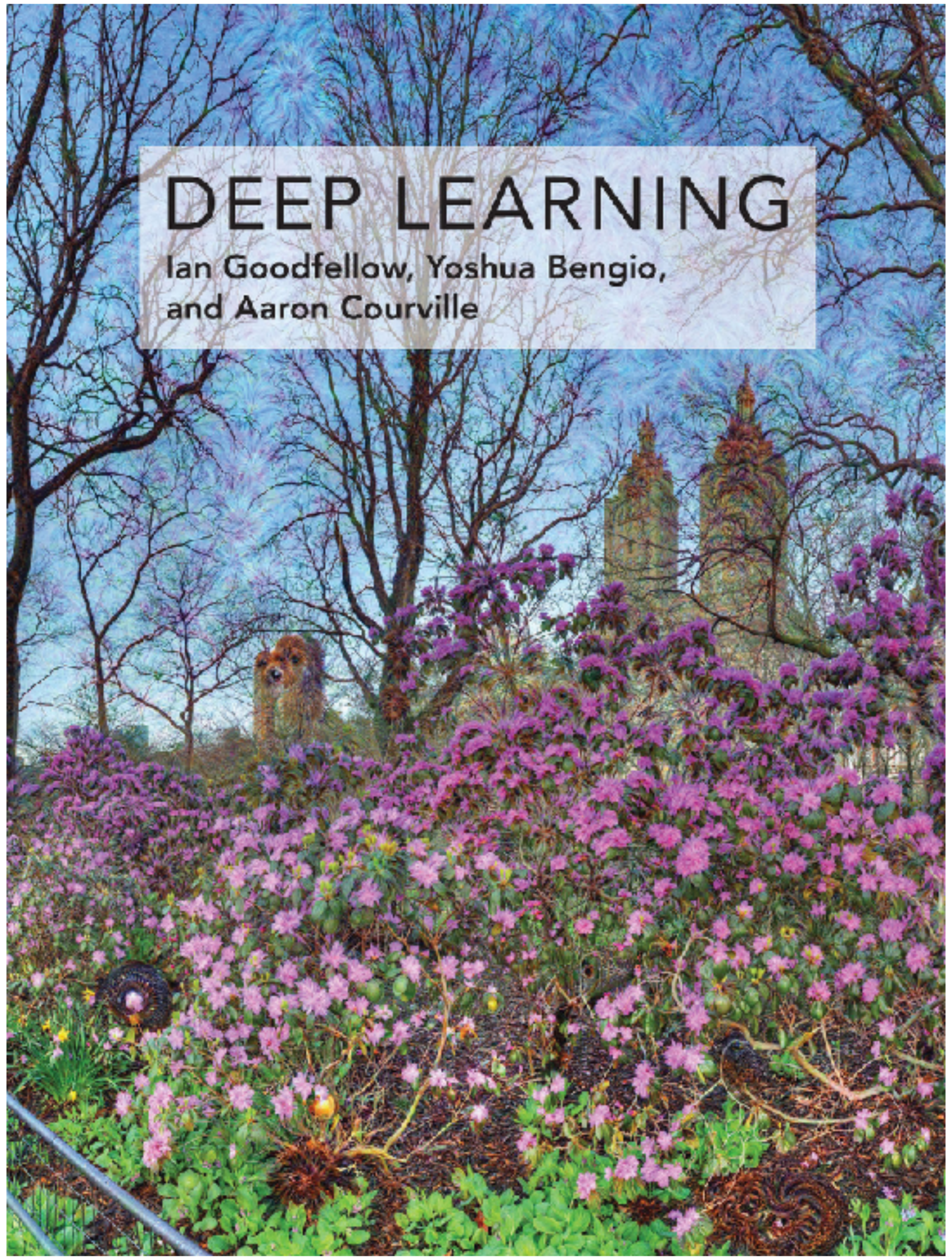
DL4J

Comparison

	TensorFlow	PyTorch	K	TensorRT
Learning:	✓	✓	✓	✗
Deployment:	✓	✓	✓	✓
Languages:	Python, C++	Python, C++	Python	C++
Usage Difficulty:	High	Mid	Low	High
Good for:	Pros	Researchers	Beginners	Runtime

https://playground.tensorflow.org

References



Deep Learning
Goodfellow, Bengio, Courville
MIT Press

http://www.deeplearningbook.org

References

[1] - Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv: 1409.1556*

[2] - He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778)

[3] - Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q., 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708)

[4] - Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680)

[5] - Badrinarayanan, V., Kendall, A. and Cipolla, R., 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *39*(12), pp.2481-2495.

[6] - Gatys, L.A., Ecker, A.S. and Bethge, M., 2015. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*.

[7] - Han, D., Kim, J. and Kim, J., 2017. Deep pyramidal residual networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5927-5935)

[8] - Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), pp.1929-1958.