Computer Vision – TP9
Introduction to Deep Learning

Miguel Coimbra, Hélder Oliveira

Outline

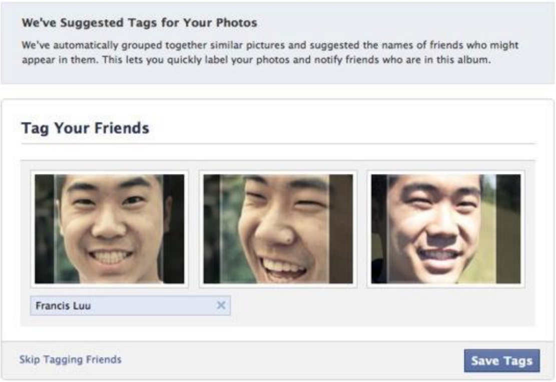
- What is deep learning?
- Artificial neural networks
- Convolutional neural networks
- CNN architectures

Outline

- What is deep learning?
- Artificial neural networks
- Convolutional neural networks
- CNN architectures

Deep learning:
did you hear about that?

- Google image recognition
- Facebook face recognition
- Image caption generation
- Google translator
- DeepMind AlphaGo player
- Netflix, Amazon, Spotify recommendation engines
- Protein folding
- Sentiment analysis
- Etc…


We've Suggested Tags for Your Photos

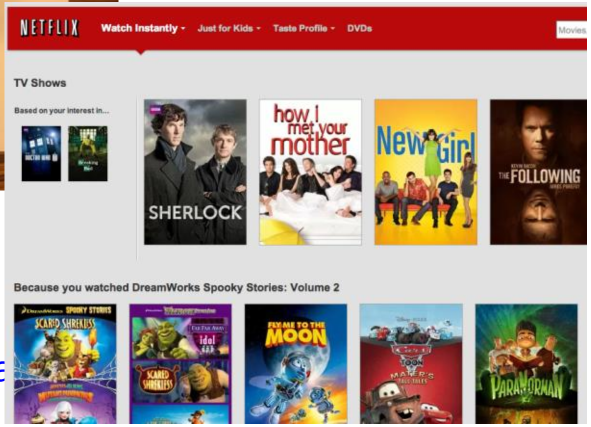
We've automatically grouped together similar pictures and suggested the names of friends who might appear in them. This lets you quickly label your photos and notify friends who are in this album.

Tag Your Friends

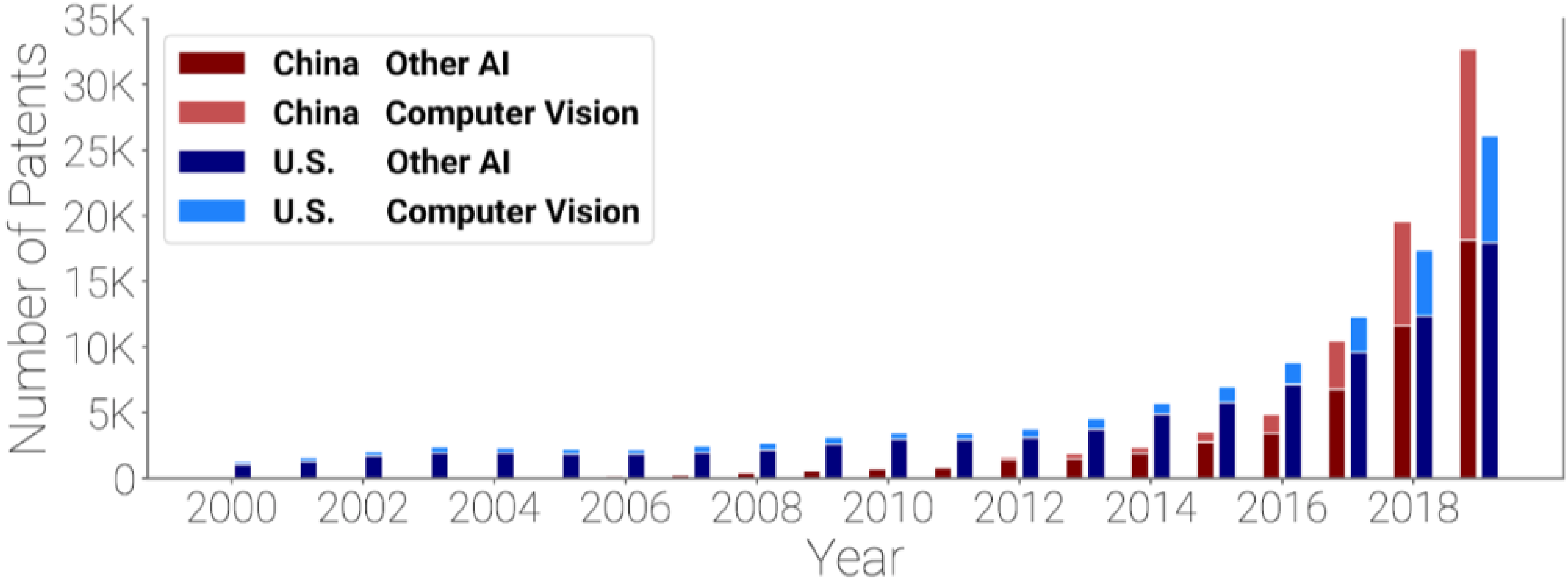
Francis Luu

Skip Tagging Friends

Save Tags



Deep learning and Computer Vision



https://cset.georgetown.edu/wp-content/uploads/CSET-Patent-Landscape-for-Computer-Vision.pdf

More specifically

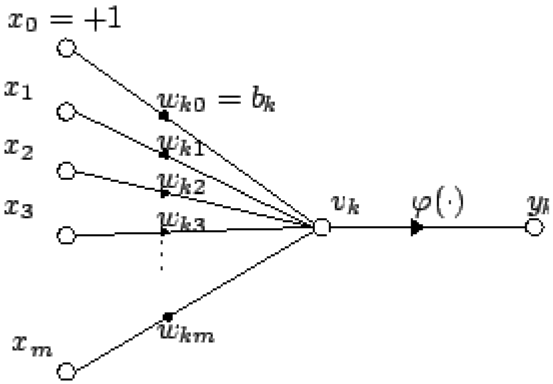
- Deep learning refers to a class of learning algorithms
- They are based on the use of a specific kind of classifiers: neural networks (NNs)

Outline

- What is deep learning?
- Artificial neural networks
- Convolutional neural networks
- CNN architectures

Artificial Neuron

- Also called the **McCulloch-Pitts neuron**
- Passes a **weighted sum of inputs**, to an **activation function**, which produces an **output** value



$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj} x_j\right)$$

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 7:115 - 133.

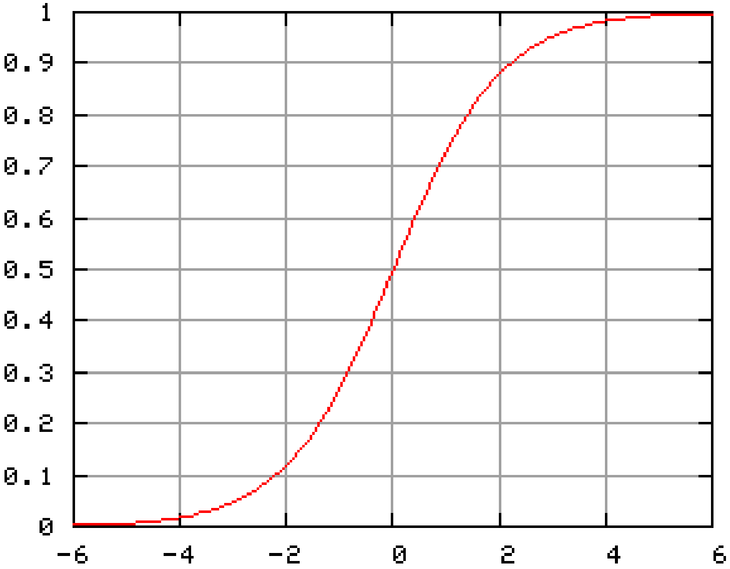
Sample activation functions

- Rectified Linear Unit (ReLU)

$$y = \begin{cases} u, & \text{if } u \geq 0 \\ 0, & \text{if } u < 0 \end{cases} \quad , \quad u = \sum_{i=1}^{n} w_i x_i$$

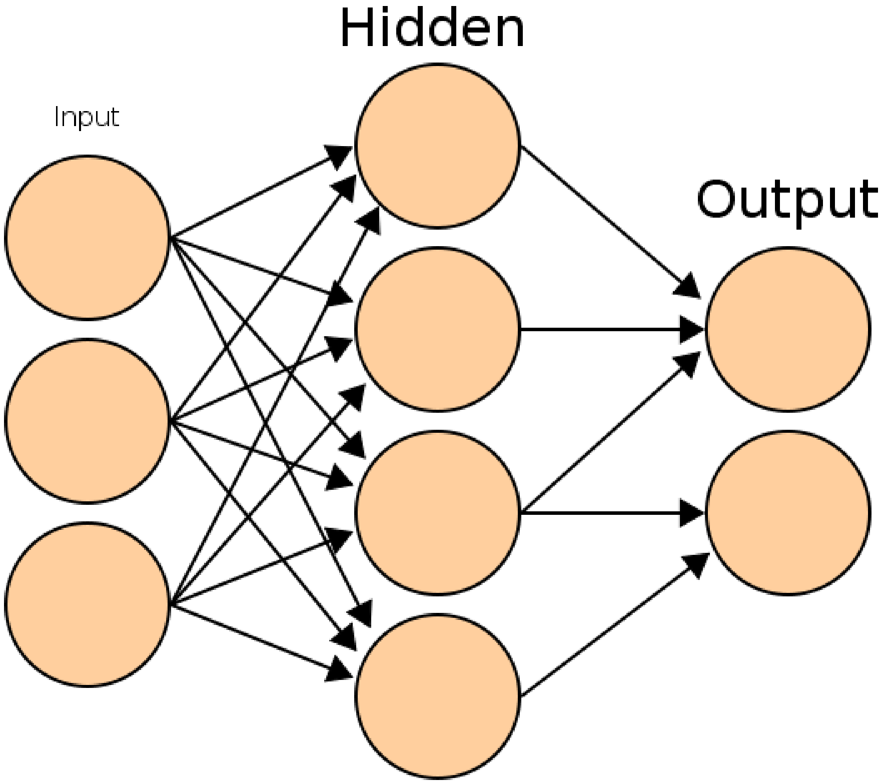
- Sigmoid function

$$y = \frac{1}{1+e^{-u}}$$



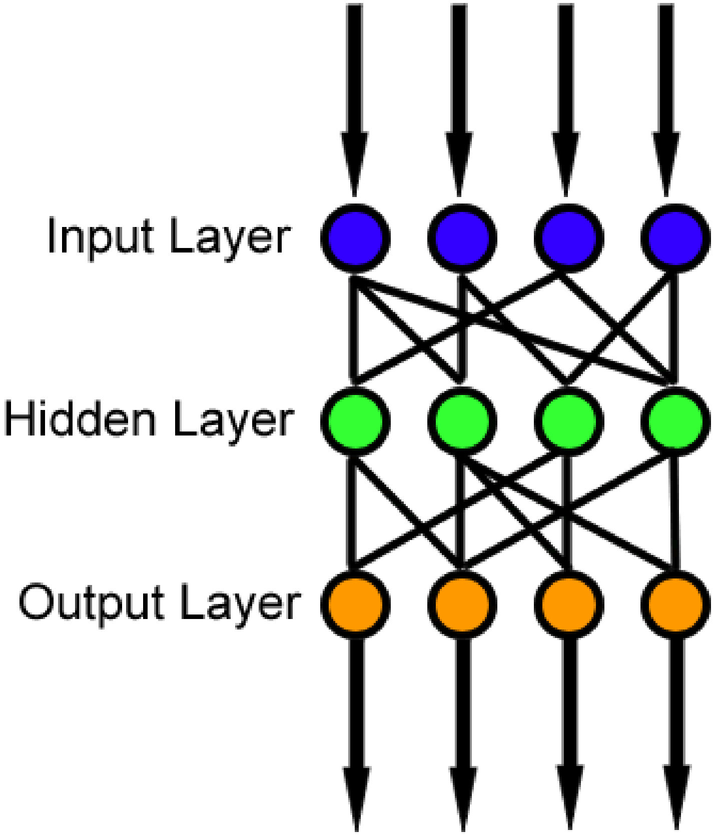
Artificial Neural Network

- Commonly refered as **Neural Network**
- Basic principles:
  - One neuron can perform a simple decision
  - Many **connected** neurons can make more **complex decisions**



Feedforward neural network

- Simplest type of NN.
- Has no *cycles*.
- Input layer
  - Need as many neurons as coefficients of my *feature vector*.
- Hidden layers.
- Output layer
  - Classification results.



Output layer

- Output values correspond to class probabilities
  - 2-class problem: sigmoid activation
  - N-class problem: softmax activation

$$\mathrm{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Characteristics of a NN

- Network configuration
  - How are the neurons inter-connected?
  - We typically use *layers* of neurons (input, output, hidden)
- Individual Neuron parameters
  - Weights associated with inputs
  - Activation function
  - Decision *thresholds*

How do we find these values?

Learning paradigms

- We can define the network configuration
- How do we define neuron ***weights*** and ***decision thresholds***?
  - **Learning** phase
  - We **train** the NN to classify what we want
- Different learning paradigms
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning

Appropriate for **Pattern Recognition**.

Learning

- We want to obtain an **optimal solution** given a set of **observations**
- A **cost function** measures how close our solution is to the **optimal solution**
- Objective of our learning step:
  - Minimize the **cost function**

Backpropagation Algorithm

In formulas

Network output: $\mathrm{Out}(x) = \varphi(\sum_m w_{nm}^{(L)} \varphi(\ldots \varphi(\sum_j w_{\ell j}^{(2)} \varphi(\sum_k w_{jk}^{(1)} x_k)))$

Training set: $\{(\overset{\substack{\text{input}\\ \downarrow}}{x_i}, \overset{\substack{\text{label}\\ \swarrow}}{y_i})\}_{i=1,\ldots,N}$

Optimization: find $[w_{jk}^{(1)}, w_{\ell j}^{(2)}, \ldots, w_{nm}^{(L)}]$ such that

$$\text{minimize} \sum_{i=1}^{N} \mathrm{Loss}(\mathrm{Out}(x_i), y_i)$$

It is solved with (variants of) the **gradient descent**, where gradients are computed via **backpropagation** algorithm

Losses

- They quantify the distance between the output of the network and the true label, i.e., the correct answer
- Classification problems:
  - The output (obtained usually with softmax) is a probability distribution
  - Loss-function: cross-entropy. It can be interpreted in terms of the Kullback-Leibler divergence between probability distributions
- Regression problems:
  - The output is a scalar or a vector of continuous values (real or complex)
  - Loss-function: mean-squared error. It is the distance associated with the L2-norm

Cross-entropy loss

- Cross-entropy

$$H(p|q) = -\sum_i p_i \log q_i$$

Output of neural network

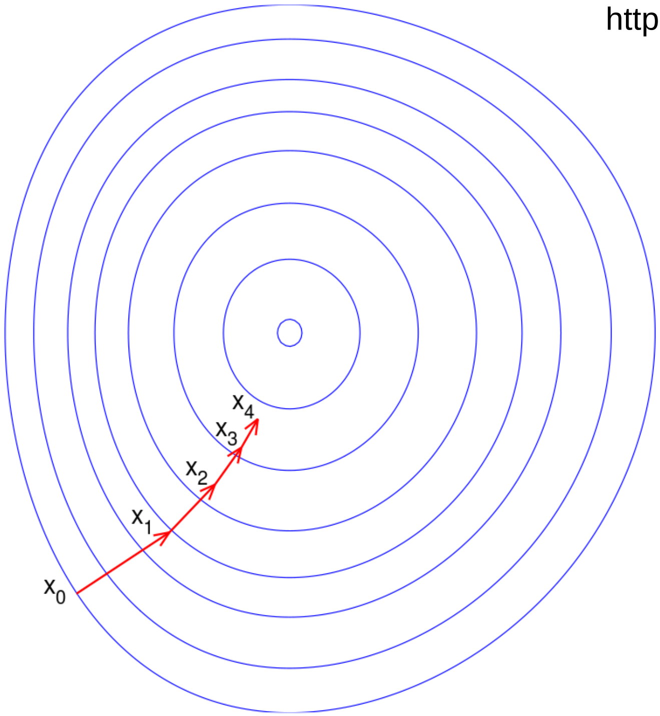
True label, One-hot encoding

$$p = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad q = \begin{bmatrix} 0.75 \\ 0.10 \\ 0.15 \end{bmatrix}$$

$$H(p|q) = -\log 0.75$$

Gradient descent

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



Learning rate

$$w^{i+1} = w^{i} + \lambda \cdot \nabla L(w^{i})$$

Stochastic (mini-batch) gradient descent

- Gradient descent:
  - Compute the gradient of the loss using all available training samples
- Stochastic gradient descent
  - Compute the gradient of the loss using one training sample
- Mini-batch gradient descent
  - Compute the gradient using a subset (mini-batch) of the training samples
- Training epochs
  - Number of passes over the entire training dataset

Deep learning = Deep neural networks

- Deep = high number of hidden layers
  - Learn a larger number of parameters!
- It has been recently (~ in the last 10 years) possible since we have:
  - Access to big amounts of (training) data
  - Increased computational capabilities (e.g., GPUs, TPUs)

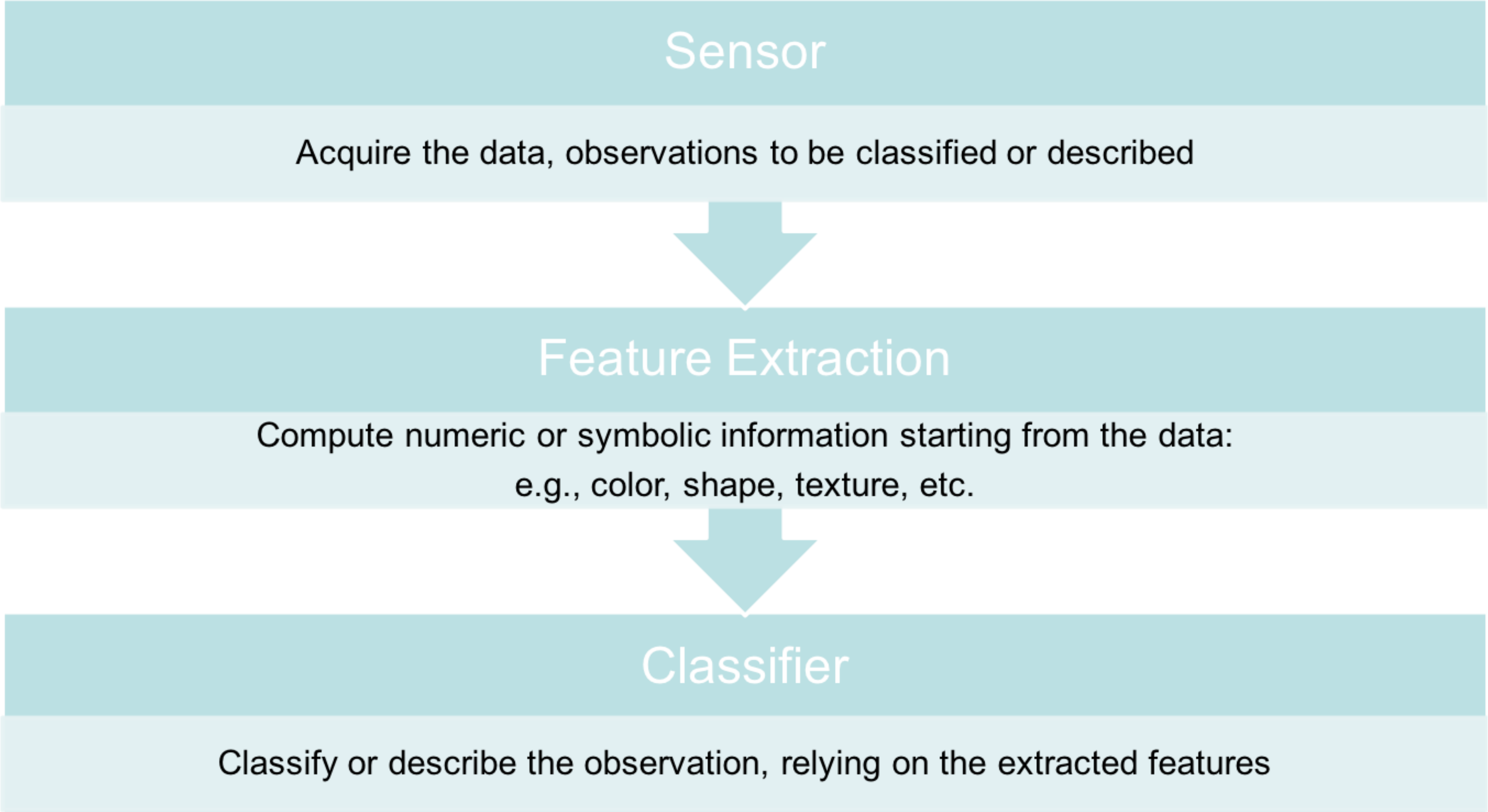
Outline

- What is deep learning?
- Artificial neural networks
- Convolutional neural networks
- CNN architectures

Convolutional neural networks (CNNs)

- Feedforward neural networks
- Weight multiplications are replaced by convolutions (filters)
- **Change of paradigm**: can be directly applied to the raw signal, without computing first *ad hoc* features
- Features are learnt automatically!!

Feature engineering



Sensor

Acquire the data, observations to be classified or described

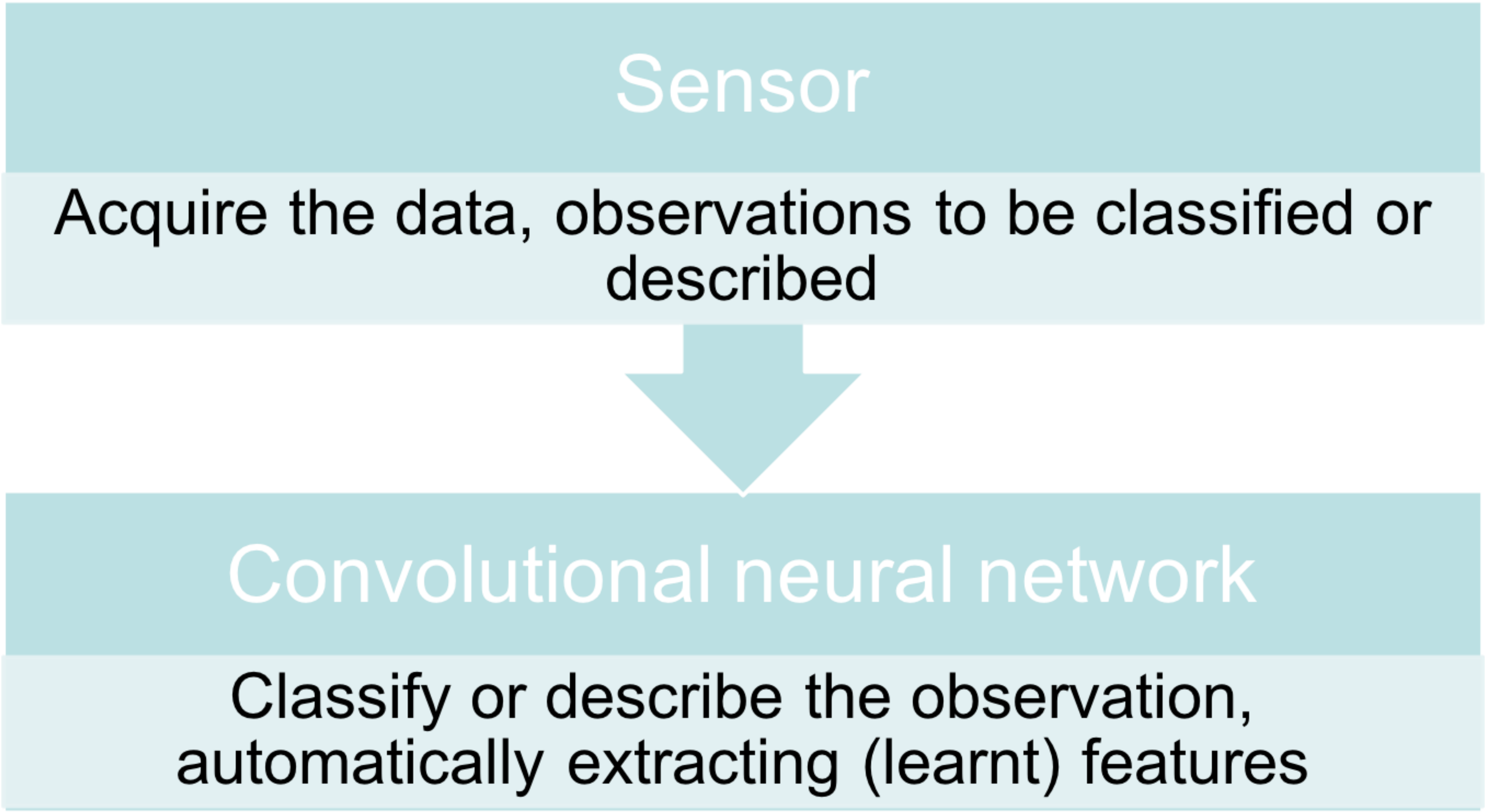
Feature Extraction

Compute numeric or symbolic information starting from the data:
e.g., color, shape, texture, etc.

Classifier

Classify or describe the observation, relying on the extracted features

End-to-end learning



Sensor

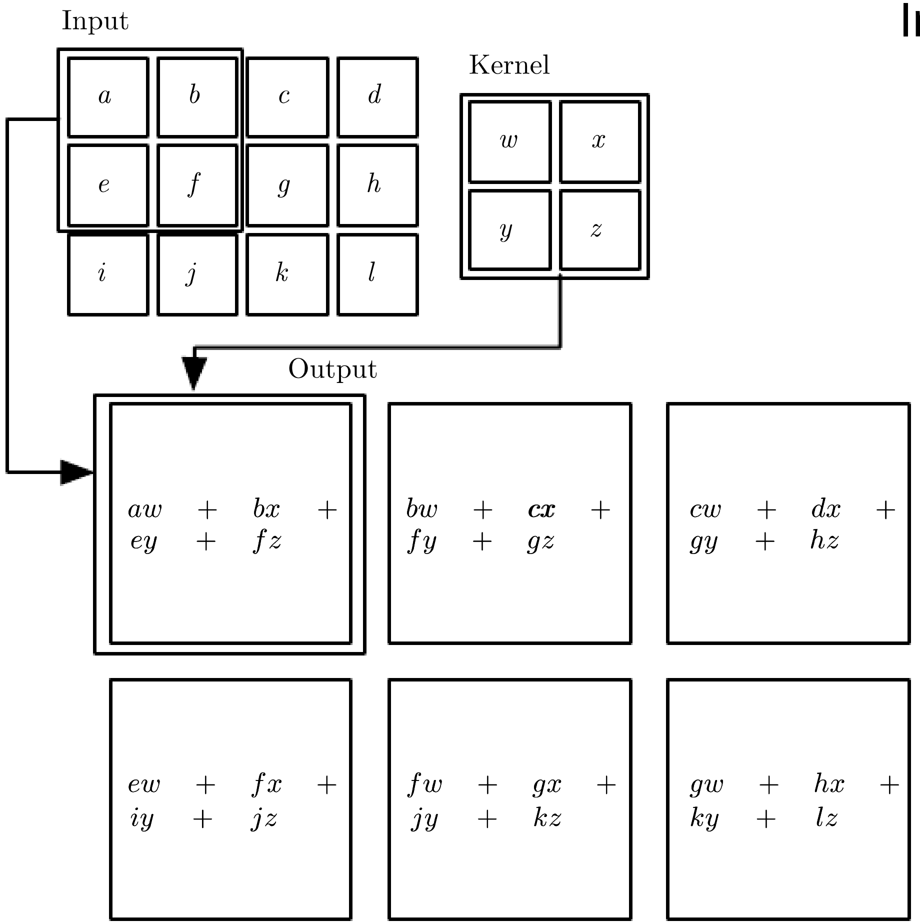
Acquire the data, observations to be classified or described

Convolutional neural network

Classify or describe the observation, automatically extracting (learnt) features

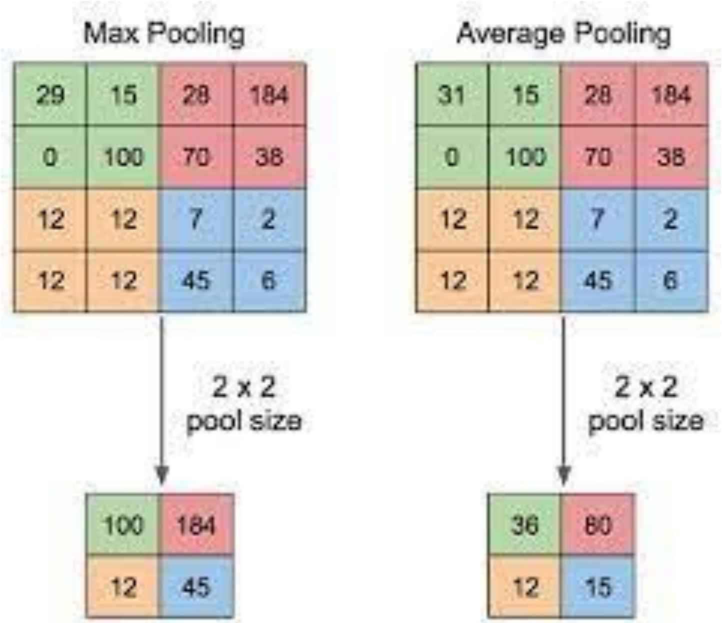
Convolution

In this example,
Stride = 1 x 1



I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. Vol. 1.
Cambridge: MIT press, 2016.

Pooling



Max Pooling

29	15	28	184
0	100	70	38
12	12	7	2
12	12	45	6

100	184
12	45

Average Pooling

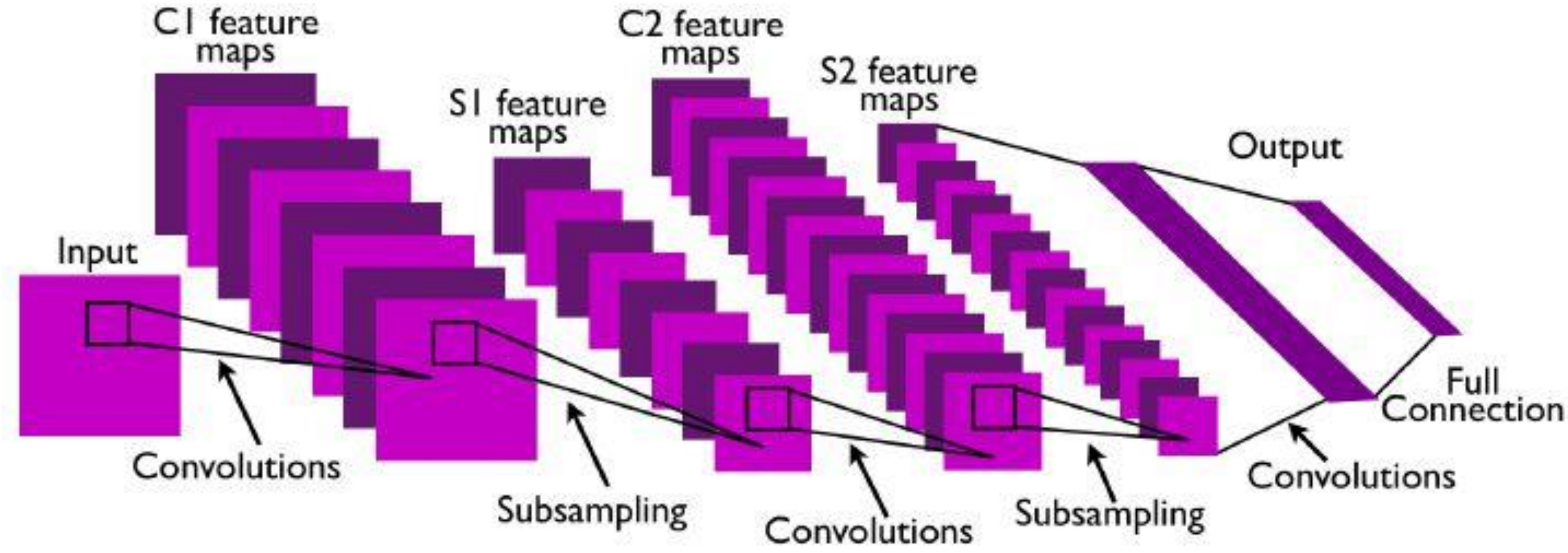
31	15	28	184
0	100	70	38
12	12	7	2
12	12	45	6

36	80
12	15

- Reduce data dimension
- Invariance against small translations

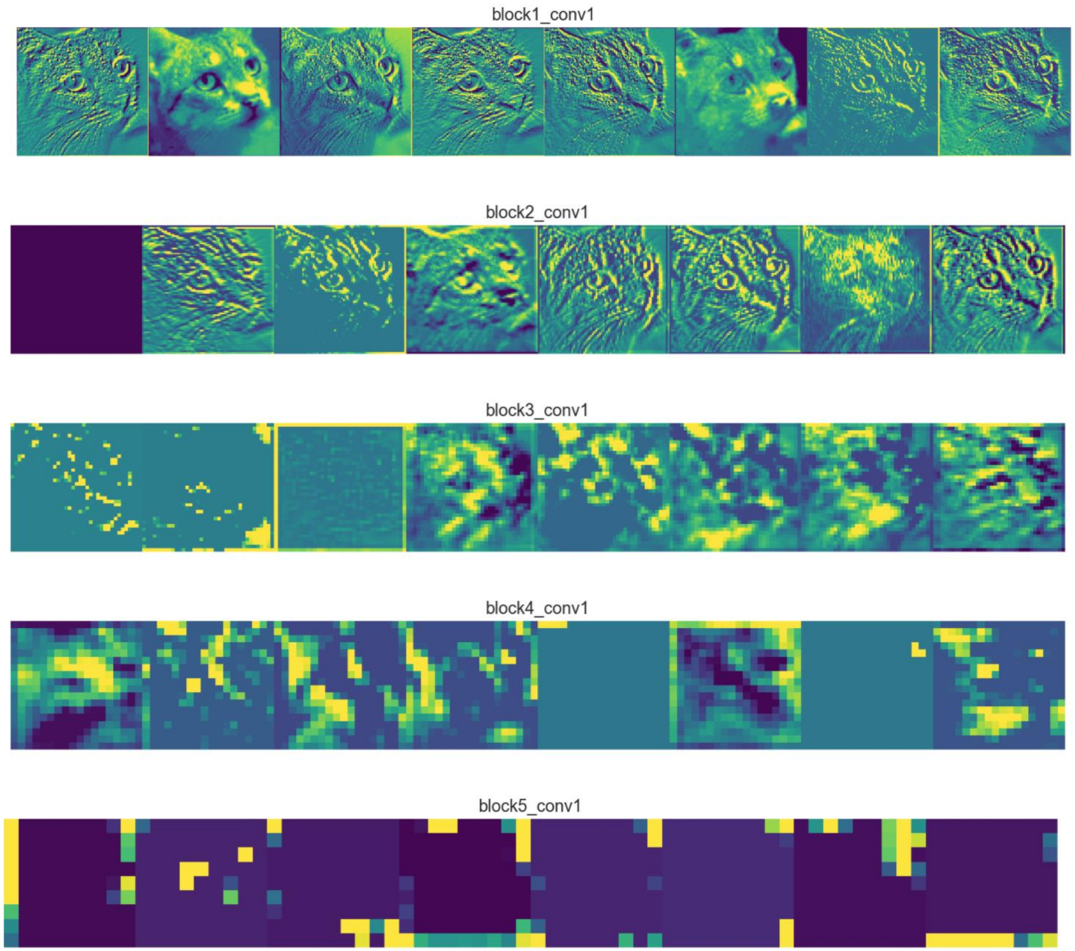
Yani M. Application of transfer learning using convolutional neural network method for early detection of terry’s nail. In Journal of Physics: Conference Series 2019 May 1 (Vol. 1201, No. 1, p. 012052). IOP Publishing

CNN example



- Convolutional layers, followed by nonlinear activation and subsampling
- Output of hidden layers (feature maps) = features learnt by the CNN
- Before classification, fully connected layers (as in “standard” NN)

Automatically learnt features



Retain most information (edge detectors)

Towards more abstract representation

Encode high level concepts

Sparser representations:
Detect less (more abstract) features

https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

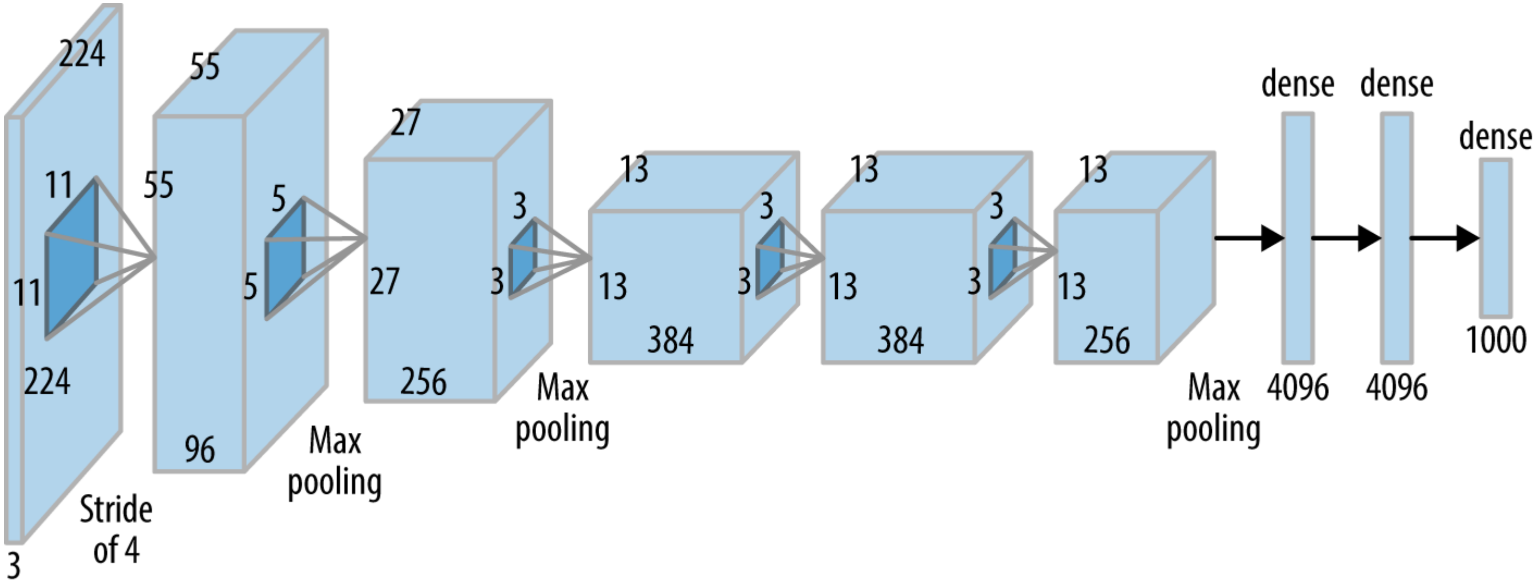
CNN - Properties

- Reduced number of parameters to learn (local features)
- More efficient than dense multiplication
- Specifically thought for images or data with grid-like topology
- Convolutional layers are equivariant to translation (useful for classification!)
- Currently state-of-the-art in several tasks

Outline

- What is deep learning?
- Artificial neural networks
- Convolutional neural networks
- CNN architectures

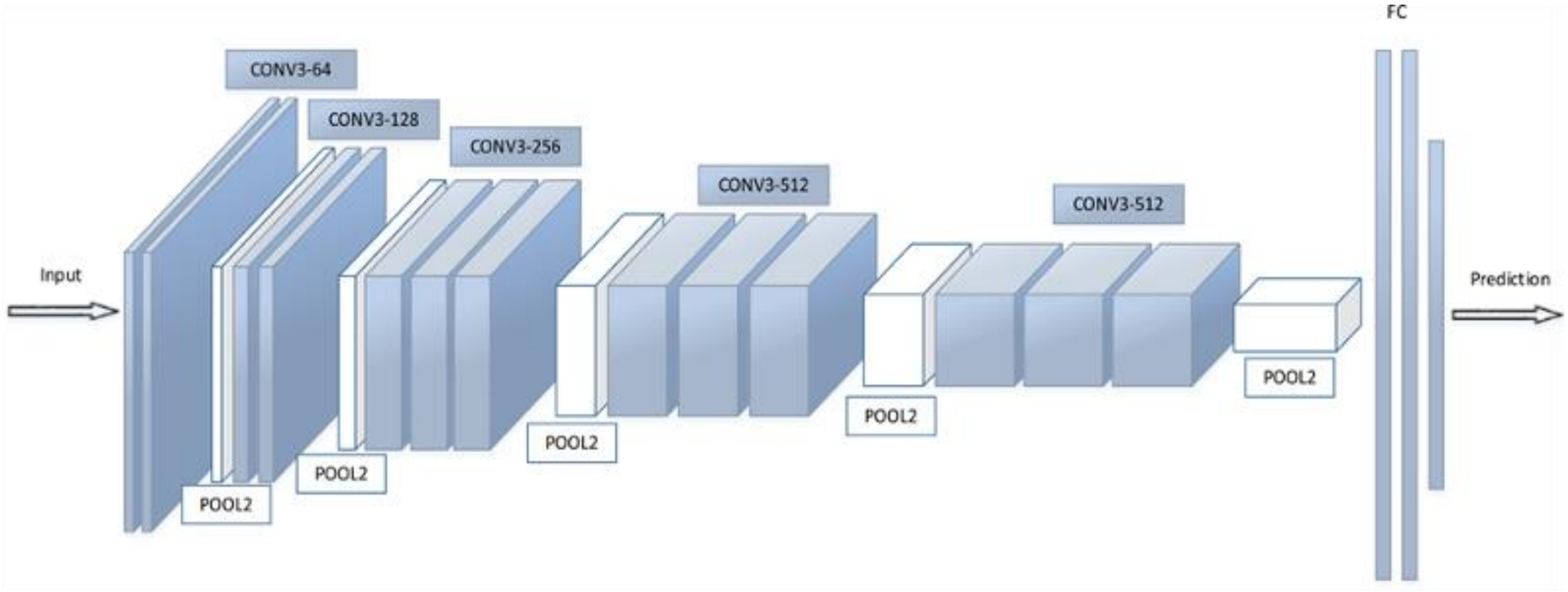
AlexNet



- Winner of ILSVRC 2012
- Marked the beginning of recent deep learning revolution

A. Krizhevsky, I. Sutskever, and G. Hinton. "ImageNet Classification with Deep Convolutional Neural." In *NIPS*, pp. 1-9. 2014.

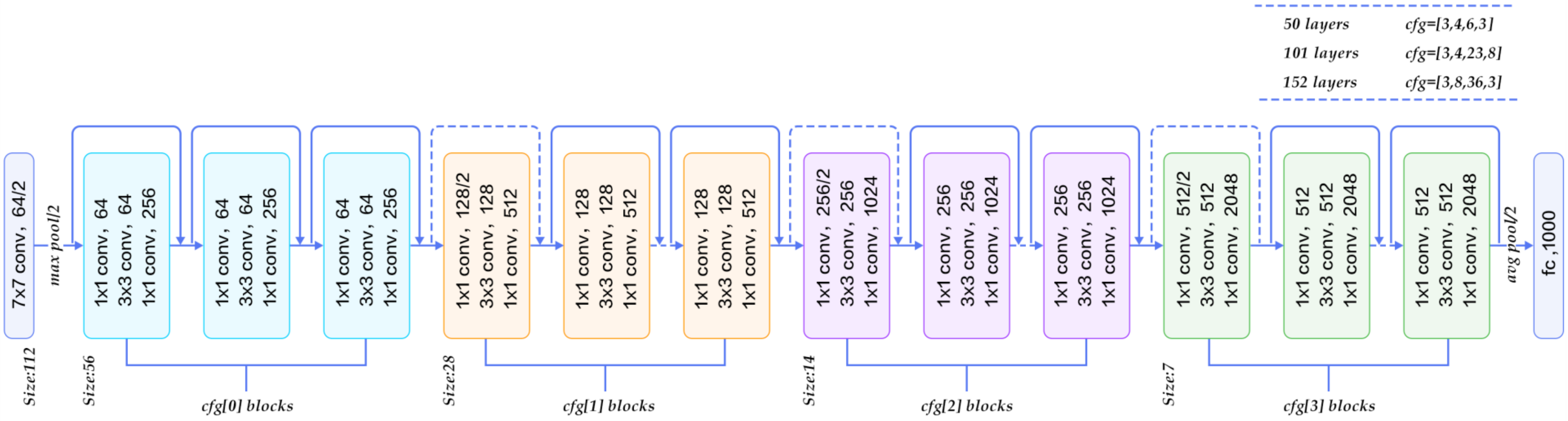
VGG-16



K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learn. Representations, 2015.

- Very small filters (3x3)
- Deeper than AlexNet:16 layers

ResNet

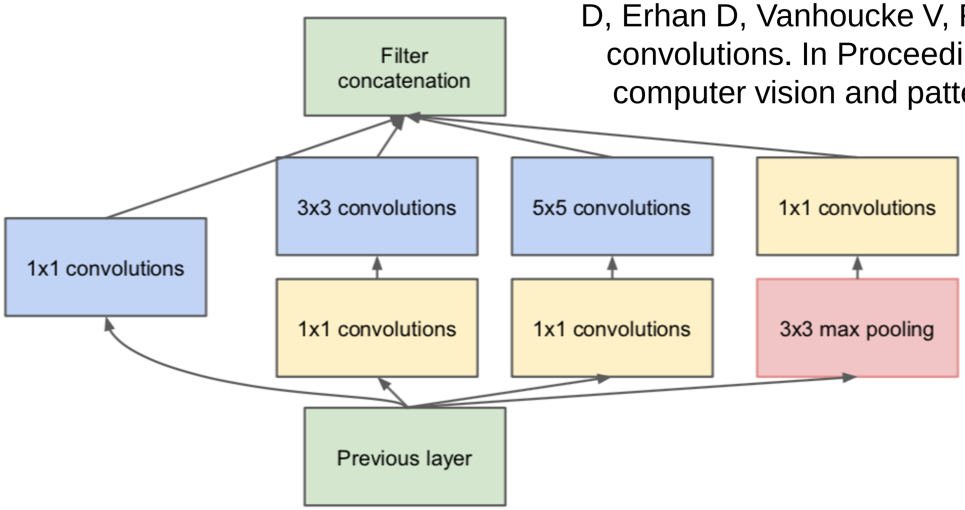


From: https://www.codeproject.com/Articles/1248963/Deep-Learning-using-Python-plus-Keras-Chapter-Re

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

- More layers by using residual connections
- Blocks are actually learning residual functions: easier!
- Less prone to vanishing gradients

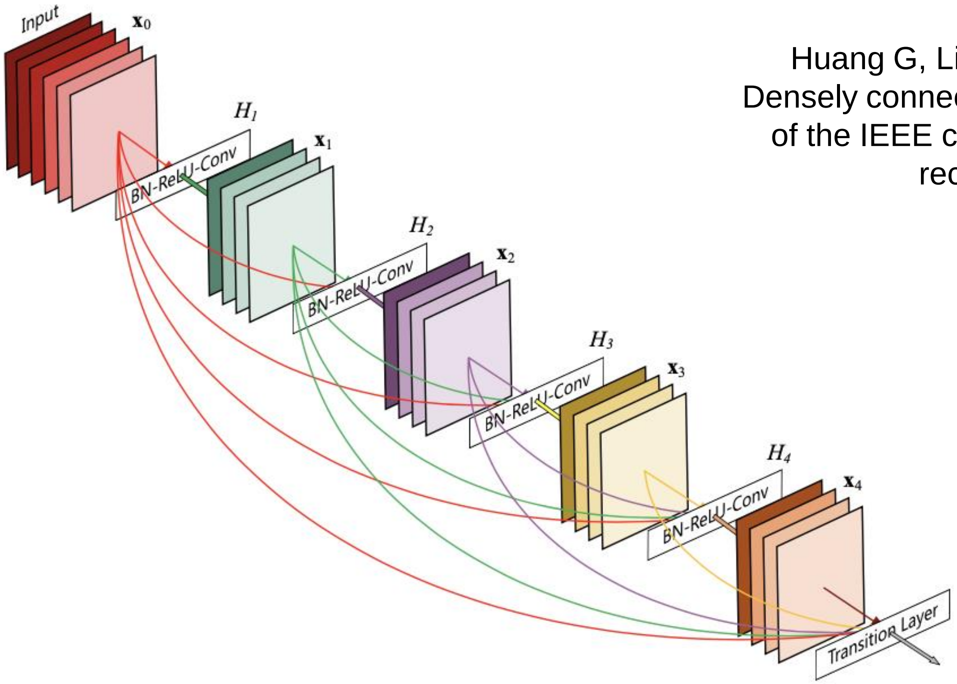
Inception v1



K. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 1-9).

- Filters of different size at the same level
- Capture patterns at different scales
- Computationally efficient implementation

DenseNet

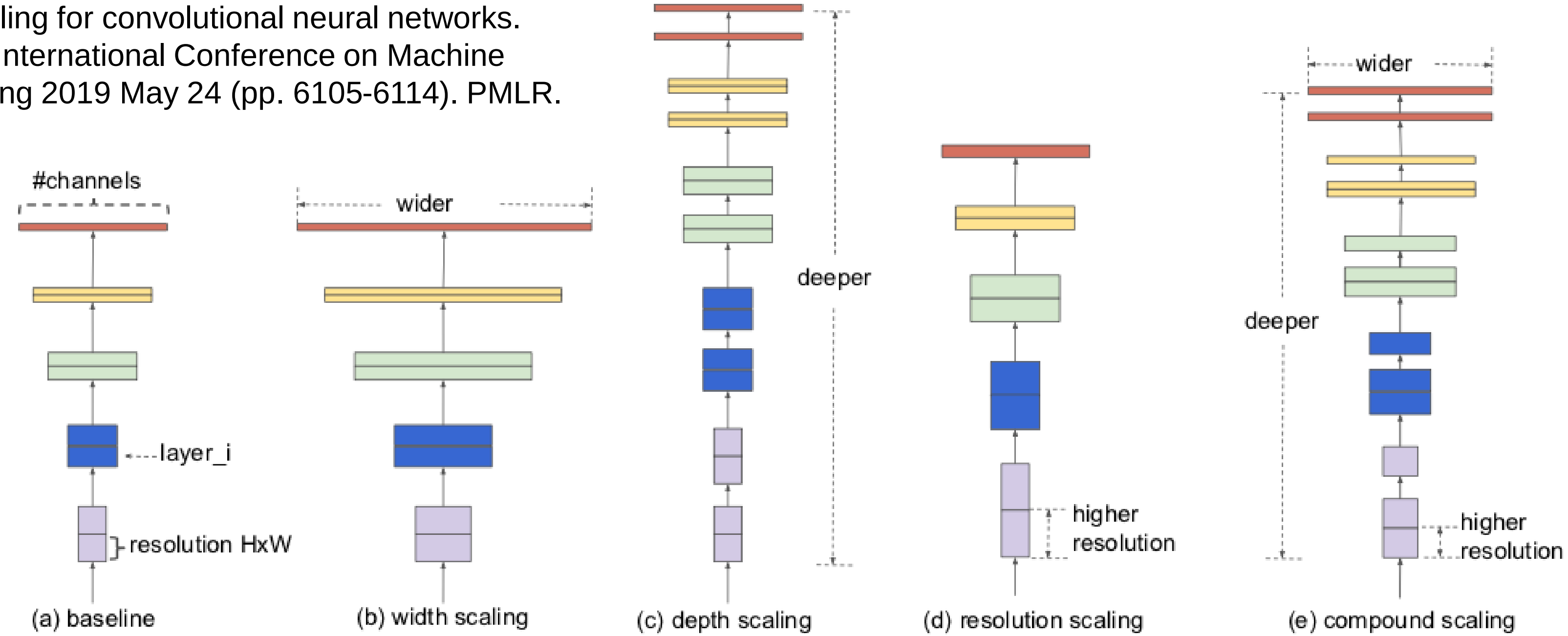


Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 4700-4708).

- Each layer connected to all previous layers
- Alleviate vanishing gradient problem

EfficientNet

Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks. InInternational Conference on Machine Learning 2019 May 24 (pp. 6105-6114). PMLR.



- Systematic approach to scale depth, width, and image resolution

Application challenges

- Great results! But…
  - – Difficult to select best architecture for a problem
  - – Require new training for each task/configuration
  - – (Most commonly) require a large training dataset to generalize well
    - Data augmentation, weight regularization, transfer learning, etc. (we will see them next week)
  - – Still not fully understood why it works so well
    - Recent effort to add explainability
    - Unstable against adversarial examples

To know more…

- Theory
  - I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. Vol. 1. Cambridge: MIT press, 2016. (https://www.deeplearningbook.org/)
- Survey papers
  - "Deep Learning for Visual Understanding," in IEEE Signal Processing Magazine, vol. 34, no. 6, Nov. 2017.
- Tutorials
  - Tensorflow tutorials (https://www.tensorflow.org/tutorials)

To start coding

- Coding frameworks for deep learning
  - TensorFlow (https://www.tensorflow.org/),
    - Version 2 recommended! (it contains Keras)
  - PyTorch (https://pytorch.org/),
  - Theano (http://deeplearning.net/software/theano/),
  - etc.
- High-level wrappers
  - Keras (https://keras.io/),
  - TensorLayer (https://tensorlayer.readthedocs.io/en/stable/),
  - Lasagne (https://lasagne.readthedocs.io/en/latest/),
  - etc.
- GPU strongly recommended!