MSE | MASTER OF SCIENCE IN ENGINEERING

Machine Learning

T-MachLe

10. Deep Learning I

Andres Perez Uribe
Jean Hennebert

Plan

10.1 From shallow to deep networks

10.2 Convolutional Neural Networks

10.3 Visual processing in the brain

10.4 A toy-problem for CNNs

10.5 MNIST digit recognition task

Practical Work 10

Deep learning

- The “Universal approximation” theorem stated that neural networks with a single hidden layer can represent a wide variety of interesting functions when given appropriate parameters.

  Cybenko., G. (1989) "Approximations by superpositions of sigmoidal functions", Mathematics of Control, Signals, and Systems, 2 (4), 303-314

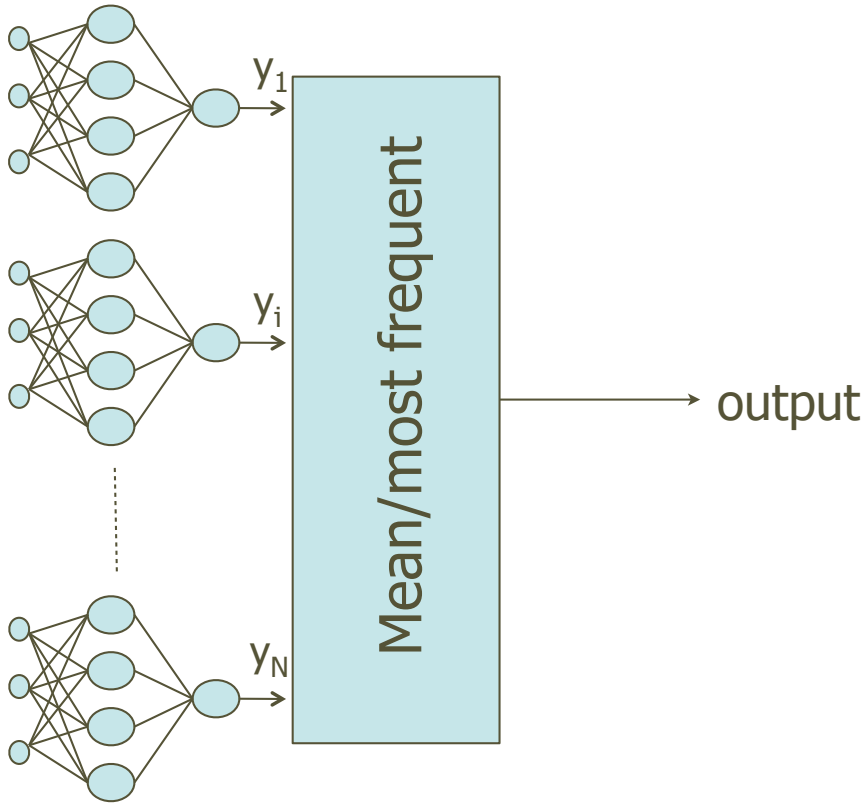
- Before 2006: training a deep feedforward neural network yielded worse results (both in training and in test error), then shallow networks (with 1 or 2 hidden layers) were preferred

“Wide but shallow” neural networks

- The availability of increasing computational capabilities allowed engineers to train multiple shallow models while searching for better generalization performances.
- The result is the development of “ensemble models”. For instance, models composed of multiple neural networks as follows:
  - multiple instances of the same neural network model having been trained on the same database, or
  - multiple instances of the same neural network model having been trained in different subsets of the training database, or
  - multiple variations of shallow neural networks (e.g., having different architectures).

Network ensembles

Ensemble averaging is the process of creating multiple models and combining them to produce a desired output, as opposed to creating just one model. Frequently an ensemble of models performs better than any individual model, because the various errors of the models "average out."



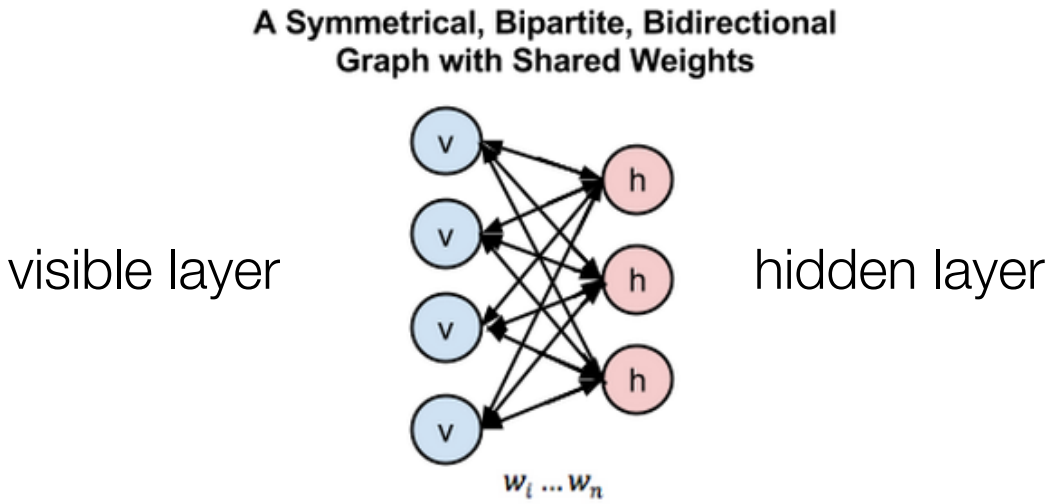
Deep learning: towards deep neural networks

G. Hinton

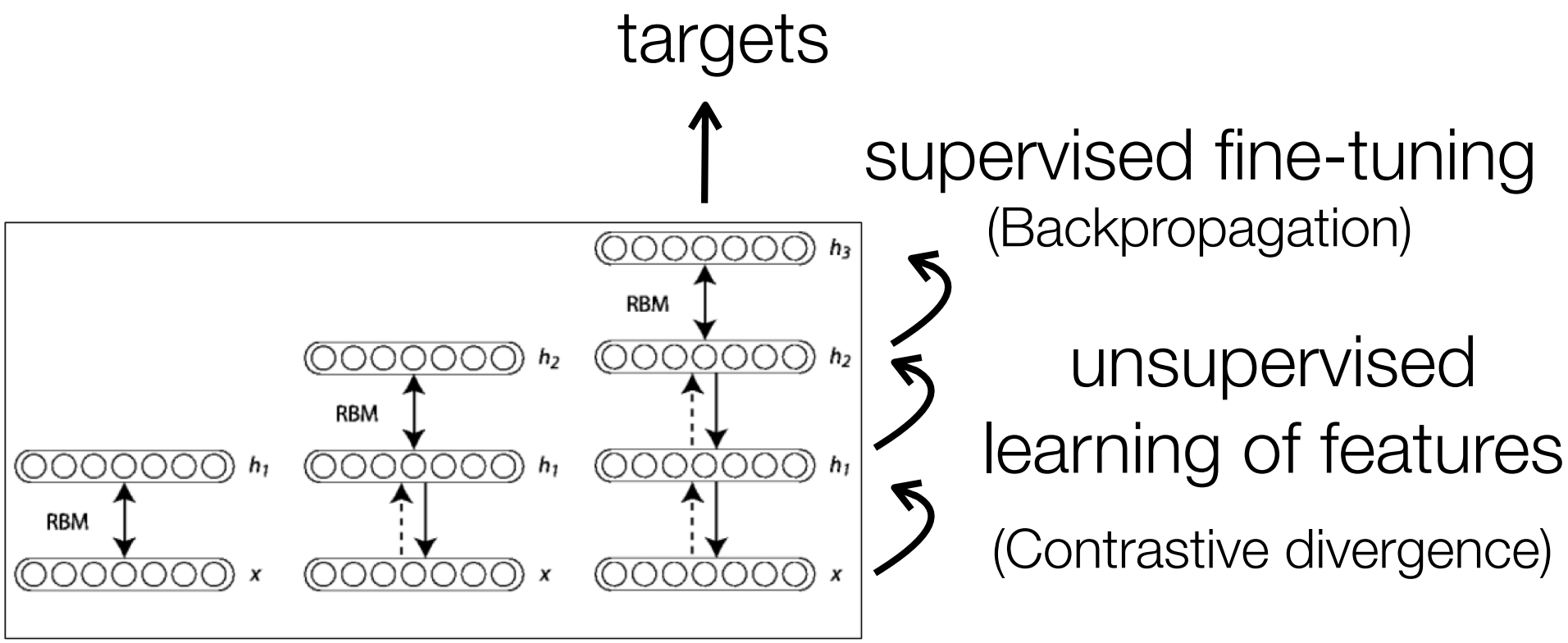
- Instead of learning p(label | data), try to learn p(data), that is a "generative" model
- Recursively learn a layer of feature detectors (in unsupervised way) capable of capturing interesting regularities in the data
- Speech recognition & english-to-chinese translation http://youtu.be/Nu-nlQqFCKg (2012)
- Key people: G. Hinton (U of Toronto & Google), Y. LeCun (NYU & Facebook), Y. Bengio (U of Montreal), J. Schmidhuber (Idsia, Lugano)

Restricted Boltzman Machines (RBM)

- Stochastic neural networks consisting of one layer of visible units (neurons) and one layer of hidden units. Units in each layer have no connections between them and are connected to all other units in the subsequent layer.
- Connections between neurons are bidirectional and symmetric. Information flows in both directions during the training and during the usage of the network and the weights are the same in both directions.



Deep-Belief Networks (Hinton, 2012)



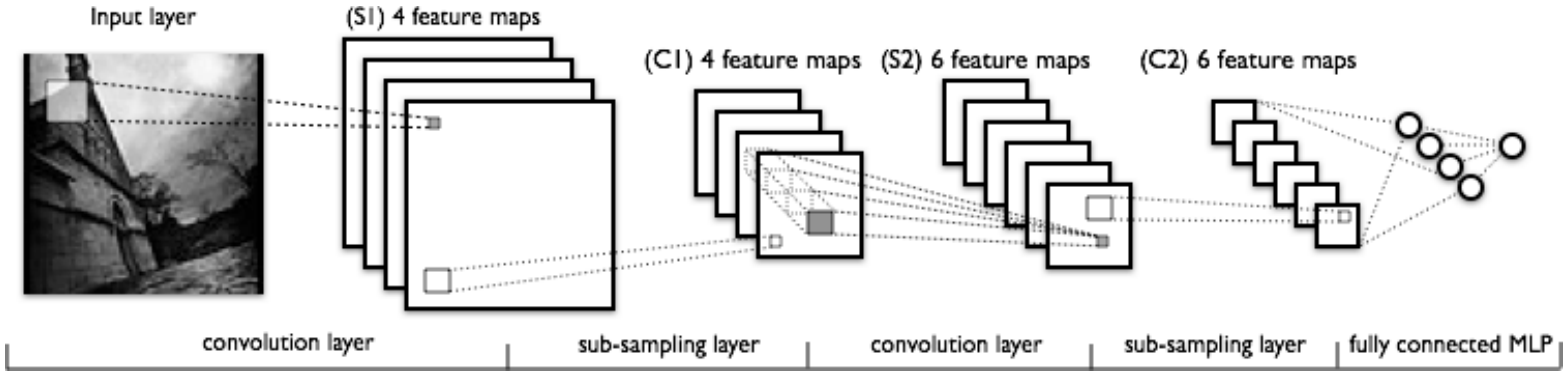
Backpropagation is applied once we have sensible features in each layer, obtained by unsupervised learning between pair of layers. Backpropagation performs local search to fine-tune what has been learned in unsupervised way.

Convolutional Neural Networks

Yann LeCun

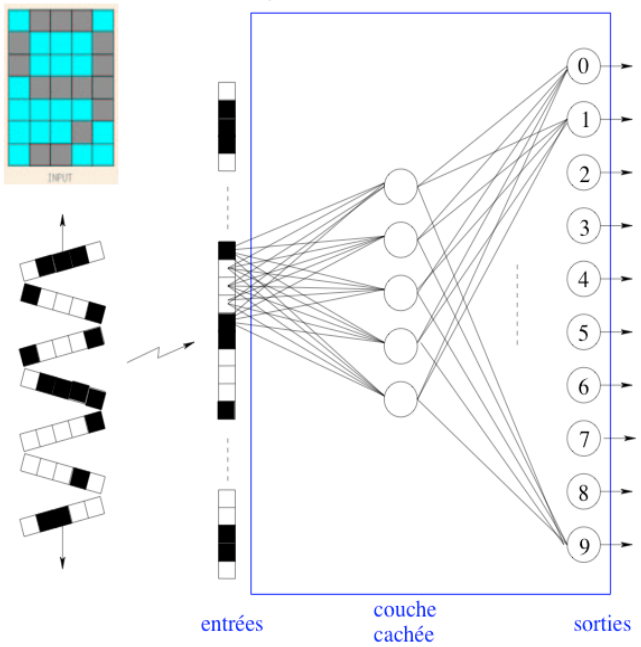
- A neural net architecture with local connections and shared weights introduced by Yann LeCun in 1989.
- After joining AT&T Bell Labs in 1988, he applied convolutional networks to the task of recognizing handwritten characters (the initial goal was to build automatic mail-sorting machines). This work was one of the first (and one of the most cited) demonstrations that Neural Networks could be applied to "real-world" applications

Le Net5

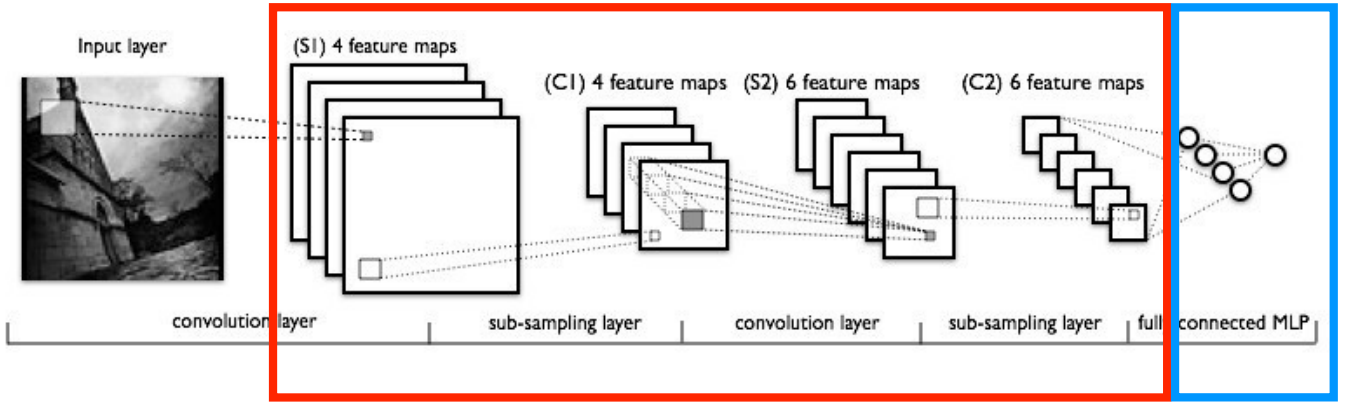


From shallow to deep

flatten input image



shallow feed-forward neural network



feature extraction

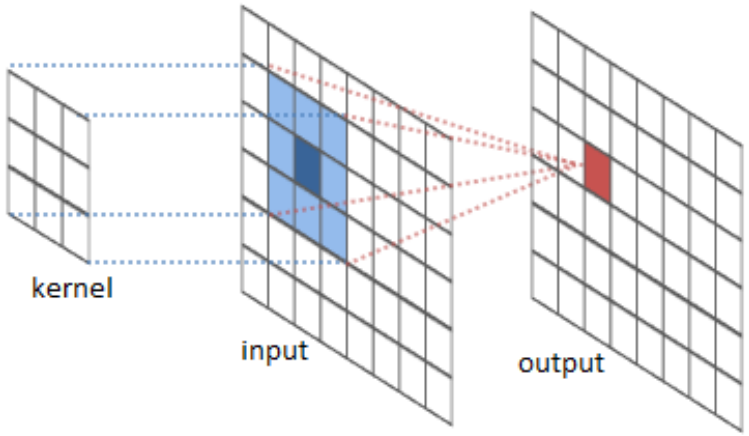
classification

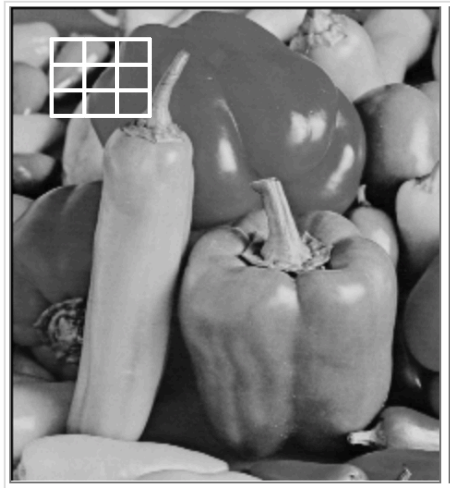
deep convolutional neural network

Convolutions on image processing (1)

- Blurring, sharpening, embossing, and edge detection are typical functions of image processing.
- They are accomplished by means of convolution between a kernel and an image.
- e.g., the Laplacian kernel of an image highlights regions of rapid intensity change
- The Laplacian is therefore often used for edge detection

Convolutions on image processing (2)





x's == pixel colors

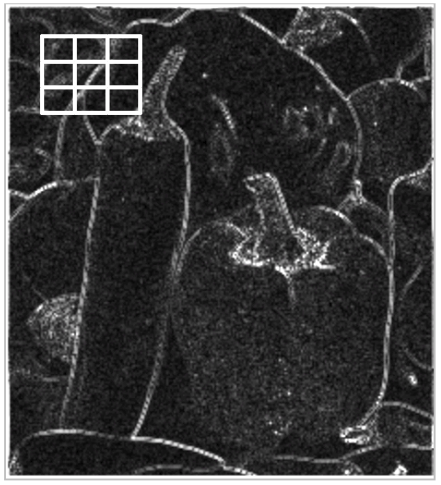
w's

X

0	1	0
1	-4	1
0	1	0

=

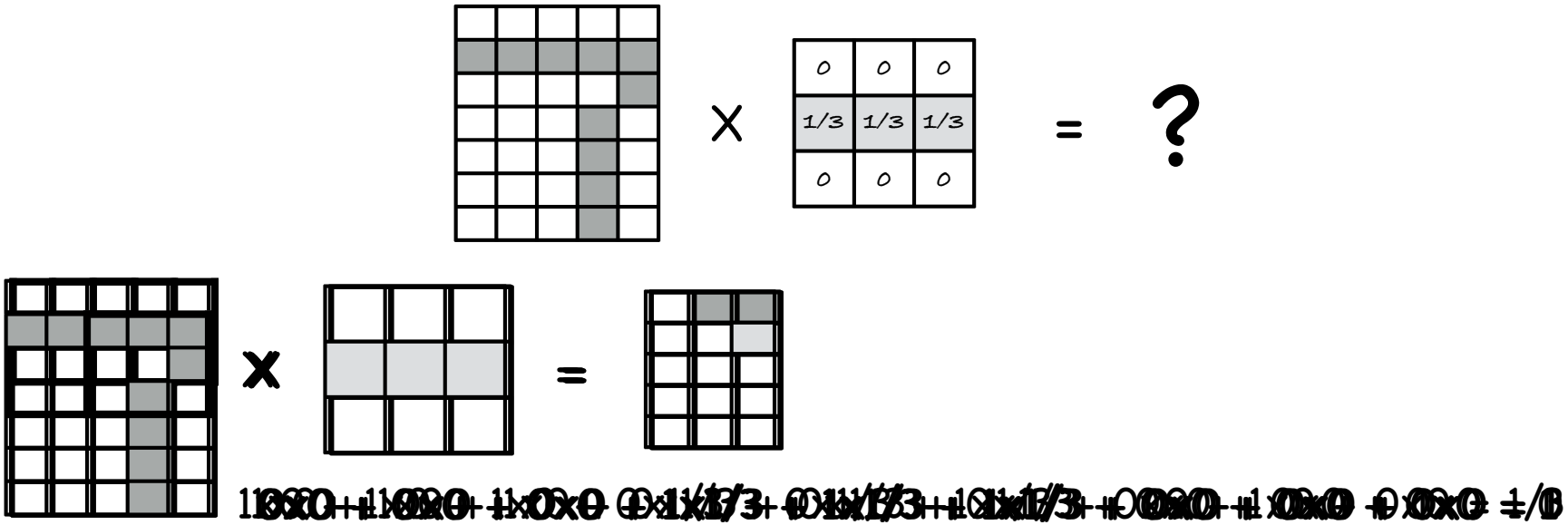
kernel for edge detection

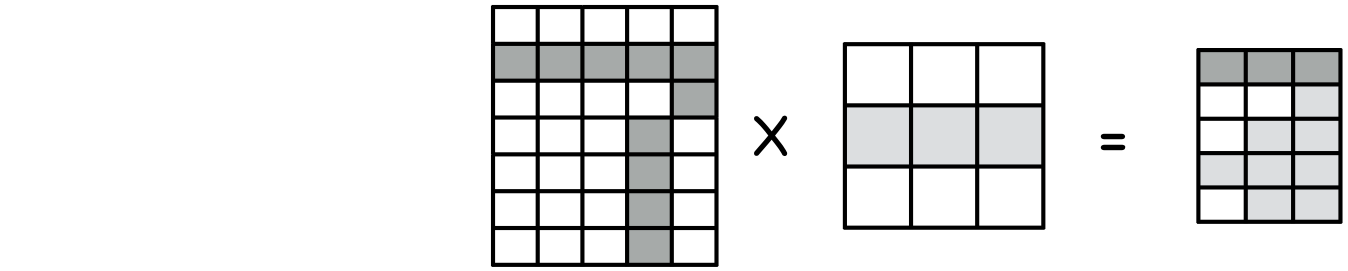


new color = $\sum_j w_j x_j + b$

http://setosa.io/ev/image-kernels/

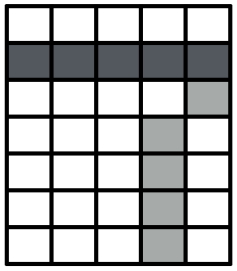
Convolution kernels as pattern detectors (1)

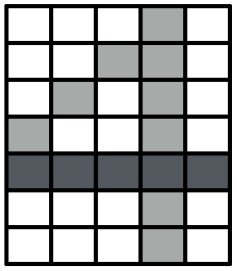


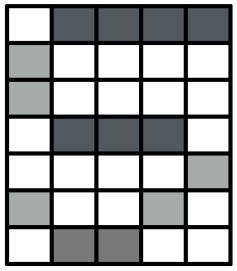


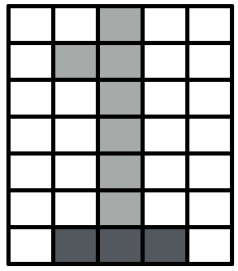
Convolution kernels as pattern detectors (2)

0	0	0
1/3	1/3	1/3
0	0	0

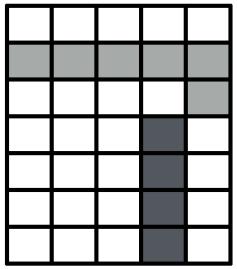


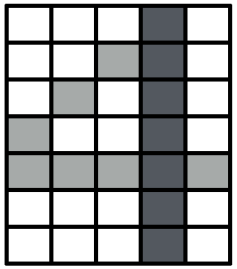


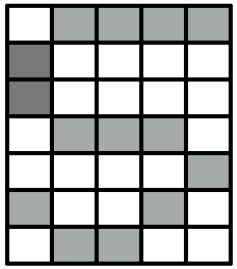


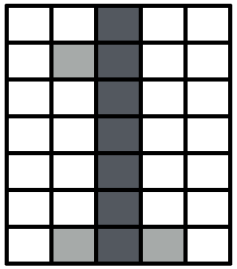


0	1/3	0
0	1/3	0
0	1/3	0







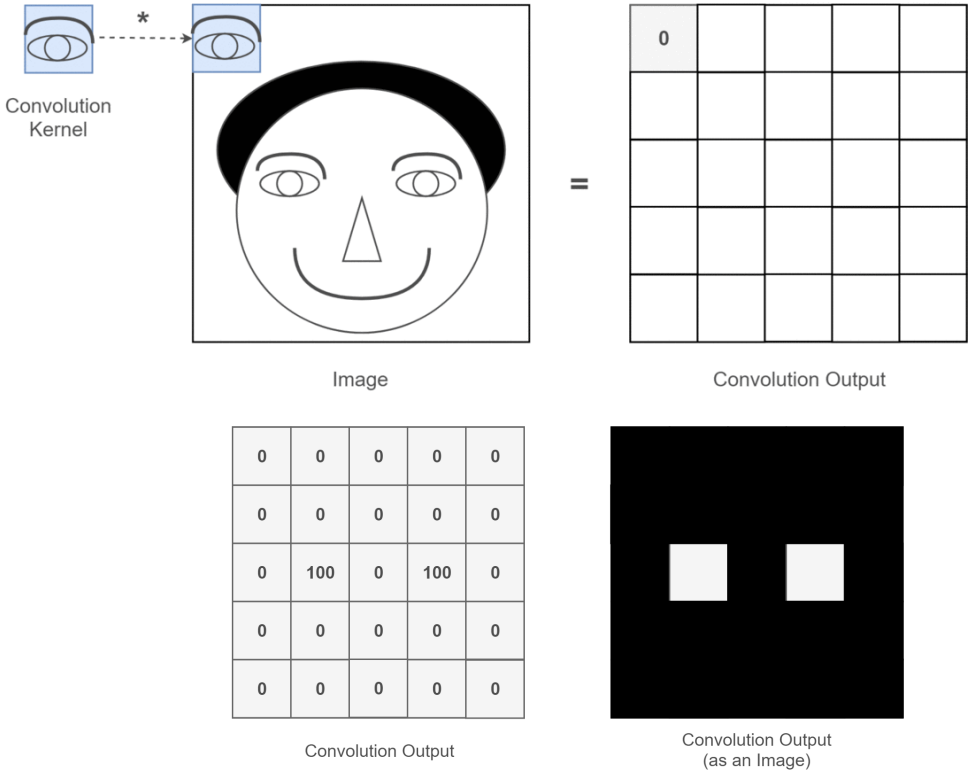


0	0	1/3
0	1/3	0
1/3	0	0

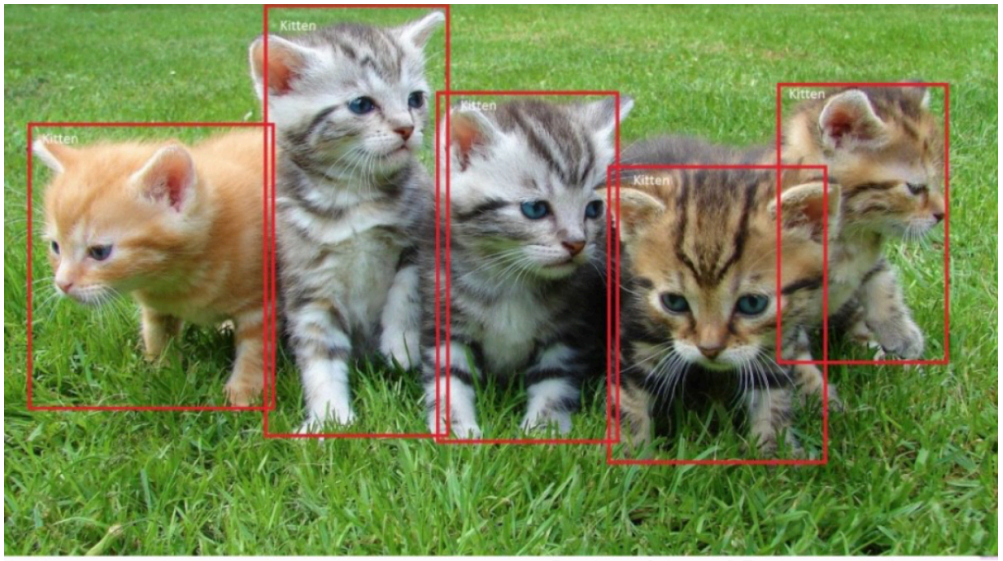
etc...

Convolution kernels as pattern detectors (3)

A convolution kernel (very often referred to as a “filter”) can be used to detect a shape, a color blob or a particular texture in an image



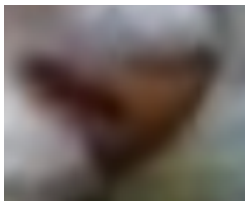
Feature detection for object recognition



In an example of learning to recognize cats, the objective of the training of a CNN is to learn the filters that detect the features that allow the network to recognize a cat independently of its position in the image, its color, its orientation, its size, etc...

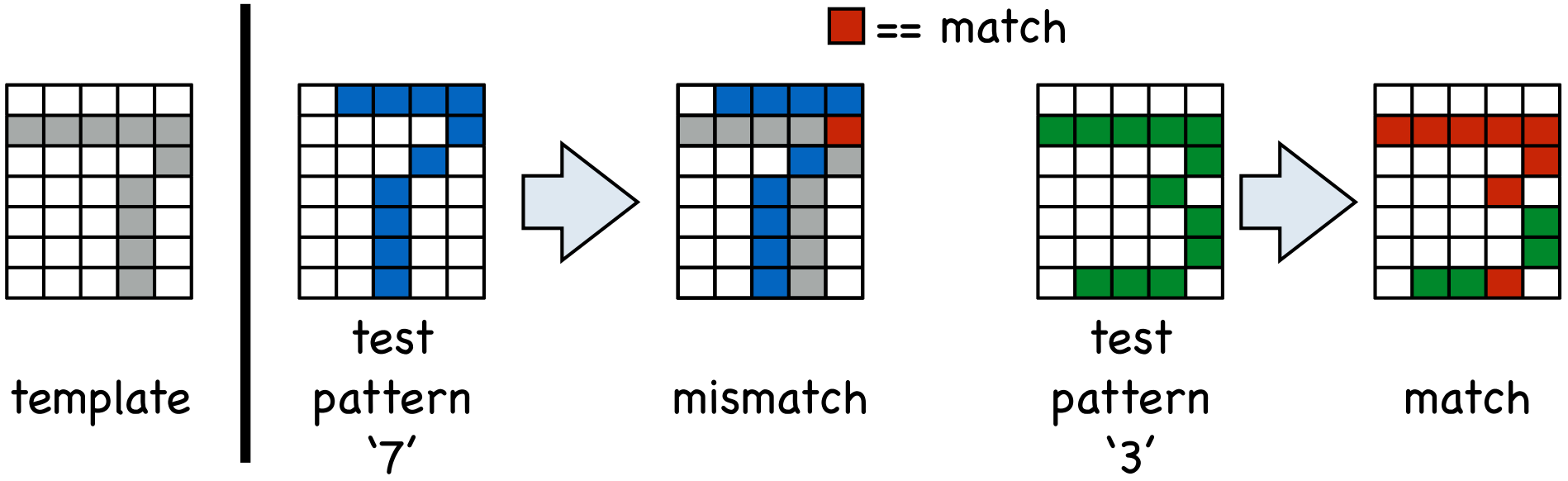
Examples of filters to be learned are: eye, ear, nose and whiskers' detection filters



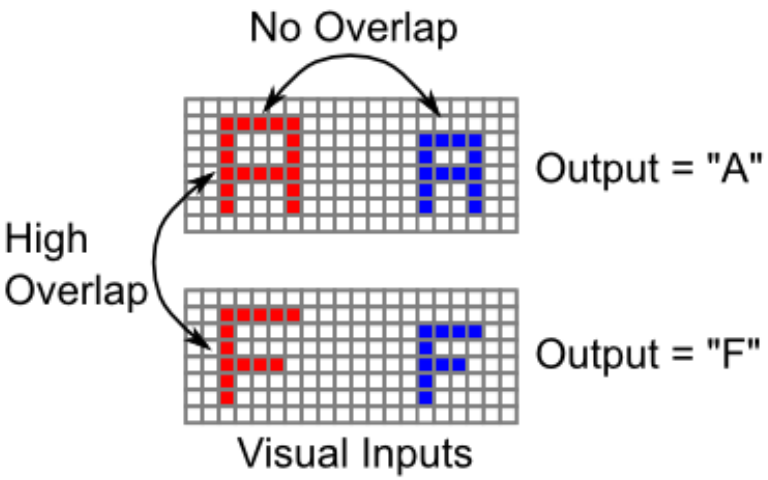



Towards invariant object recognition

If an object does not appear in the inputs with the same size and at the same location, the overlap between the « training pattern » and the object being recognized can be low. Thus, we want invariant object recognition.



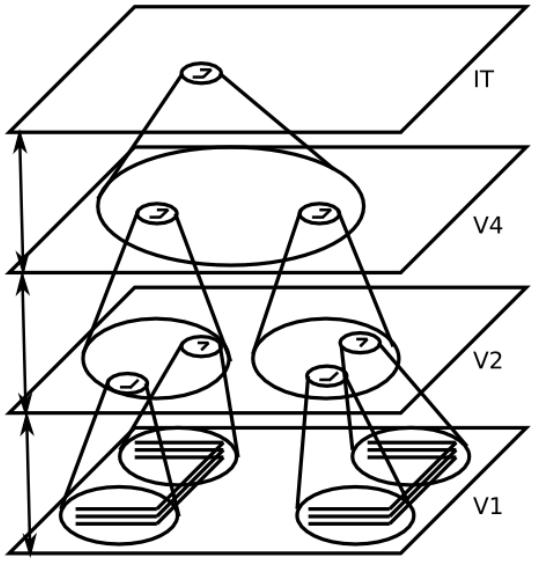
Location invariant object recognition



If an object does not appear in the inputs with the same size and at the same location, the overlap between the « training pattern » and the object being recognized can be low.

Hierarchical feature extraction:

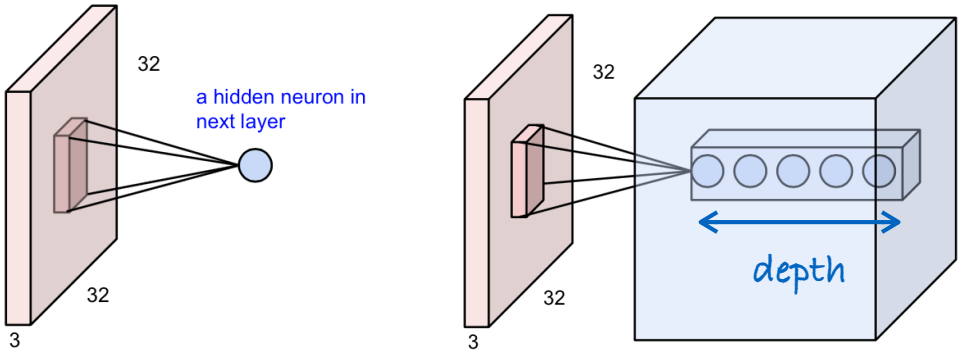
Multiple levels of processing incrementally allows the system to appropriately bind together features and their relationships, while also gradually building up overall spatial invariance.



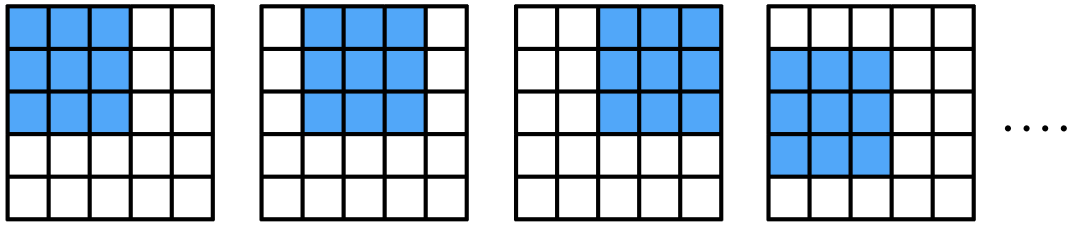
Convolutional Neural Network ingredients

1. Capability of spatial processing by means of convolutions
2. Use of multiple convolution filters to extract diverse features from the images
3. Hierarchical feature detection by connecting the output of a layer of filters to a subsequent layer of filters computing features of features
4. Use of new activation functions to deal with the problem of "vanishing gradients"
5. Use of down-sampling layers to summarize statistics of features in lower layers
6. Use of tricks to deal with the problem of overfitting (e.g., dropout)

1 - Spatial processing (convolutions) & weight sharing

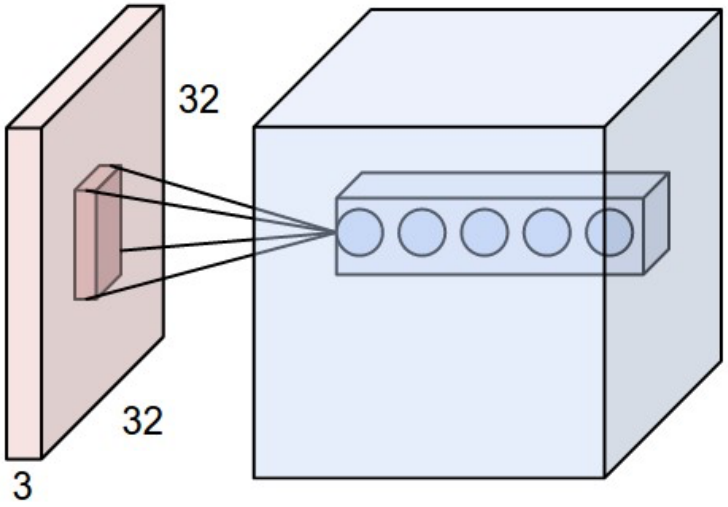


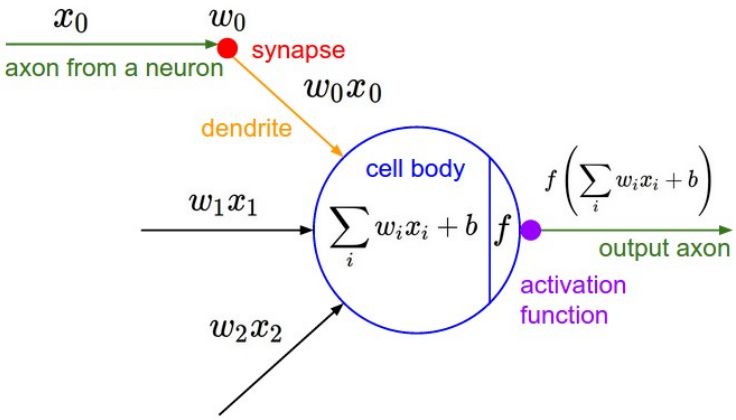
Each hidden neuron process a region of the RGB 32x32 pixel image; multiple neurons process the same region differently



replicate each column of hidden neurons across space; use the same weights in each « depth slice »

Convolution computation





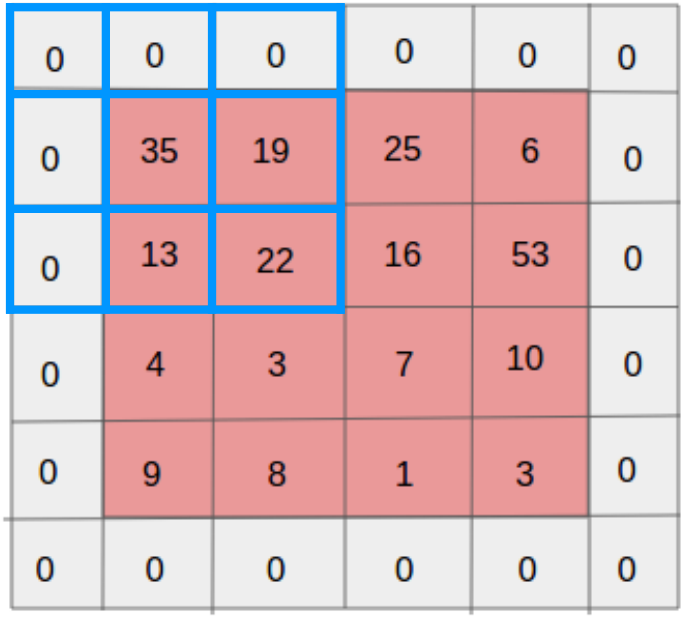
The neurons from the Neural Networks chapter remain unchanged: they still compute a dot product of their weights with the input followed by a non-linearity, but their connectivity is now restricted to be local spatially.

Convolution kernel sizes

- Different kernel sizes (3x3, 5x5, 7x7, etc) allows the identification of features at different scales
- Researchers have found that multiple layers of 3x3 kernels can implement other kernel sizes.
- Some architectures (e.g., inception) use different kernel sizes in parallel at each layer.

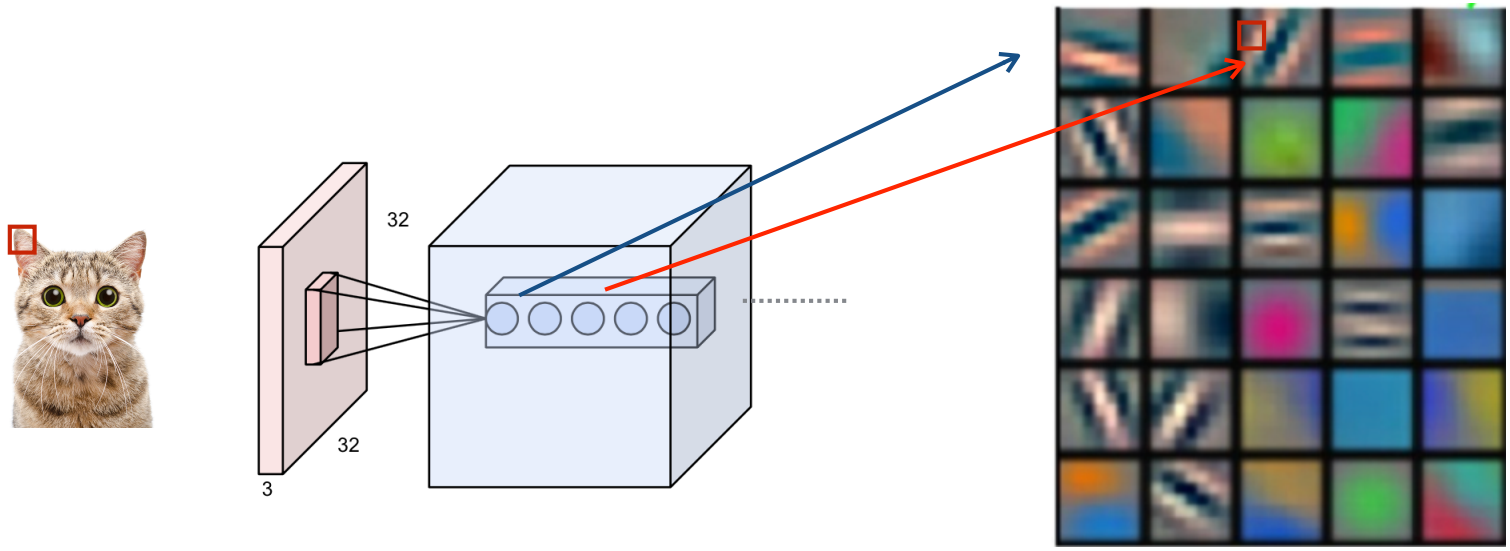
Zero-padding

- Zero-padding refers to the process of symmetrically adding zeroes to the input matrix. It is mostly used when the dimensions of the input need to be preserved in the output, by allowing the computing of an output for the pixels on the border of the input image.



http://cs231n.github.io/convolutional-networks/

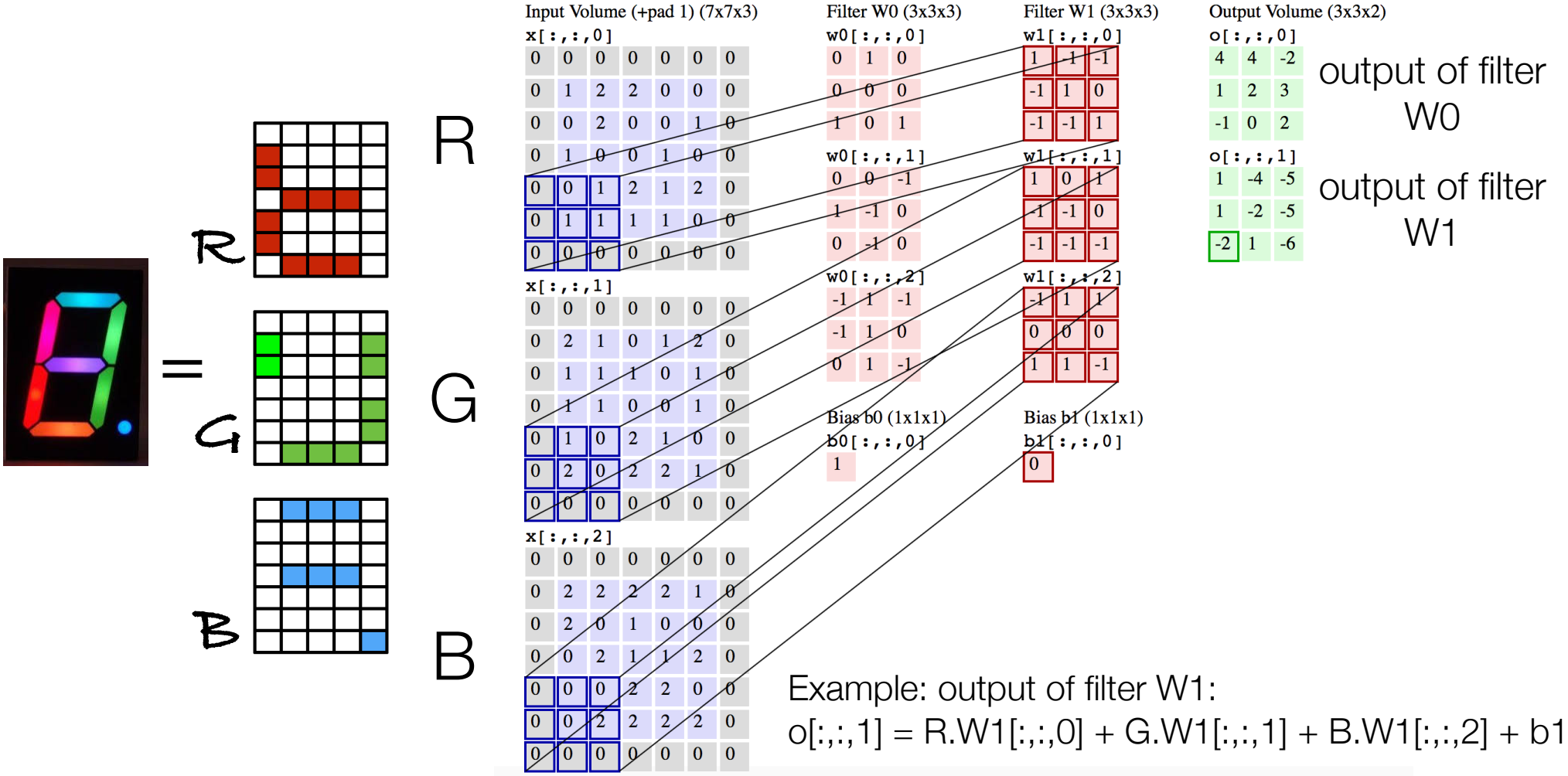
2 - Multiple convolutions with different kernels



activation maps resulting of the application of the learned « filters » (weights)

- Multiple convolutions look for detecting multiple features to recognize the object in the image being processed: e.g., are there eyes ? is there a nose ? what about whiskers ?
- Each kernel generates a new image from the input one. E.g., the output of a convolutional layer with 30 kernels is composed of 30 images!

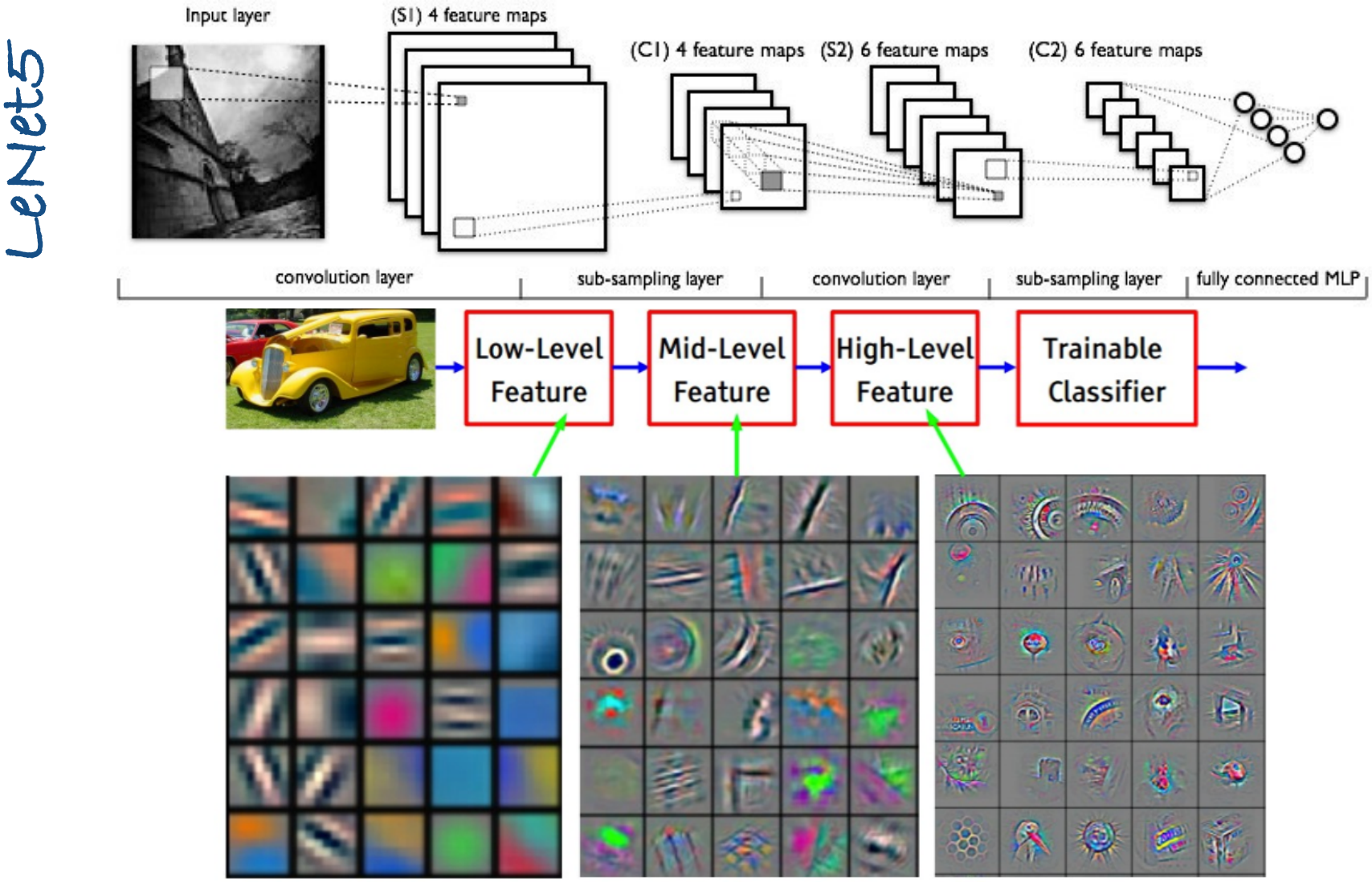
Dealing with color images



Example: output of filter W1:
o[:,:,1] = R.W1[:,:,0] + G.W1[:,:,1] + B.W1[:,:,2] + b1

When processing multidimensional images (e.g., RGB or hyperspectral), each neuron adds up the outputs of the filters applied on each channel.

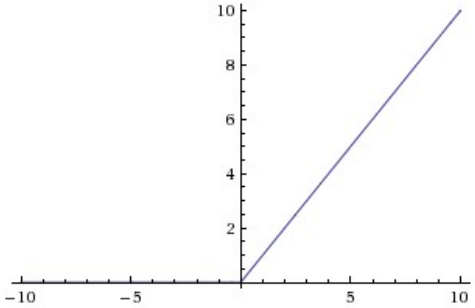
3 - Hierarchical feature detection



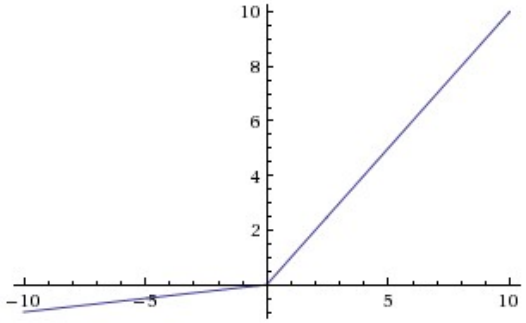
From Fei-Fei & Karpathy slides; deeplearning.com/tutorial

4 - New activation functions

- Saturating activation functions like the sigmoid contribute to the problem of "vanishing gradients" in a network. As they approach 0 or 1, the derivative approaches zero too and the error signal used for learning.



ReLU: Rectified Learning Unit

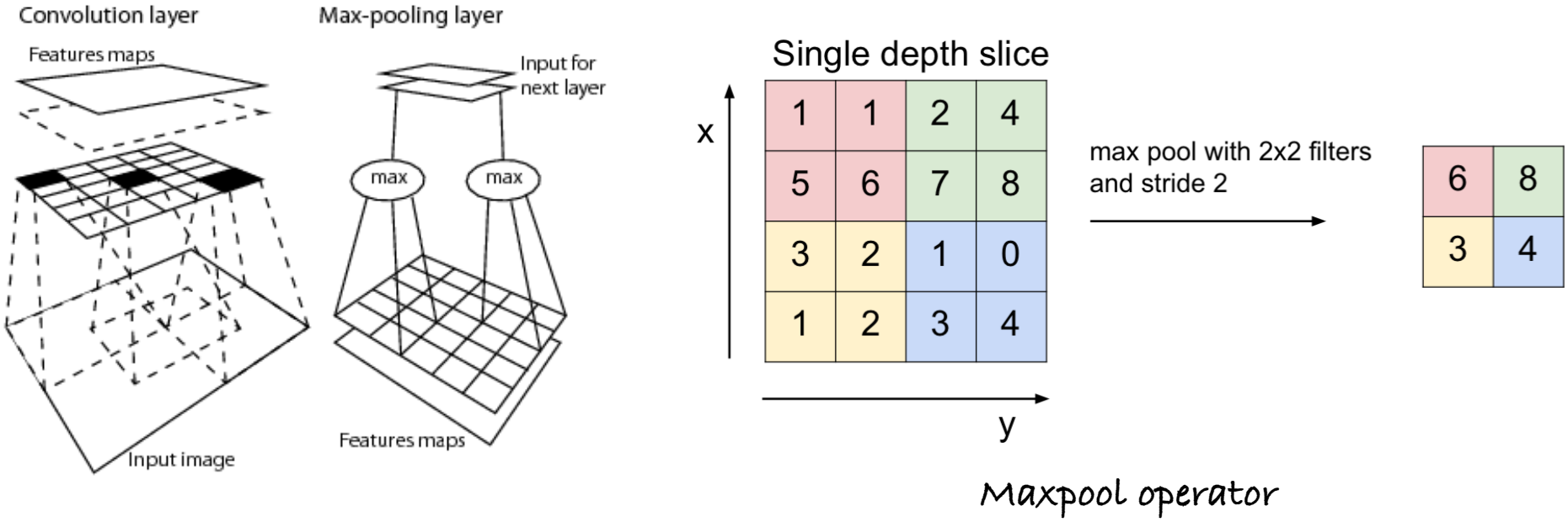


Leaky ReLU function

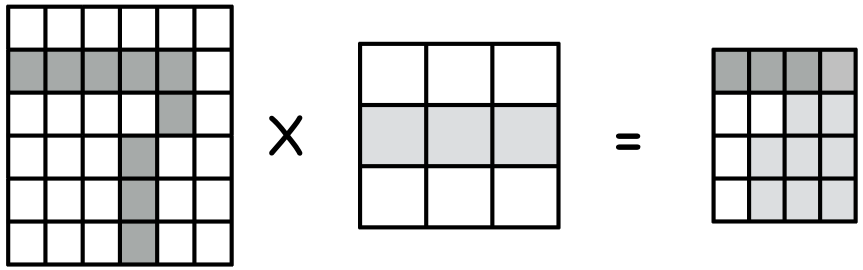
- A softmax function is often used in the output layer to compute P(target/inputs, weights, bias):
- $f(x_i) = e^{xi} / \sum e^{xj}$, j = 1…K
- The sum of $f(x_i)$ equals 1, thus the outputs represent a categorical probability distribution.

5- Down-sampling layers

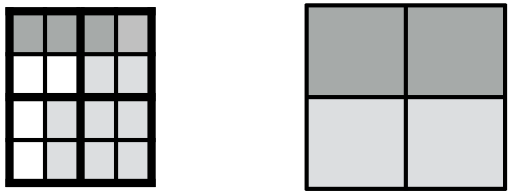
- Maxpool after a convolution layer eliminates non maximal values: it is a form of non-linear down-sampling that reduces computation for upper layers and provides a « summary » of statistics of features in lower layers. The resulting images are smaller than those having been processed in previous layers. /* average pooling is an option too */



Maxpooling example



convolution -> pattern detection over all the input image

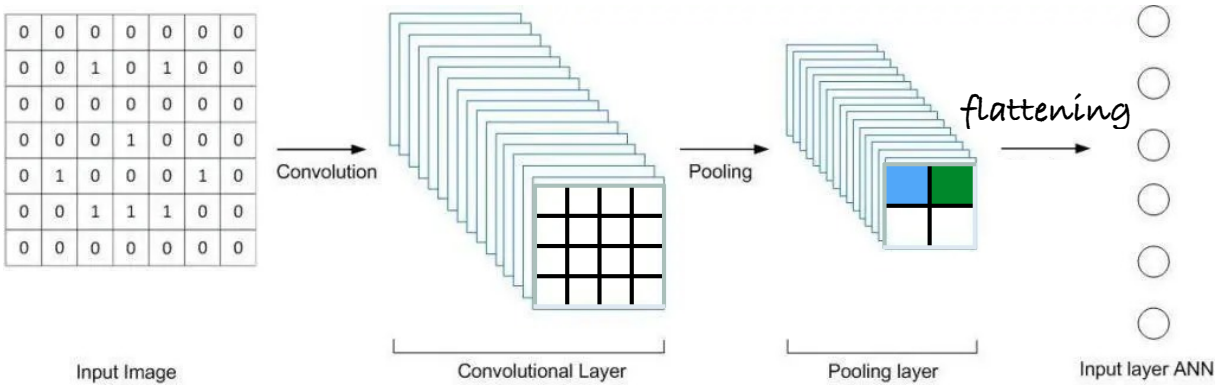


maxpooling -> summary statistics of pattern detection

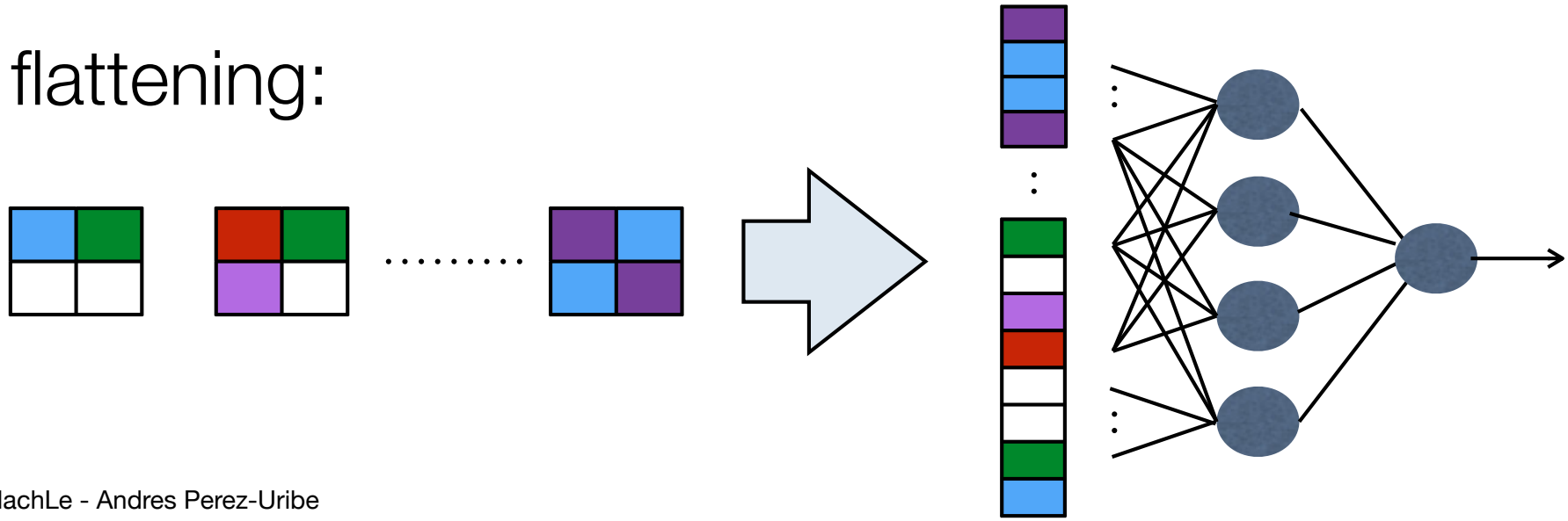
"a horizontal line is present in the upper part of the image"

Flattening images before the output MLP

- The output of the convolution and sampling layers is composed of images that need to be flattened before being processed by an output MLP.

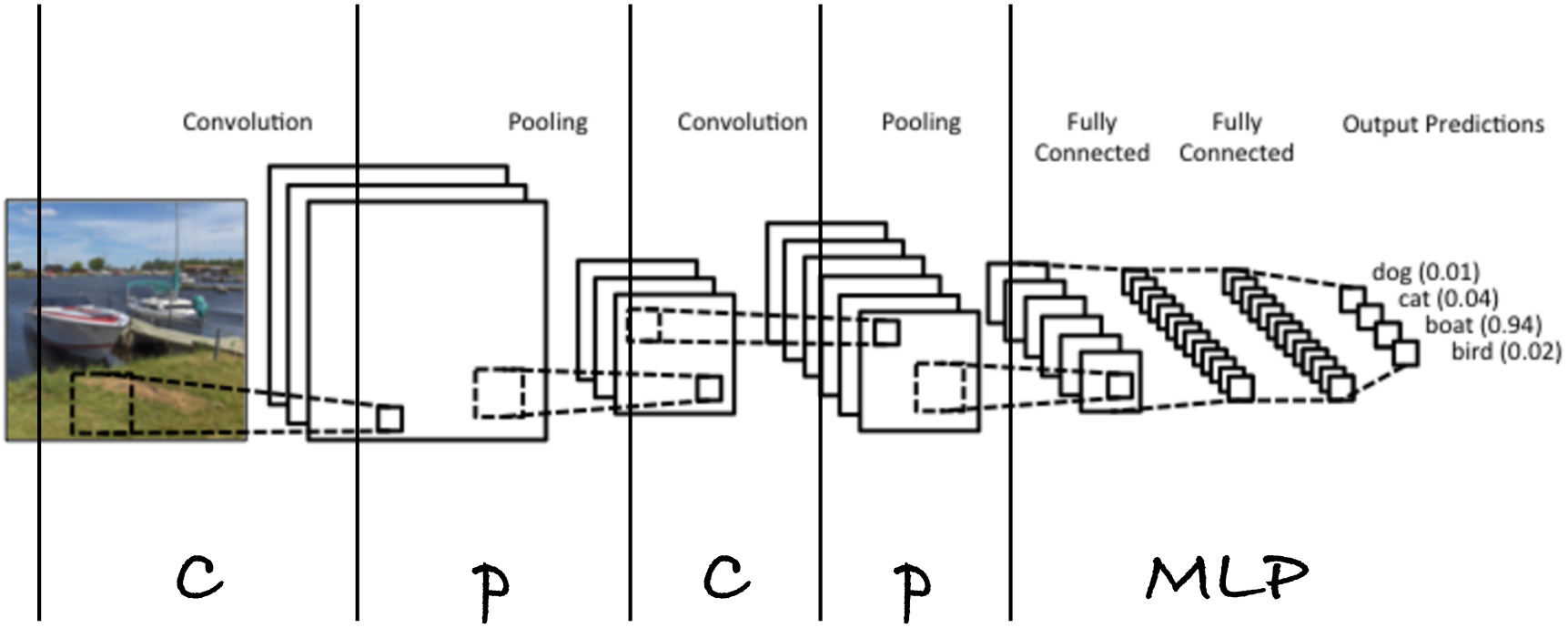


flattening:



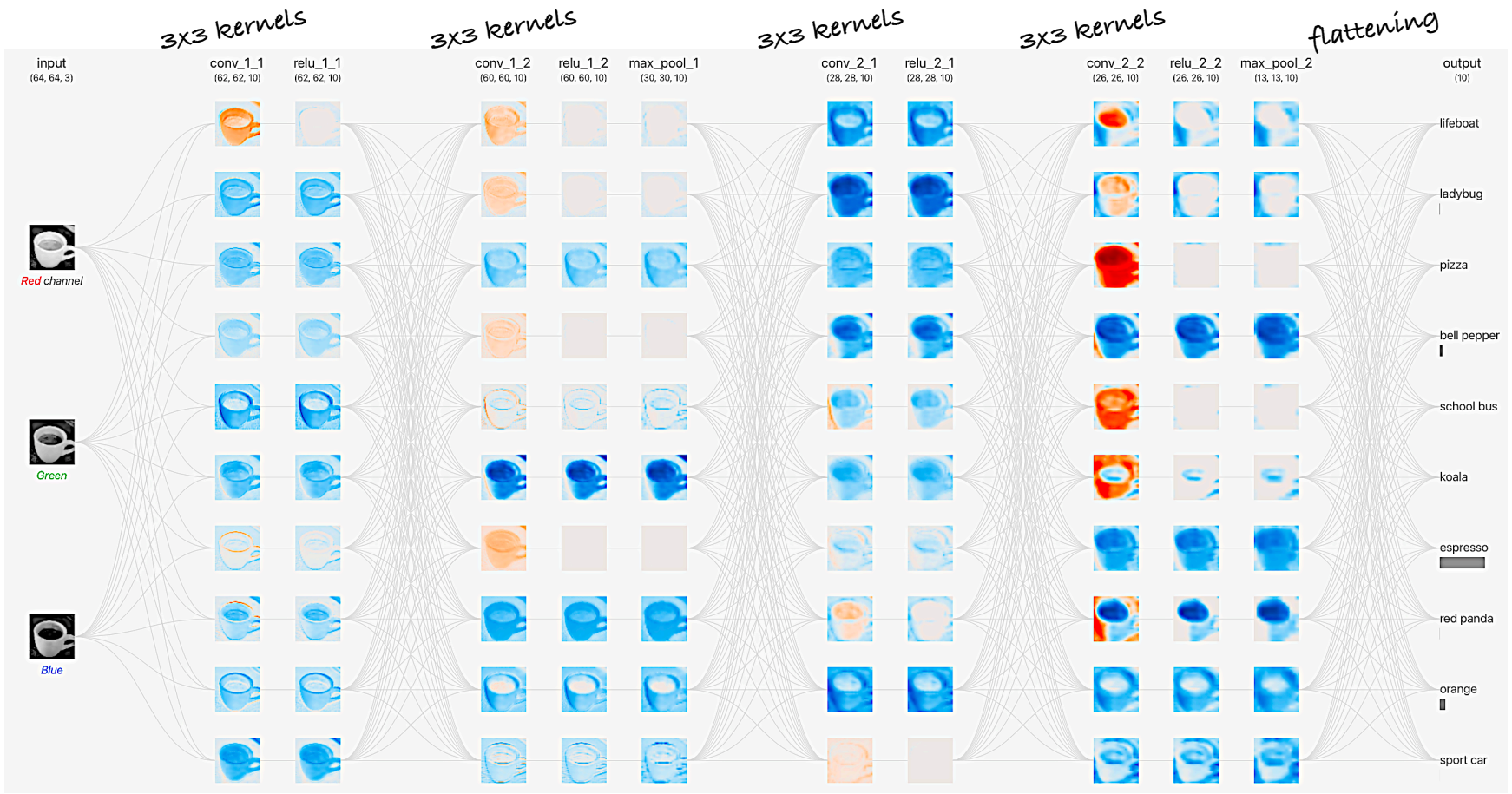
Resulting architecture

- The whole network is trained using Backpropagation



- softmax output for classification
- tanh output for regression

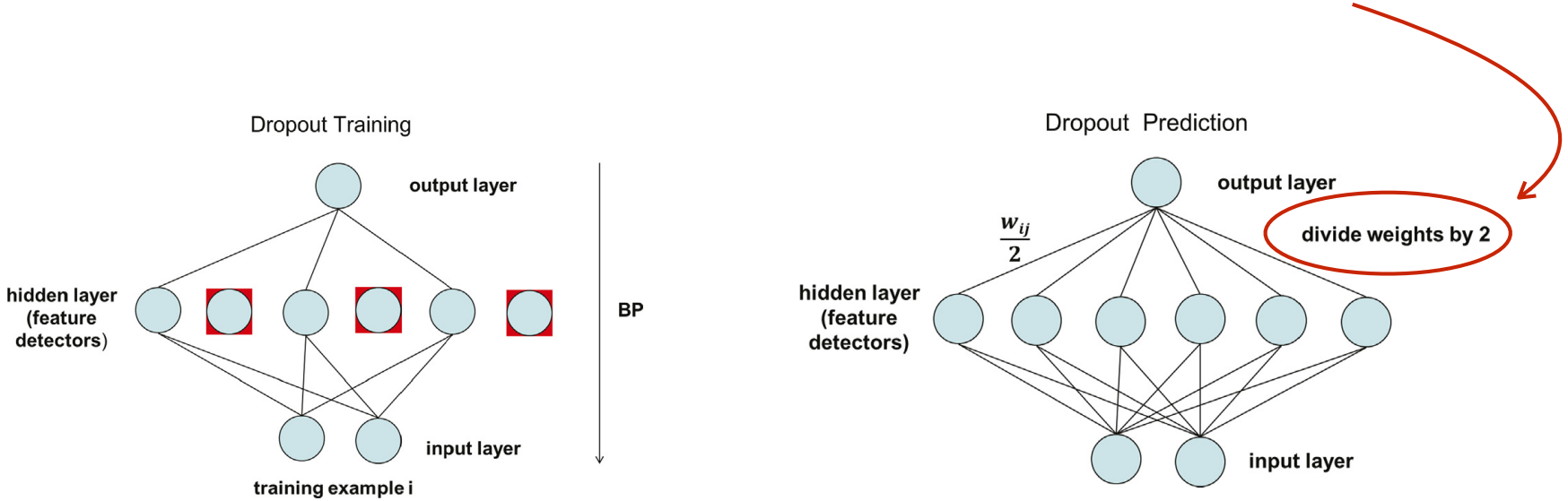
A CNN example



https://poloclub.github.io/cnn-explainer/

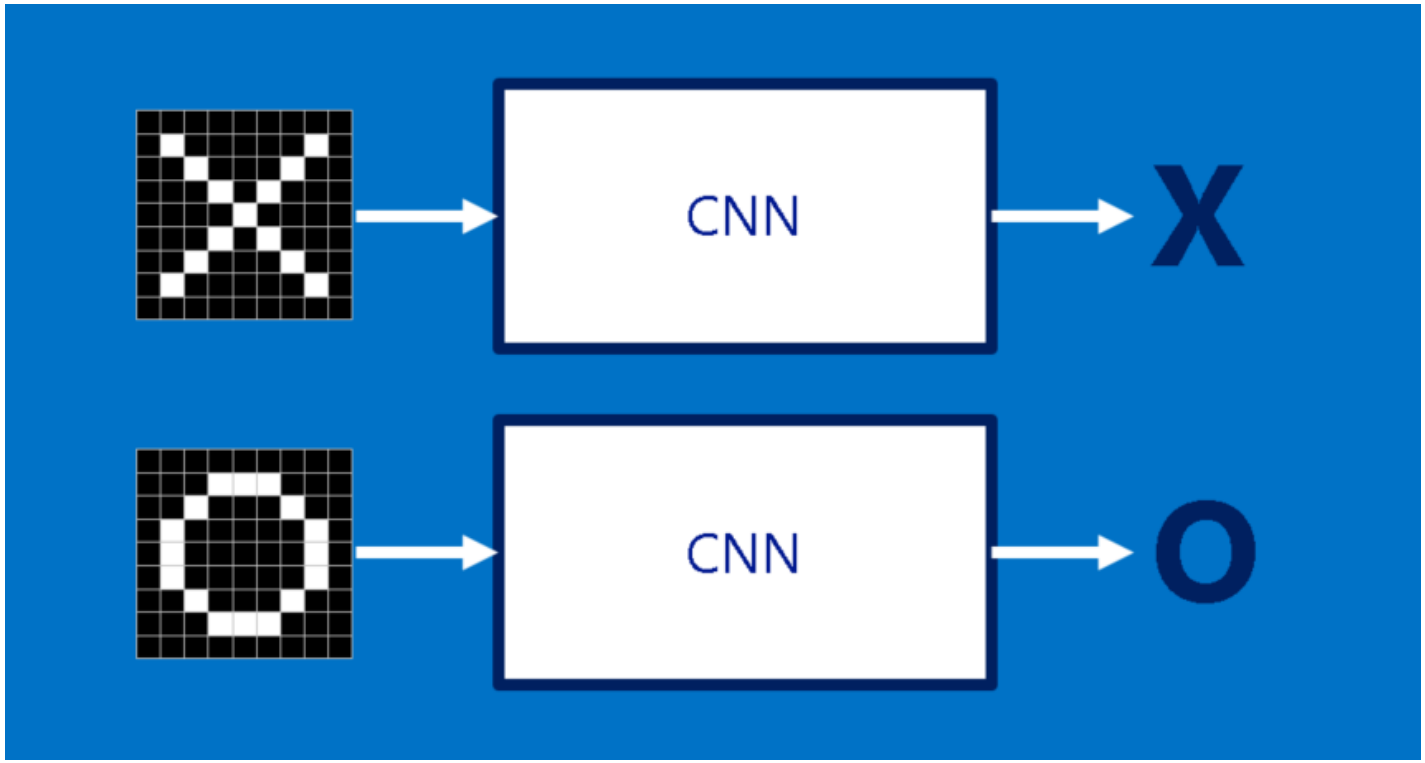
6 - Dropout to avoid overfitting

- Dropout (Hinton et al, 2012) is a technique for training neural networks by randomly dropping units during training to prevent « overfitting »
- If we dropout each feature detector with a probability of 0.5:



- In classical ANNs: training set size >> number of weights
- In CNNs: training set size is ≲ number of weights

How do CNN's work ? (by Brandon Rohrer)



https://brohrer.github.io/how_convolutional_neural_networks_work.html

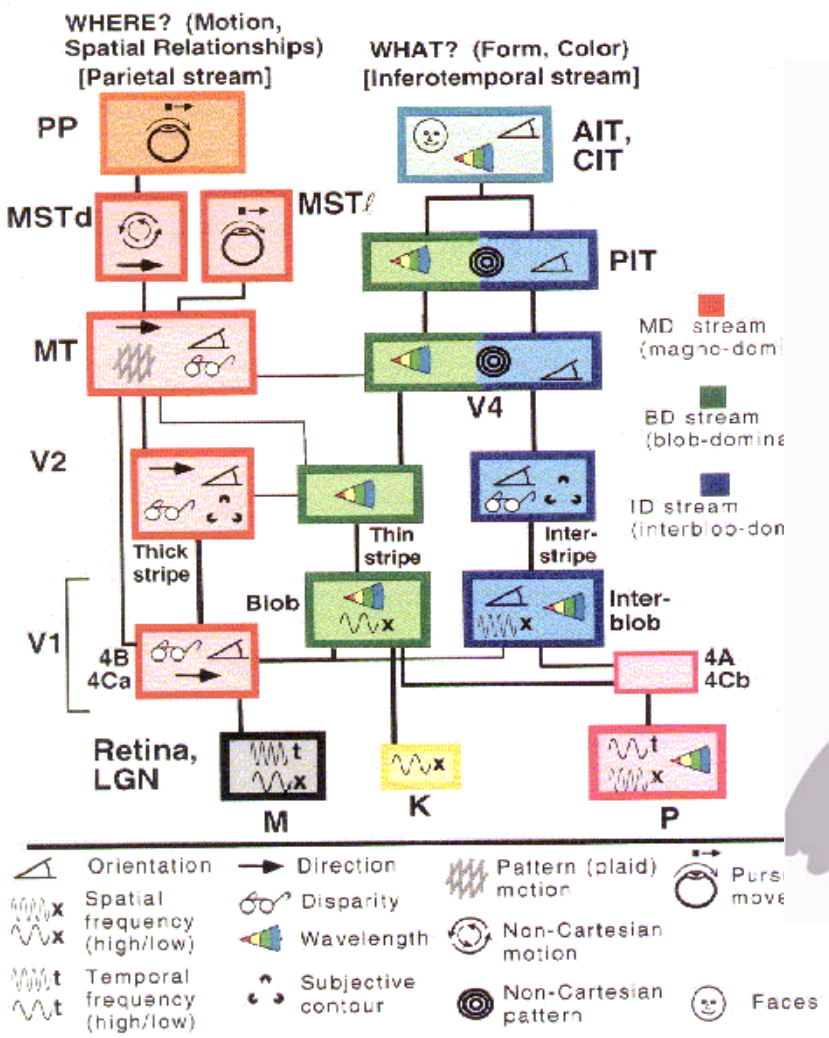
https://medium.com/@CharlesCrouspeyre/comment-les-réseaux-de-neurones-à-convolution-fonctionnent-b288519dbcf8 (french translation)

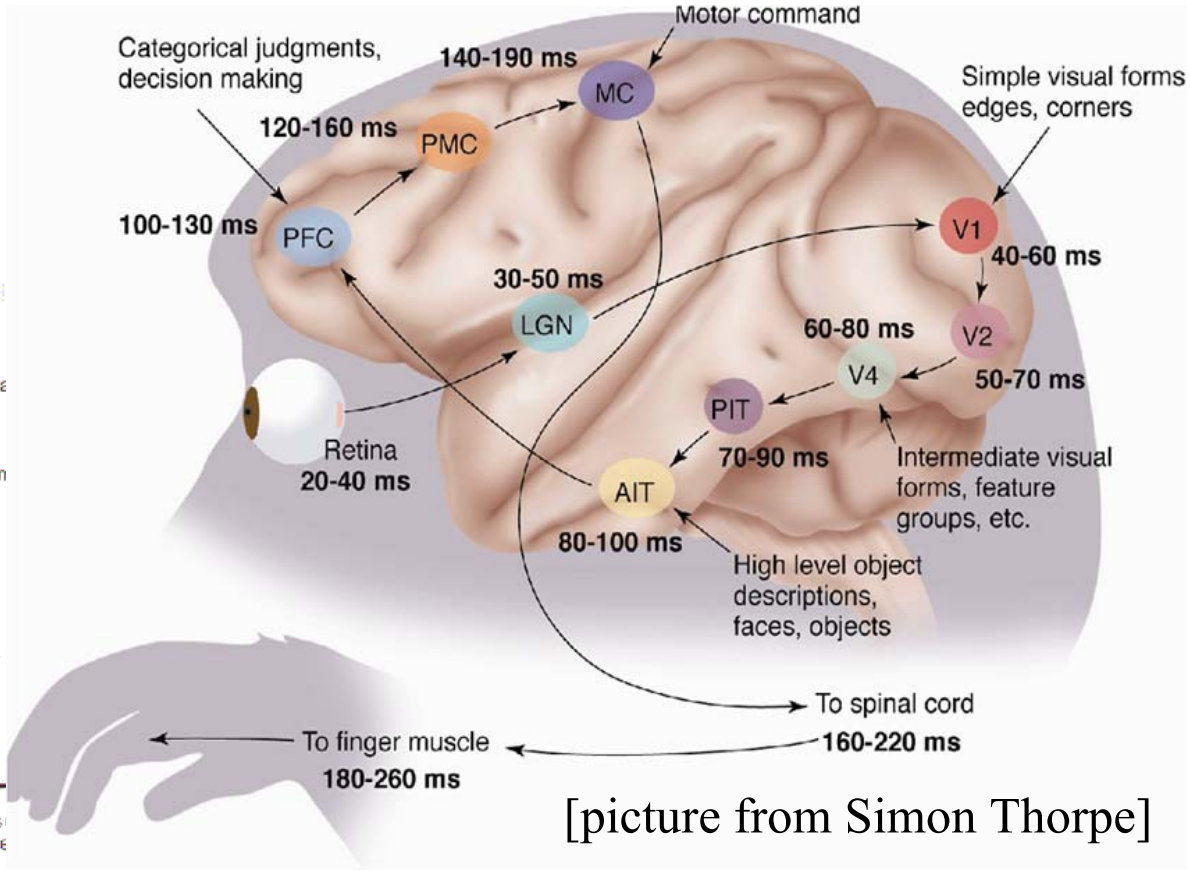
CNNs making the headlines

- AI has made the front page of the NY times three times in recent times: 1) when Deep Blue beat Kasparov, 2) when Watson beat the best human Jeopardy players, and 3) when students using Deep Learning algorithms won the Kaggle's « Merck molecular activity challenge » in Nov. 2012
- Deep Learning in your browser:

  http://cs.stanford.edu/people/karpathy/convnetjs/index.html

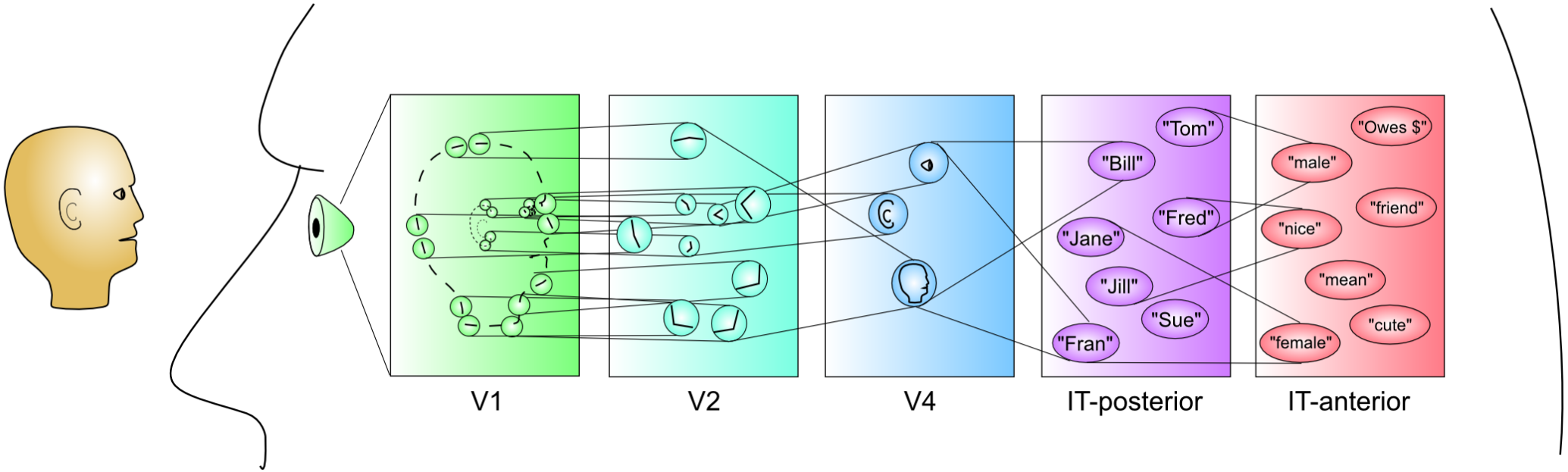
Visual processing in the brain





[Gallant & Van Essen]

Visual pathway



V1 has elementary feature detectors (oriented edges). Next, these are combined into junctions of lines in V2, followed by more complex visual features in V4. Individual faces are recognized at the next level in IT (even here multiple face units are active in graded proportion to how similar people look). Finally, at the highest level are important functional "semantic" categories that serve as a good basis for actions that one might take

Learnable feature hierarchies

- Image recognition:

  pixel -> edge -> texton -> motif -> part -> object

- Text processing:

  character -> word -> word group -> clause -> sentence -> story

- Speech:

  sample -> spectral band -> sound -> phoneme -> word -> …

"Anything humans can do in 0.1 sec, the right big 10-layer network can do too »
in Large Scale Deep Learning by Jeff Dean (Google)

Practical work

- Objective: experiment with large and deep networks in the task of digit recognition using the MNIST database using Keras.

- Keras is a high-level neural networks library, written in Python and capable of running on top of either TensorFlow or Theano. Its primary author and maintainer is François Chollet.

- Keras is a leading deep learning framework for Python, with over 50,000 users and over 200 open-source contributors. Keras is in use at a considerable number of startups, research labs (including CERN, Microsoft Research and OpenAI), and large companies such as Netflix, Yelp, Square, Google, etc.

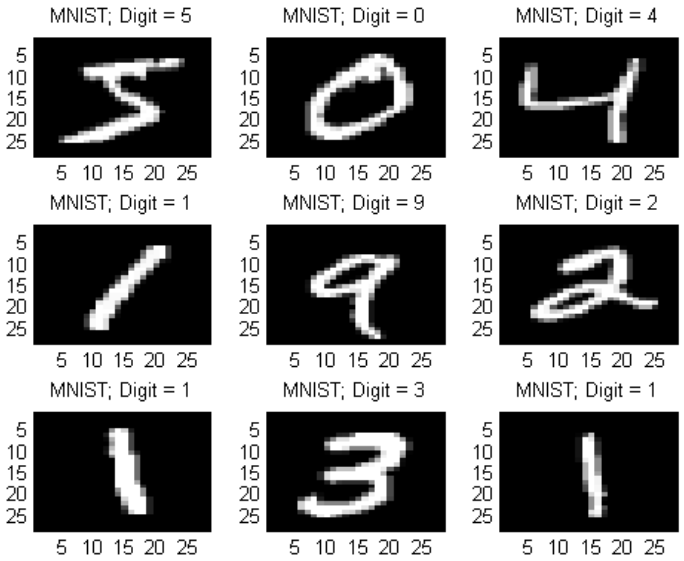
Practical work (2)

- TensorFlow is an open source software library for numerical computation using data flow graphs.
- Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them.
- This flexible architecture lets you deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code.
- TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization.

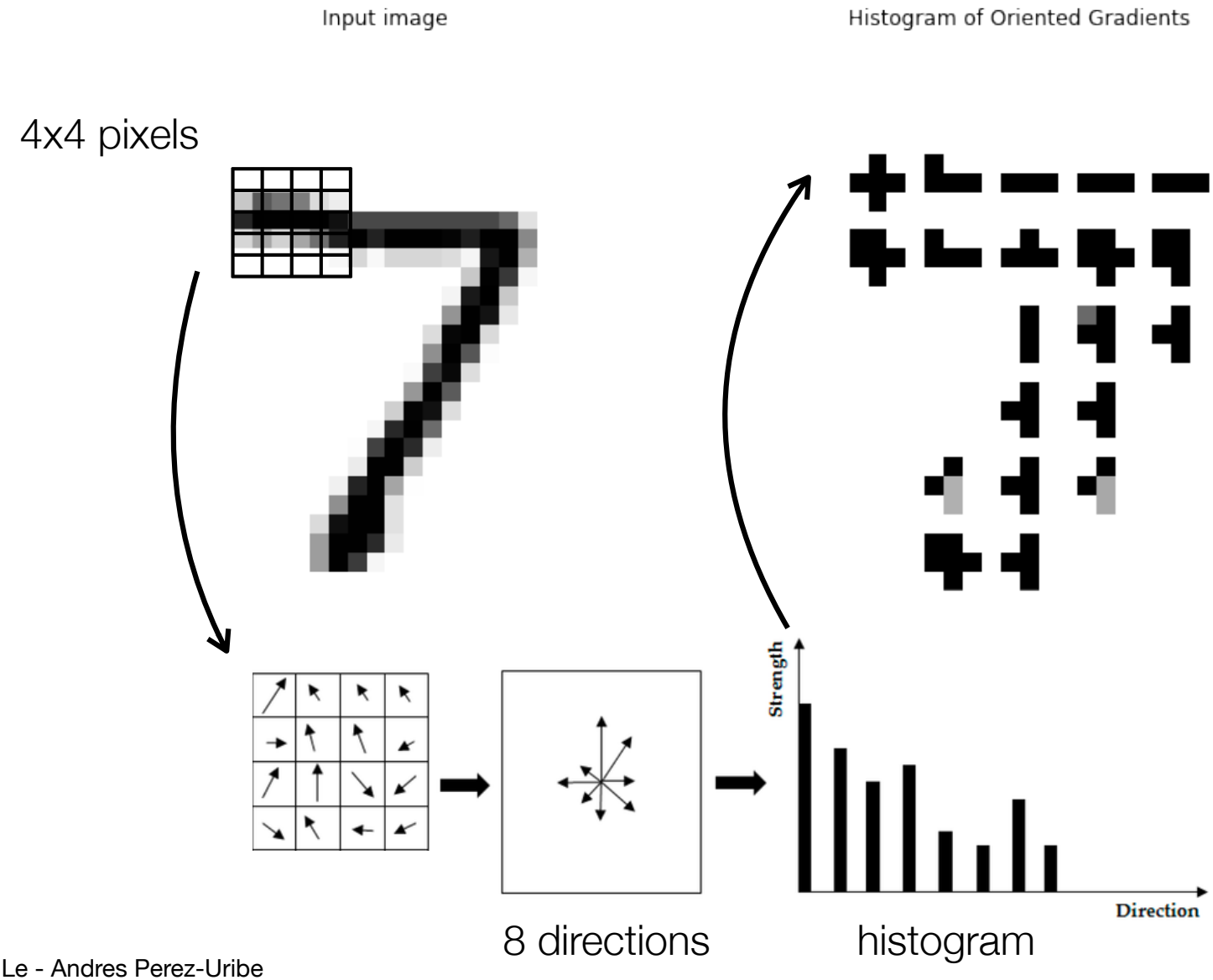
Digit recognition

- From raw data to digit recognition
- Classification based on features
  - Histogram of Gradients (HOG)
- Convolutional Neural Network

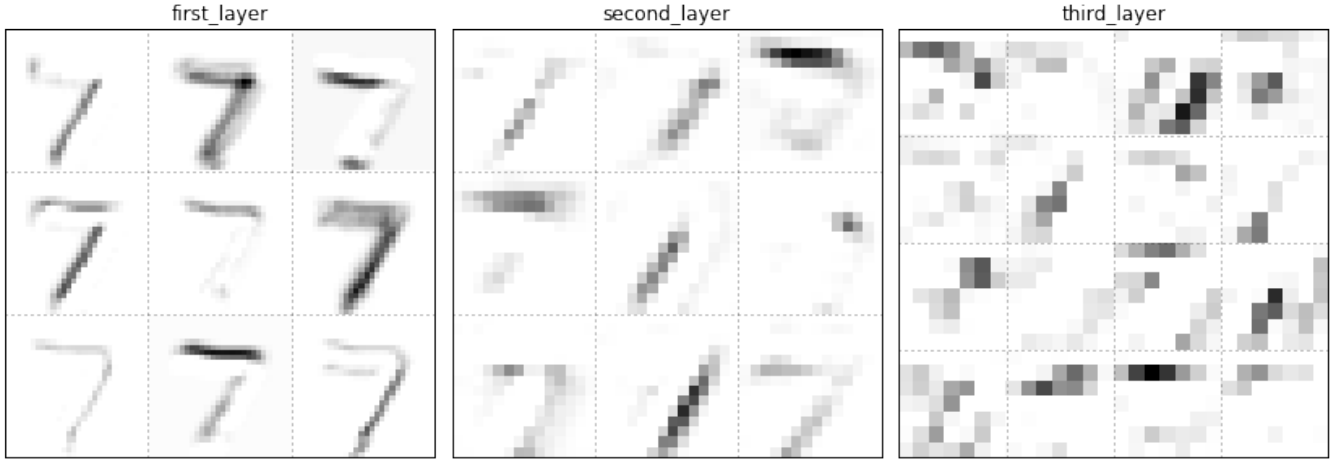
28x28 pixel images

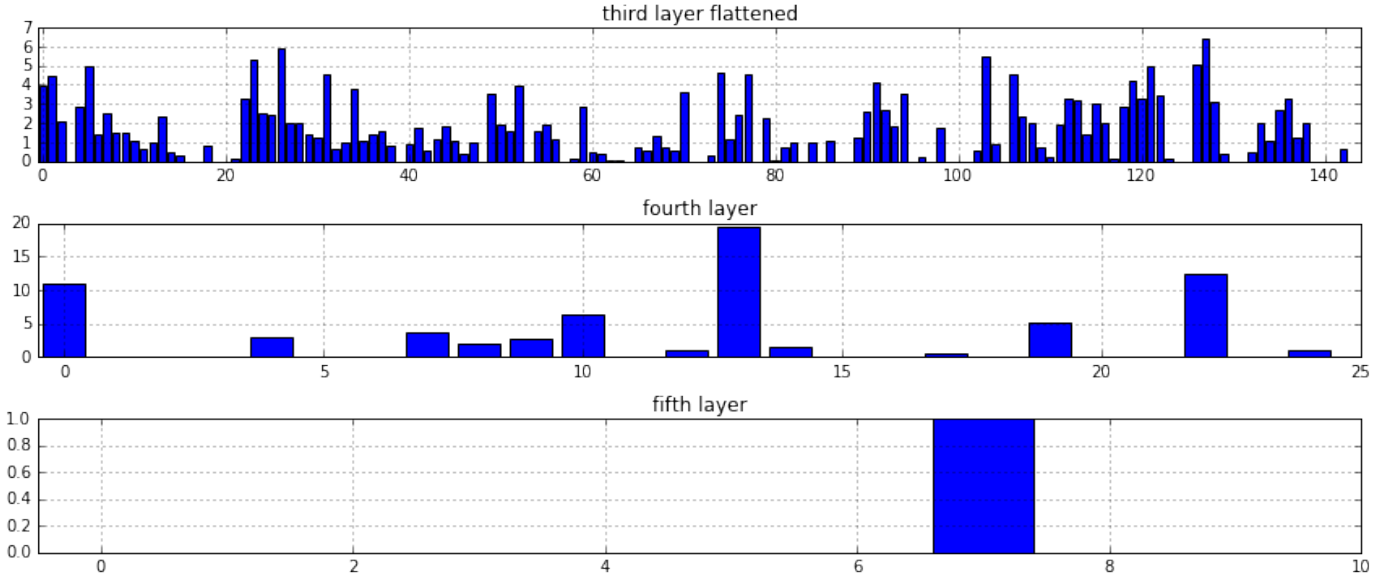


HOG feature-based digit recognition

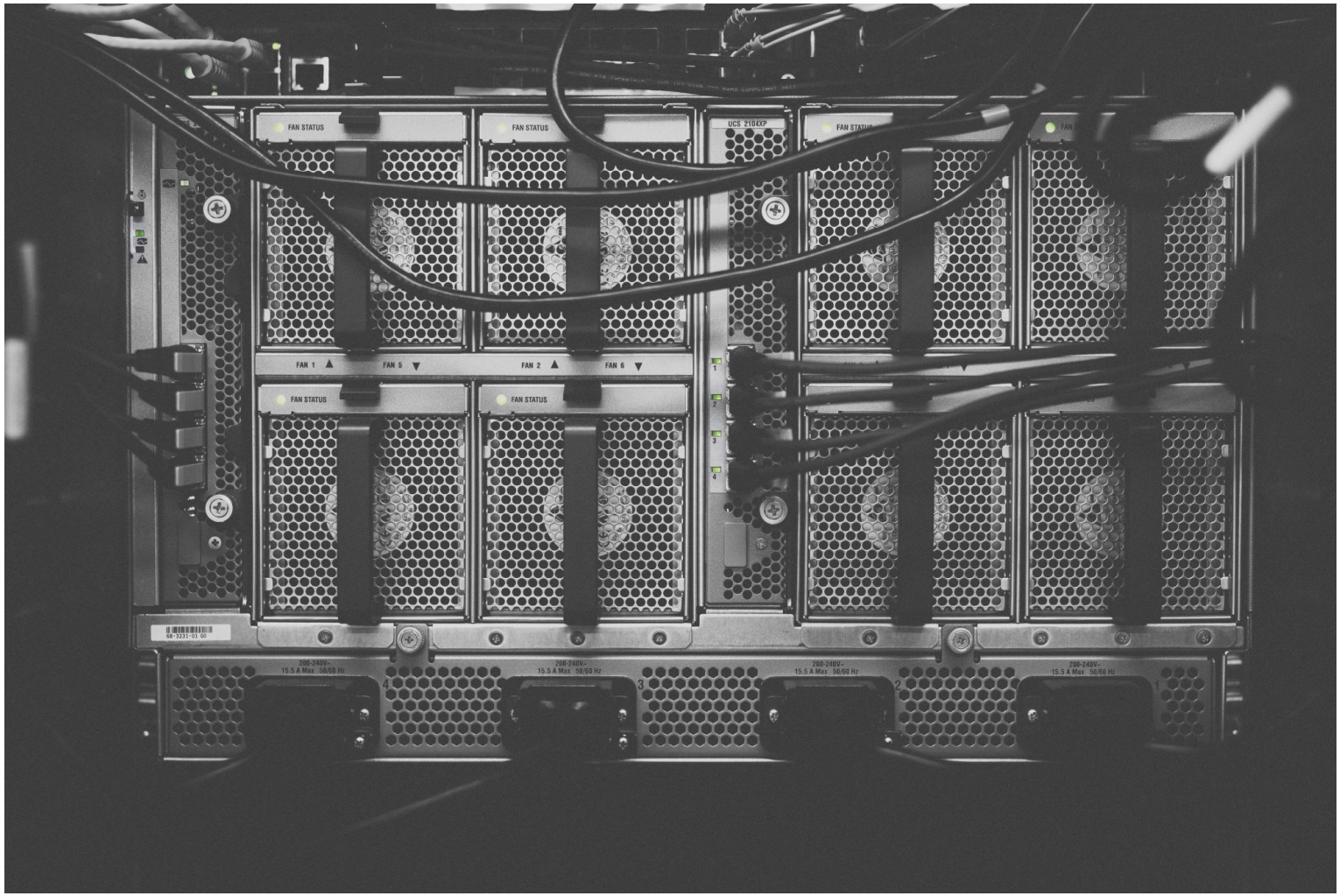


CNN-based digit recognition





GPU acceleration



Google colab

- **Google Colab**oratory is a free, in-the-browser, collaborative programming environment that provides an interactive and easy to use platform for deep learning researcher.
- Google allows to use CPUs, GPUs and TPUs.
- TPUs or Tensor Processing Units are ASICs developed by Google, tailored to run Tensorflow processes. e.g., They were used to train AlphaGo.
- Introduction and example notebooks:

  https://medium.com/dair-ai/primer-for-learning-google-colab-bb4cabca5dd6