H2I Workshop – Image Classification using Deep Learning

H2I Workshop 16/09/2021

unibz
Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

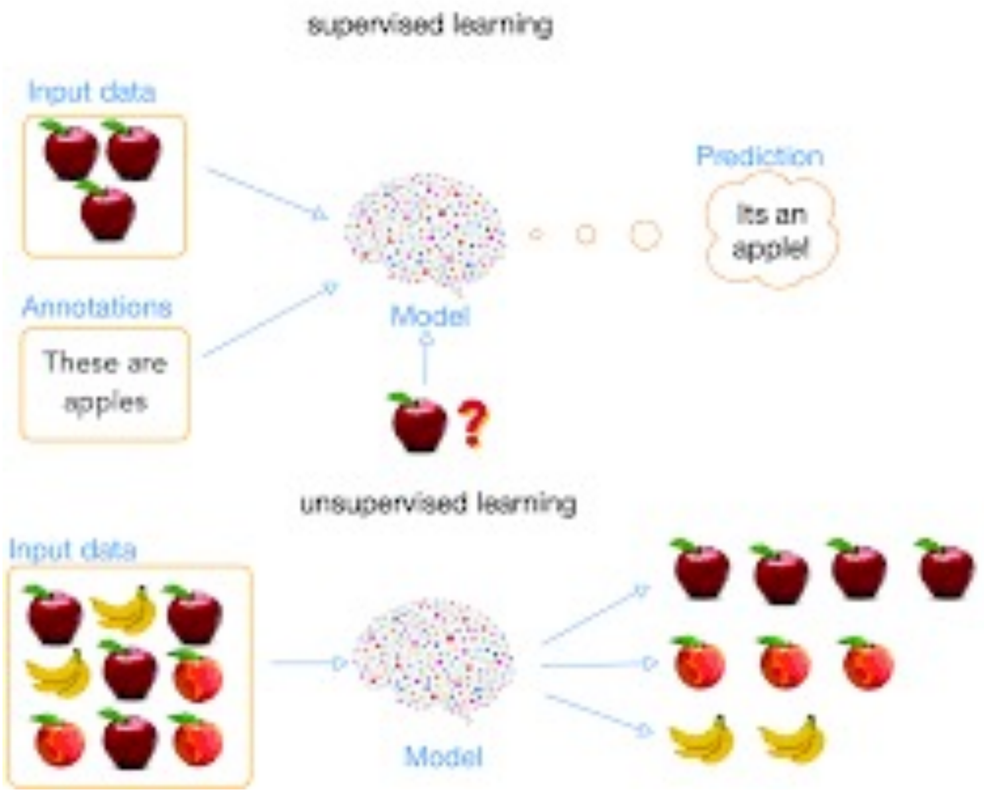
efre · fesr
Südtirol · Alto Adige
Europäischer Fonds für regionale Entwicklung
Fondo europeo di sviluppo regionale
EUROPEAN UNION
AUTONOME PROVINZ BOZEN SÜDTIROL
PROVINCIA AUTONOMA DI BOLZANO ALTO ADIGE

Outline

- (Very) Brief introduction to ML/DL
- Convolutional Neural Networks (CNNs)
- CNNs Architectures
- Computer Vision Applications
- Segmentation Framework for Hyperspectral Images Classification

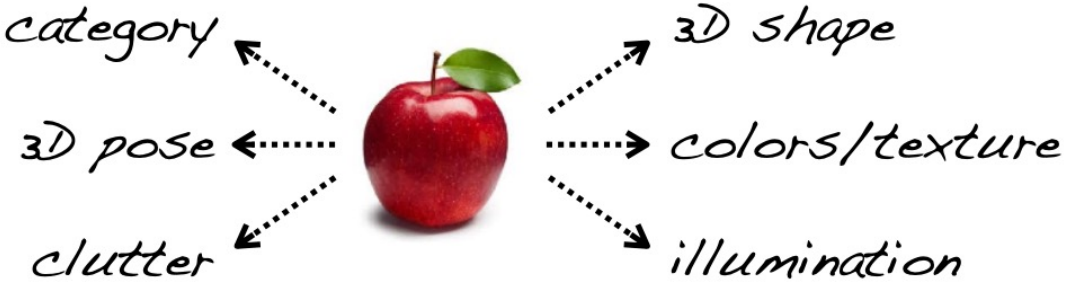
What is Machine Learning?

- Machine learning is the science (and art) of programming computers such that they can *learn from data*
- There exist different types of Machine Learning algorithms
  - Supervised
  - Unsupervised
  - Reinforcement learning



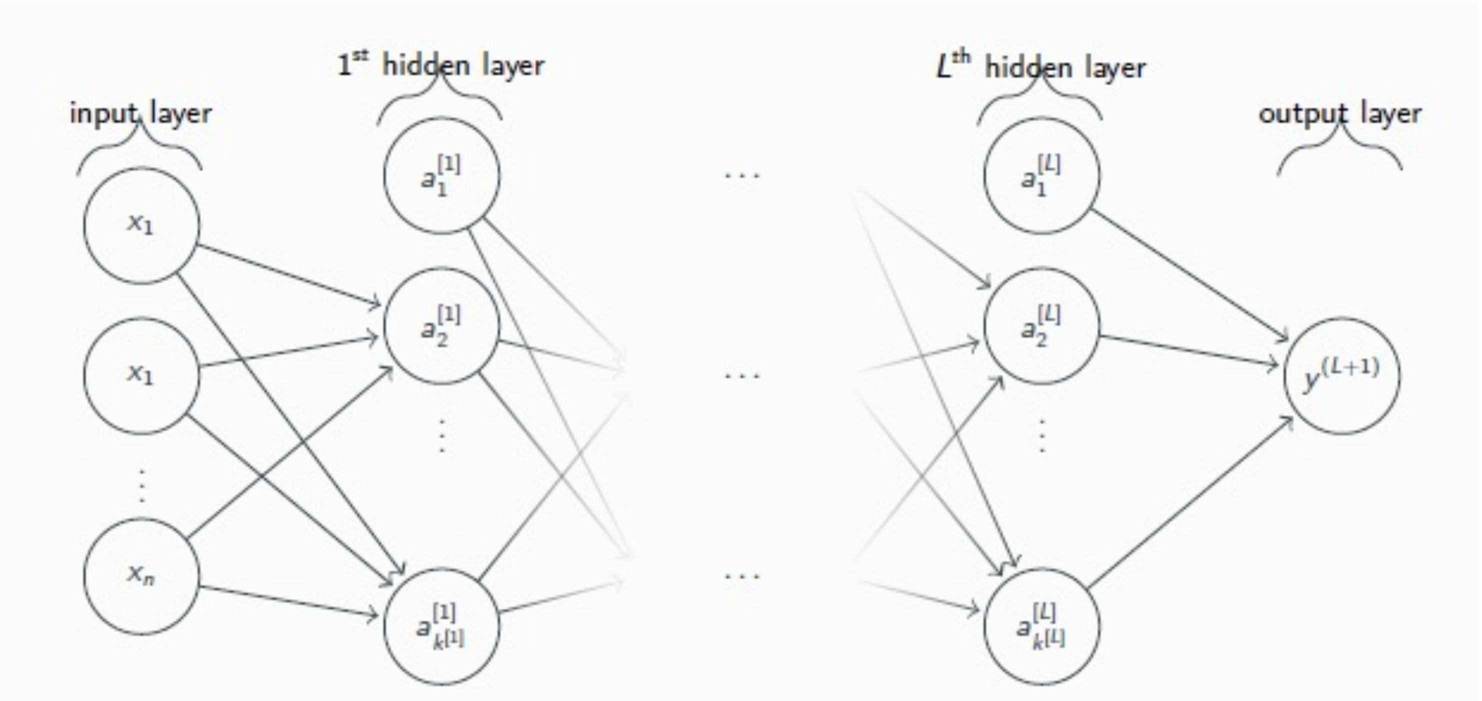
What is Deep Learning?

- Deep Learning is an ‘advanced’ form of Machine Learning
- It solves the central problem of **representation learning**
- It introduces hierarchical representations (from simple to complex, from low-level features to high-level features).



Feed Forward Neural Network

- Input neurons -> input data.
- Hidden neurons -> features extraction
- Output neurons -> categories

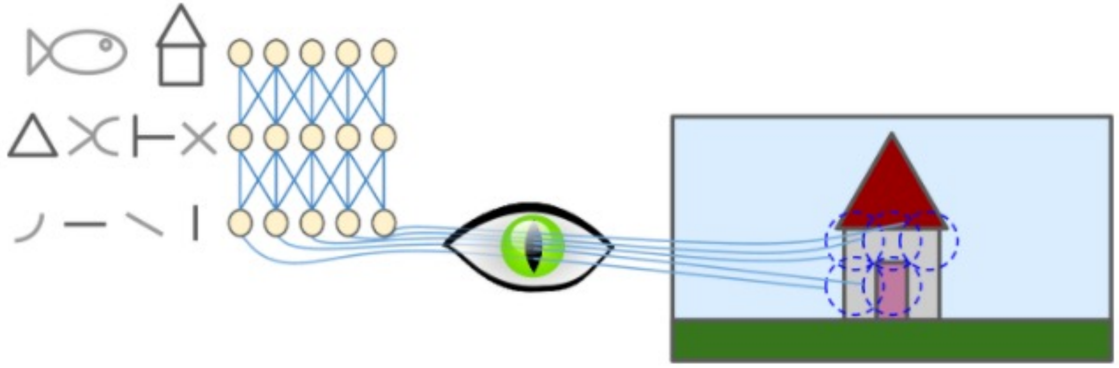


Outline

- (Very) Brief introduction to ML/DL
- **Convolutional Neural Networks (CNNs)**
- CNNs Architectures
- Computer Vision Applications
- Segmentation Framework for Hyperspectral Images Classification

Origins

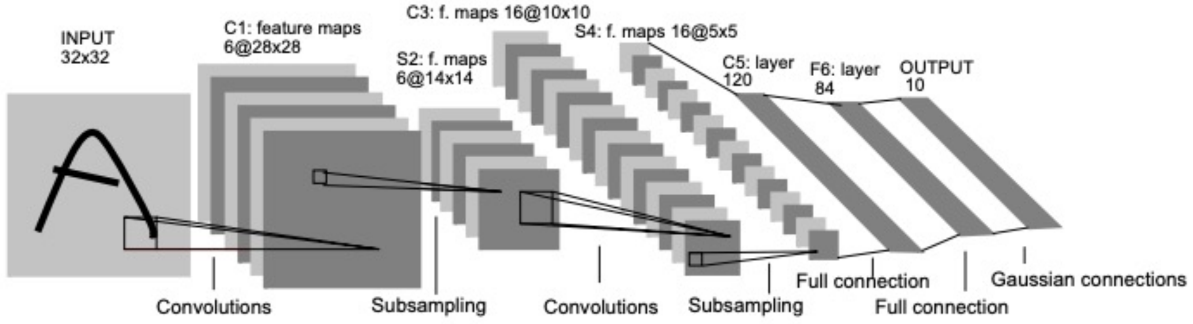
- Convolutional neural networks (CNNs) emerged from the study of the **brain's visual cortex**



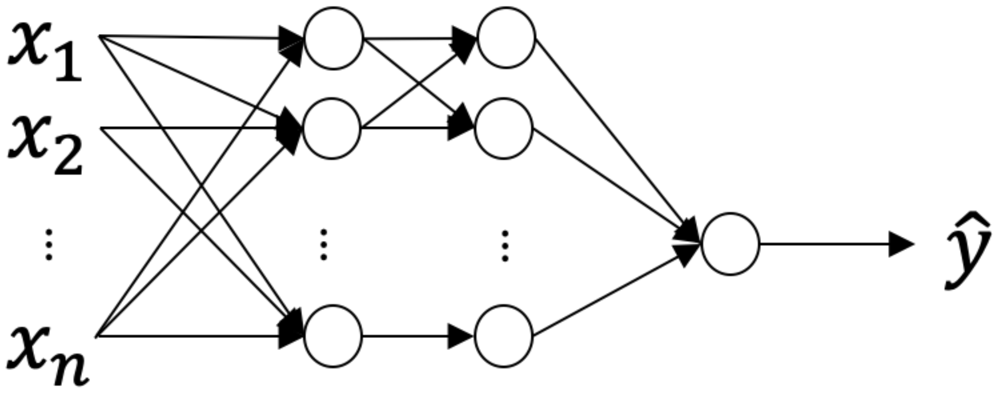
- David H. Hubel and Torsten Wiesel, recipients of the Nobel Prize in 1981, showed in a series of experiments on cats and monkeys that:
  - Many neurons in the visual cortex react only to a visual stimuli located in a limited region of the visual field (*local receptive field*)
  - Some neurons react only to images of horizontal lines, while others only to lines with different orientations
  - Some neurons react to more complex patterns, that are combinations of lower-level patterns

Origins

- The famous paper by Lecun et al. 1998 introduced:
  - The LeNet-5 architecture, used by banks to recognise handwritten check numbers.
  - New building blocks **convolutional** and **pooling layers**.
  - It introduced the concept of **partially connected layers**.

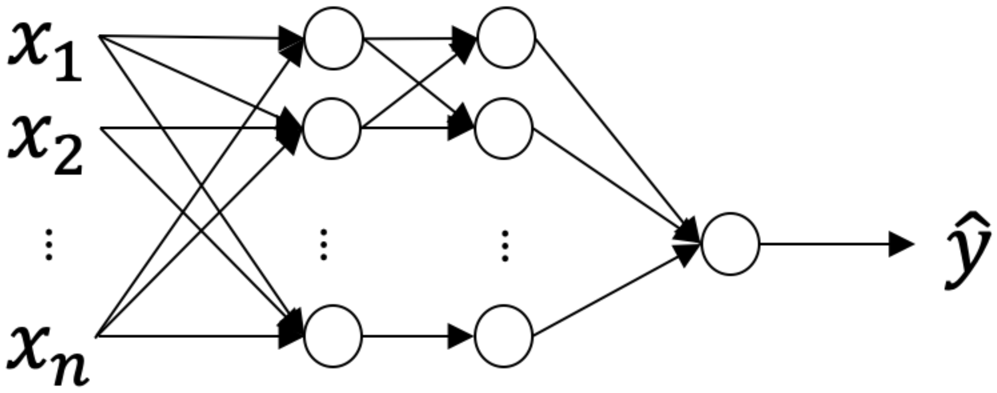


Convolutional Neural Networks - Intro



- Feed Forward Deep Neural Networks on Large Images
- What will happen if the image has a resolution of 1M pixels?

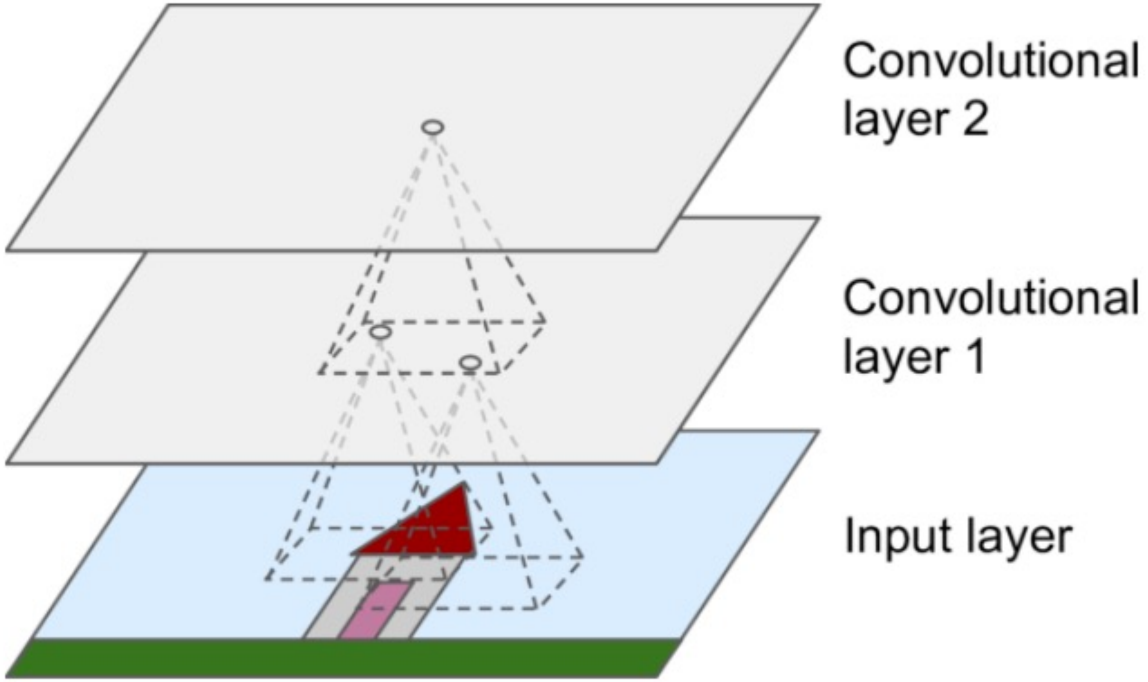
Convolutional Neural Networks - Intro



- Feed Forward Deep Neural Networks on Large Images
- What will happen if the image has a resolution of 1M pixels?
- In FFNN the input network would be of 1000 * 1000 * 3 = 3M of inputs! (and much more parameters to learn)

Convolutional Layers

- Neurons in the first convolutional layer are not connected to every single pixel in the input image.
- Each neuron in the second convolutional layer is connected only to neurons located within a small rectangle in the first layer



Convolutional Layers

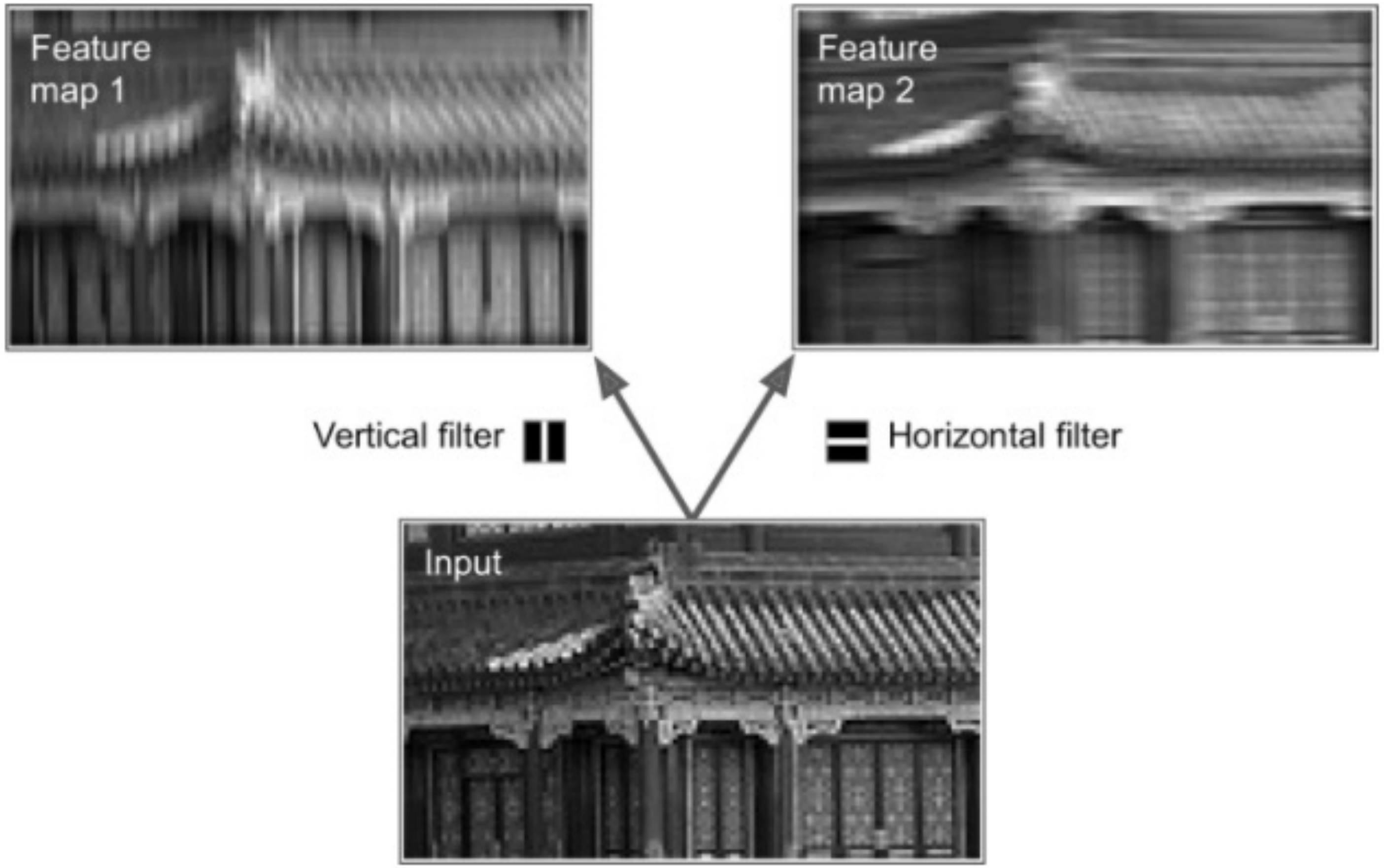
- Identify low features and then higher, more abstract, features

Convolution

- Convolution operator allows you to detect features, e.g., (vertical) edges

$$
\begin{bmatrix}
10 & 10 & 10 & 0 & 0 & 0 \\
10 & 10 & 10 & 0 & 0 & 0 \\
10 & 10 & 10 & 0 & 0 & 0 \\
10 & 10 & 10 & 0 & 0 & 0 \\
10 & 10 & 10 & 0 & 0 & 0 \\
10 & 10 & 10 & 0 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
1 & 0 & -1 \\
1 & 0 & -1 \\
1 & 0 & -1
\end{bmatrix}
=
\begin{bmatrix}
0 & 30 & 30 & 0 \\
0 & 30 & 30 & 0 \\
0 & 30 & 30 & 0 \\
0 & 30 & 30 & 0
\end{bmatrix}
$$

Filters and Feature Maps



Filters (Kernels)

- Filters to be learned (features extractors)
- Treat the filter's numbers as parameters to be learned by means of back-propagation.
  - Can learn features of filters more robustly
  - Can learn edges that are not only horizontal/vertical

Filters - Convolution

- Convolution operator:
  - Input: a input matrix of dimension n x n, and a filter of dim. f x f
  - Output: a matrix of dim: $(n - f + 1) \times (n - f + 1)$

Filters - Padding

- Convolution operator:
  - Input: a input matrix of dimension n x n, and a filter of dim. f x f
  - Output: a matrix of dim: $(n - f + 1) \times (n - f + 1)$

- Padding: Add pixels around the image corners
  - Padded with dimension p filled with 0s
  - Output matrix of dim: $(n + 2p - f + 1) \times (n + 2p - f + 1)$

Filters - Stride

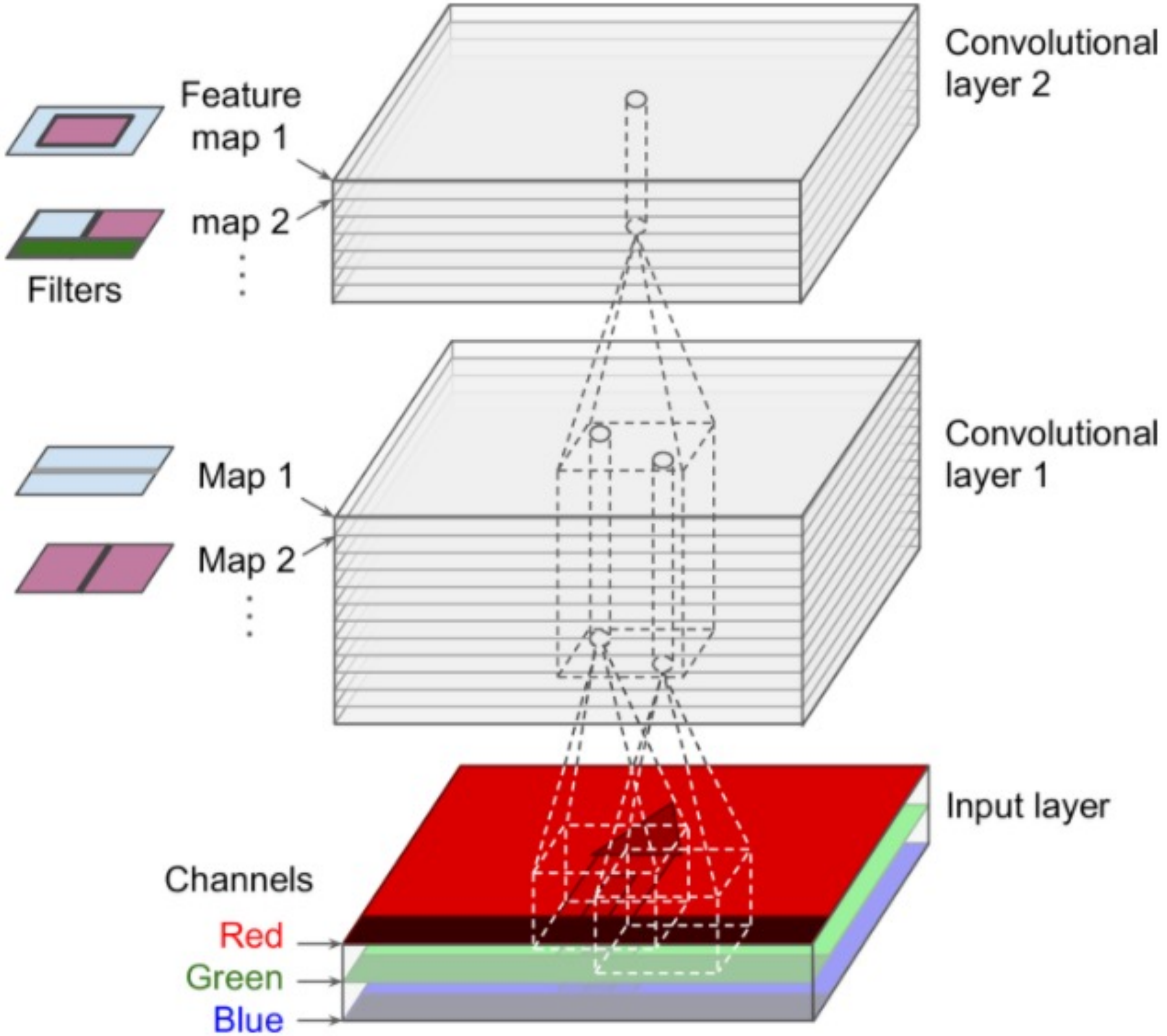
- Convolution operator:
  - Input: a input matrix of dimension n x n, and a filter of dim. f x f
  - Output: a matrix of dim: $(n - f + 1) \times (n - f + 1)$

- Padding: Add pixels around the image corners
  - Padded with dimension p filled with 0s
  - Output matrix of dim: $(n + 2p - f + 1) \times (n + 2p - f + 1)$

- Stride: Filter shifted by *s* positions
  - Output matrix of dim: $(n + 2p - f)/s + 1 \times (n + 2p - f)/s + 1$

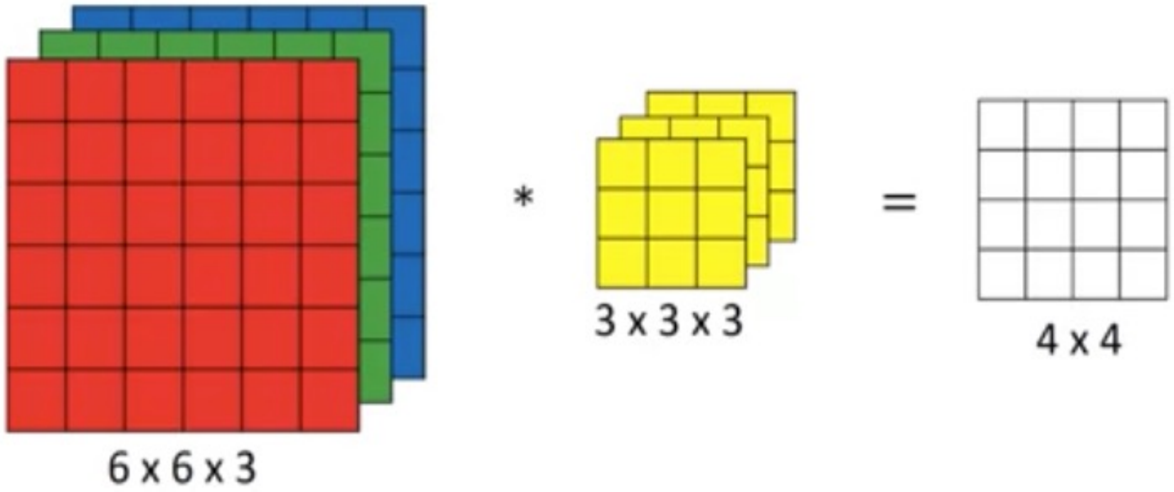
2D Convolutions

- Convolutional layers usually have multiple feature maps, and images with three color channels
- Detecting features or edges in RGB images means to apply convolution over volumes



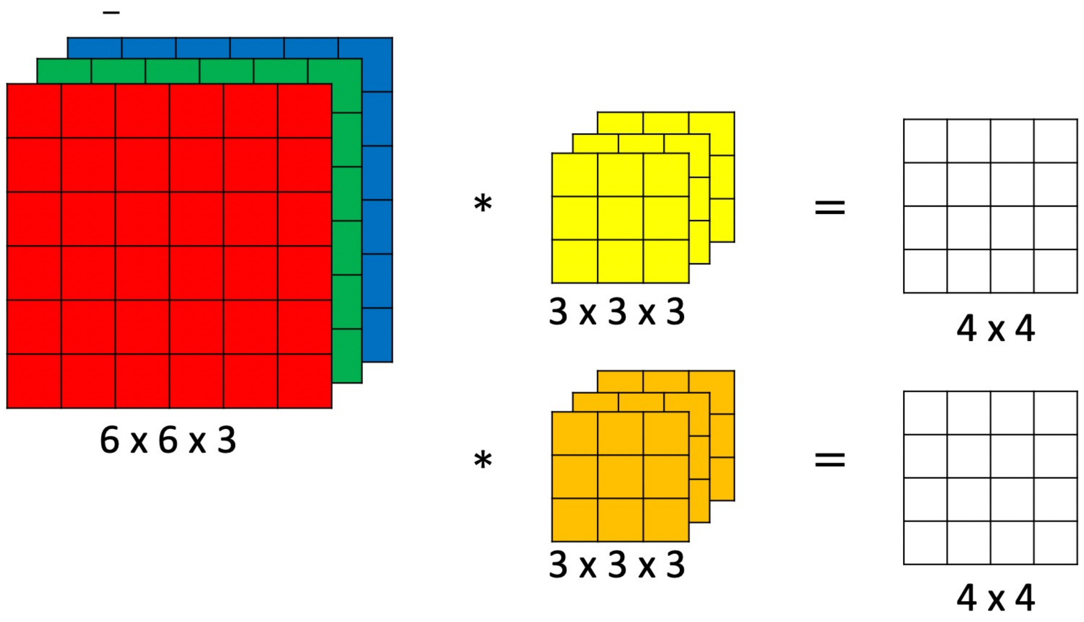
2D Convolutions

- Convolution operator on volumes:
  - Input: a input matrix of dimension h x w x 3 (RGB)
  - h x w x c, and a filter of dim. f x f x c
  - Output: a matrix of dim: n-f+1 x n-f+1



2D Convolutions – Multiple Filters

- Convolution operator on volumes with multiple filters:
  - Input
    - a input matrix of dimension h x w x c (c= 3 for RGB)
    - *k* a filter of dim. f x f x c
  - Output: a matrix of dim: n-f+1 x n-f+1 x k



Layers of a CNN

- Convolutional layers
- Pooling layers
- Fully connected layers

Pooling

- Pooling is used to reduce the size of the representation, to speed the computation, as well as make some of the features that detects a bit more robust.
- Strategies:
  - Max-pooling
  - Avg. pooling

Pooling

- Input:
  - h x w x c
- Output:
  - (h-f)/s +1 x (w-f)/s +1 x c
- Hyper-parameters
  - f: filter size (common, f=2, f=3)
  - s: stride (commonly, s=2)
  - p: normally not used (i.e., p = 0)
- No parameters to learn/train

Pooling Examples

- Max

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 2 & 9 & 1 & 1 \\ 1 & 3 & 2 & 3 \\ 5 & 6 & 1 & 2 \end{bmatrix} \qquad \begin{bmatrix} 9 & 2 \\ 6 & 3 \end{bmatrix}$$

  - Hyper-parameters:
    - $f = 2$
    - $s = 2$

- Avg.

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 2 & 9 & 1 & 1 \\ 1 & 3 & 2 & 3 \\ 5 & 6 & 1 & 2 \end{bmatrix} \qquad \begin{bmatrix} 3.75 & 1.25 \\ 4 & 2 \end{bmatrix}$$

  - Hyper-parameters:
    - $f = 2$
    - $s = 2$

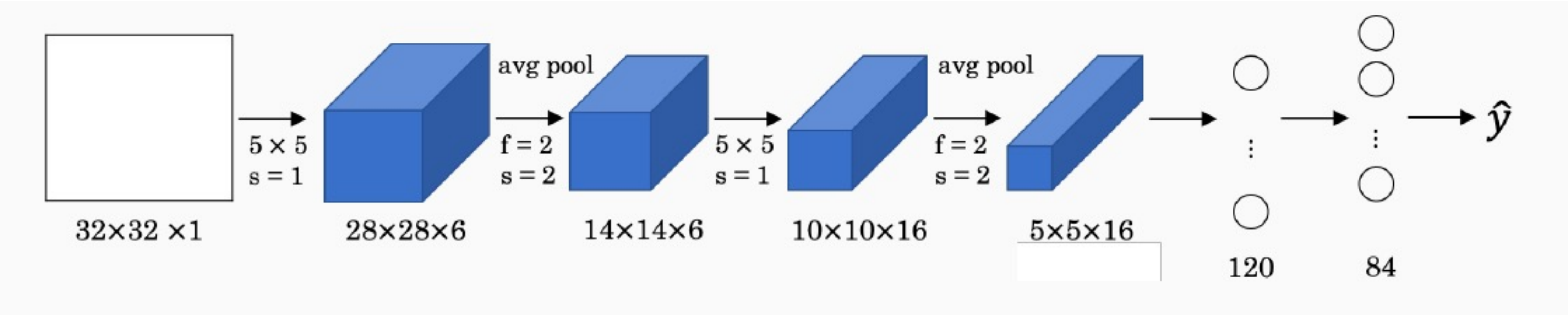
CNNs Examples

Outline

- (Very) Brief introduction to ML/DL
- Convolutional Neural Networks (CNNs)
- **CNNs Architectures**
- Computer Vision Applications
- Segmentation Framework for Hyperspectral Images Classification

CNNs Architectures – LeNet5

- LeNet-5 a neural network architecture for handwritten and machine-printed digits recognition proposed by (Lecun et al. 1998)
  - Input: 32 x 32 x 1
  - Output: 10 classes
  - It has ~60k parameters

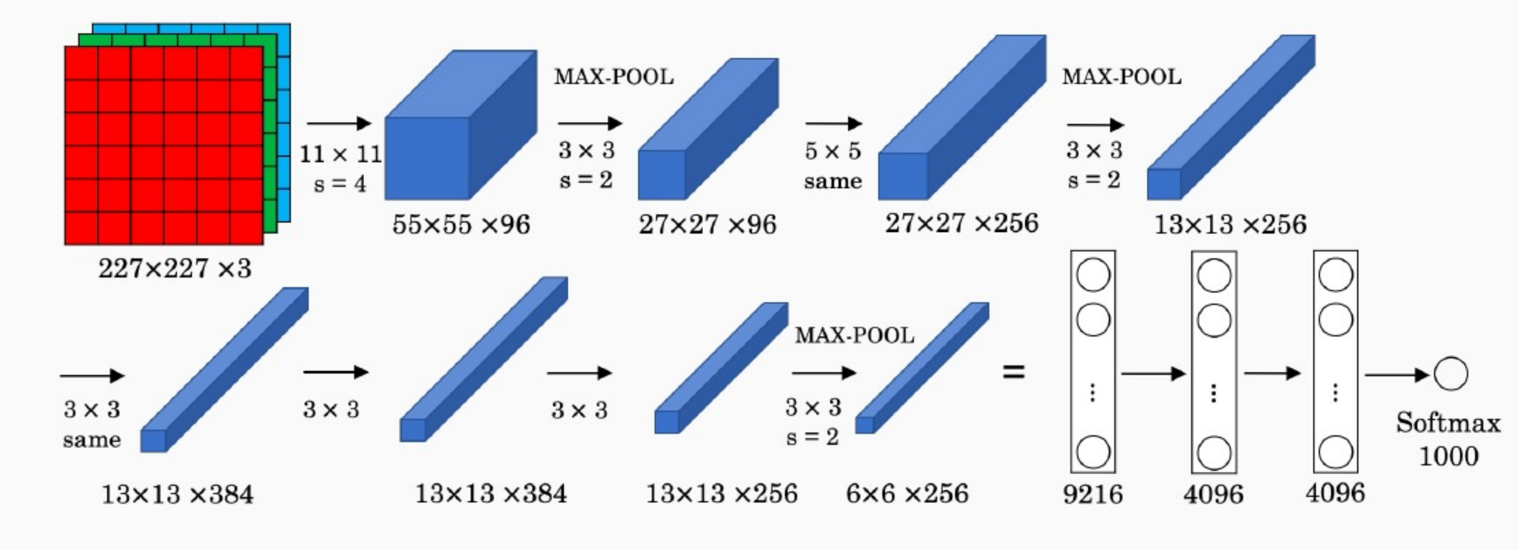


CNN Parameters - Example

	Act. shape	Act. size	#pars
Input	$32 \times 32 \times 3$	3072	0
CONV1 ($f = 5$, $s = 1$)	$28 \times 28 \times 6$	4704	456
POOL1 ($f = 2$, $s = 2$)	$14 \times 14 \times 6$	1176	0
CONV2 ($f = 5$, $s = 1$)	$10 \times 10 \times 16$	1600	2416
POOL2 ($f = 2$, $s = 2$)	$5 \times 5 \times 16$	400	0
FC3	120×1	120	48120
FC4	84×1	84	10164
softmax	10×1	10	850

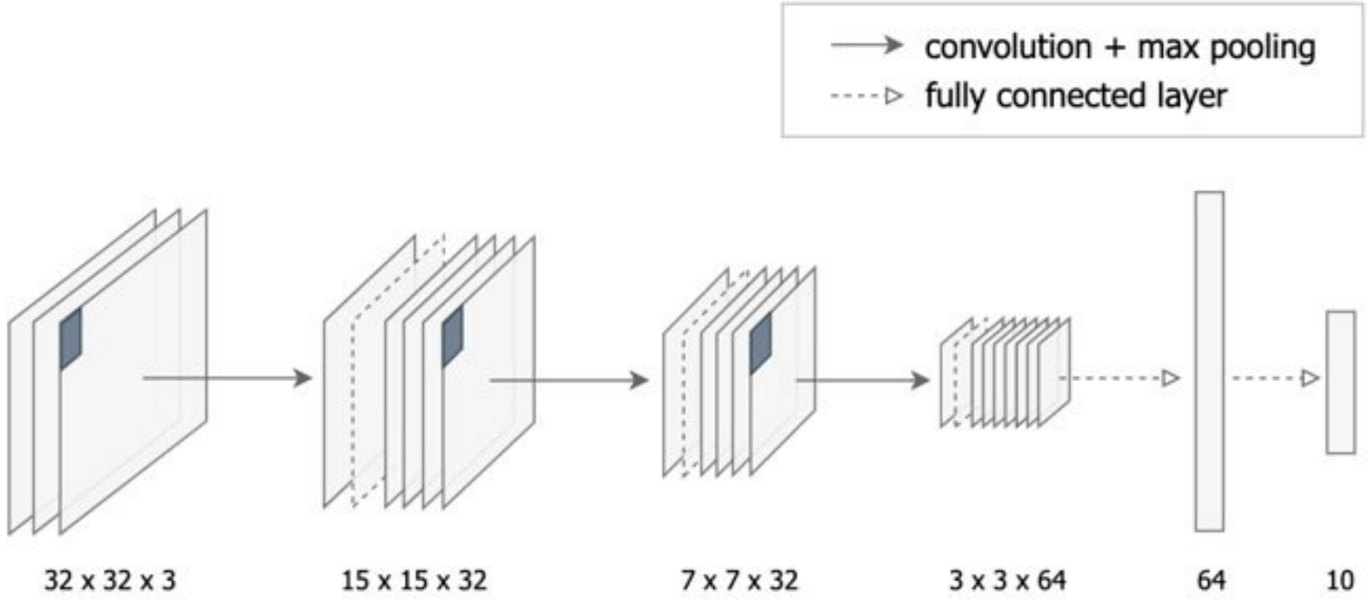
CNNs Architectures - AlexNet

- AlexNet (Krizhevsky, Sutskever, and Hinton 2012), the CNN architecture that promoted DL, is similar to LeNet5, but much deeper
  - Input: 227 x 227 x 3
  - Output: 100 classes
  - It has ~138M parameters



CNNs Architectures – Cifar10Net

- Cifar10Net is a simple and architecture used to recognise images from 10 categories, namely ‘airplane’, ’automobile’, ’bird’, ’cat’, ’deer’, ’dog’, ’frog’, ’horse’, ’ship’, ’truck’
  - Input: 32 x 32 x 3
  - Output: 10 classes
  - It has ~90k parameters



Outline

- (Very) Brief introduction to ML/DL
- Convolutional Neural Networks (CNNs)
- CNNs Architectures
- **Computer Vision Applications**
- Segmentation Framework for Hyperspectral Images Classification

Computer Vision Applications

- Classification
- Classification with Localisation
- Object Detection
- Semantic Segmentation

Localisation vs. Detection

- Object localisation problem
  - Usually there is one object to be classified and localised
  - Learning a bounding box (b_x, b_y, b_h, b_w)
- Object detection problem
  - Usually there are multiple objects

Semantic Segmentation

- In semantic segmentation, each pixel is classified according to the class of the object it belongs to (e.g., road, car, pedestrian, building, etc.)

Semantic Segmentation - Wood Detection

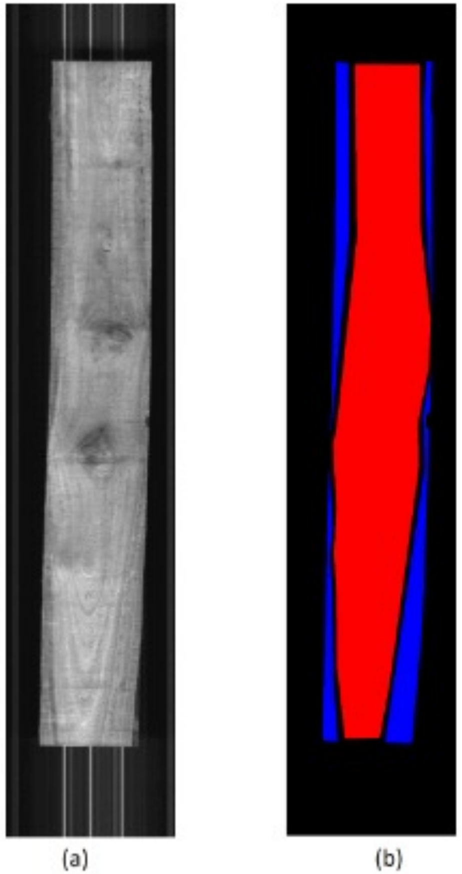
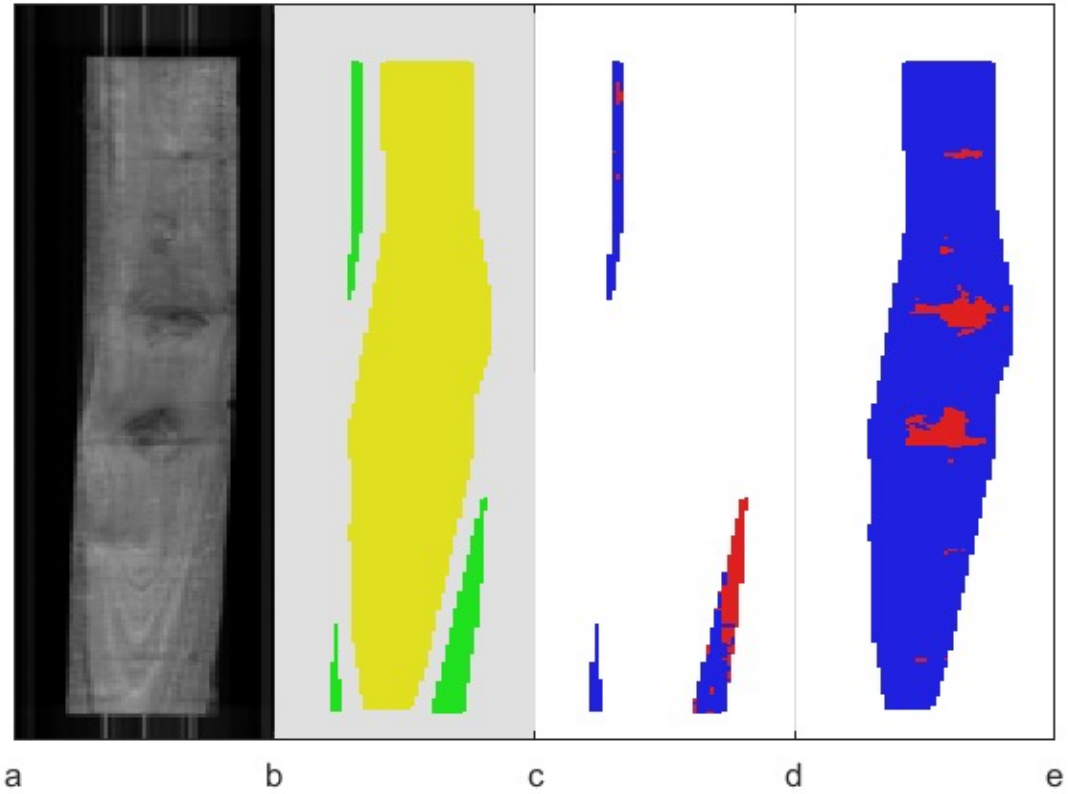
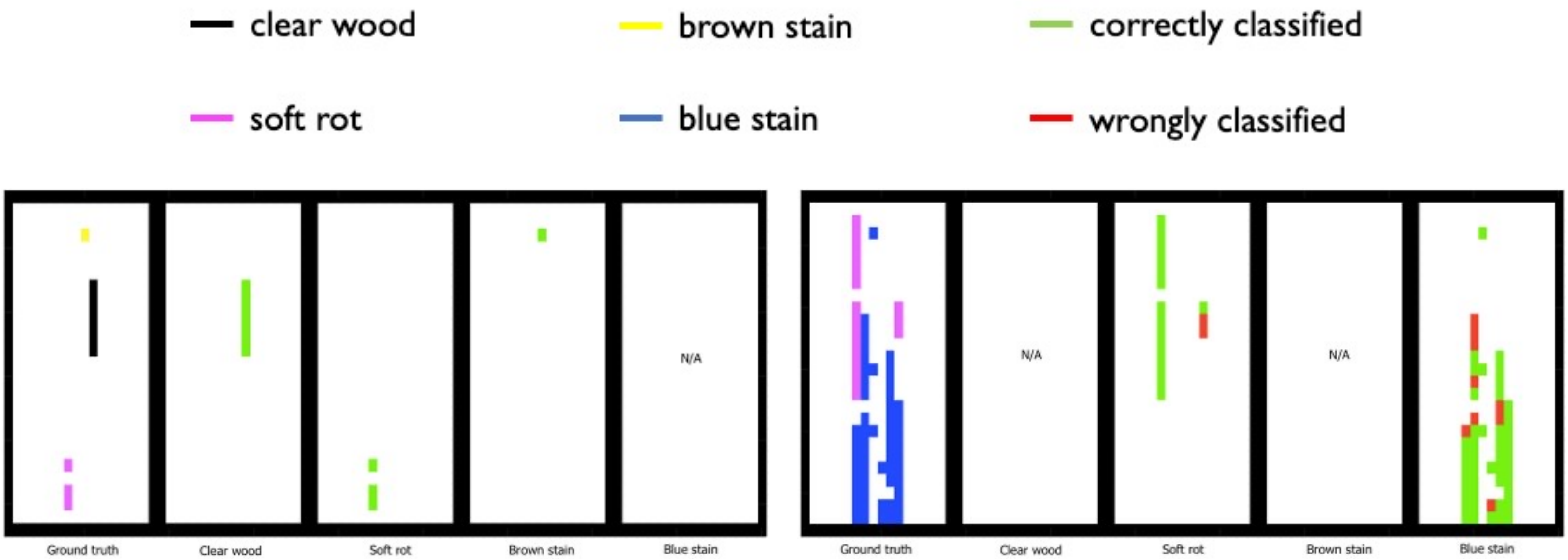


Fig. 1. Example of HS image of a board of wood. (a) is the display in gray level of the band 6, (b) is the corresponding ground truth (heartwood in red and sapwood in blue).



a to b: input image, b to c: ground truth, c to d: sapwood, d to e: heartwood
green is sapwood, yellow is heartwood
blue is correctly classified, red is uncorrectly classified

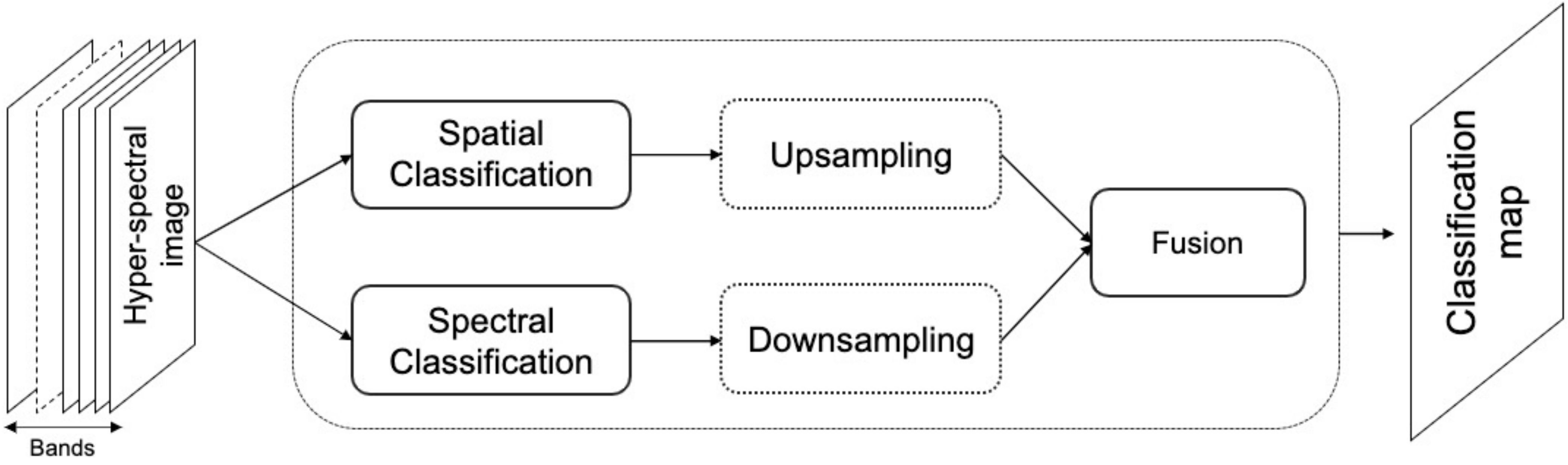
Semantic Segmentation - Fungi Detection



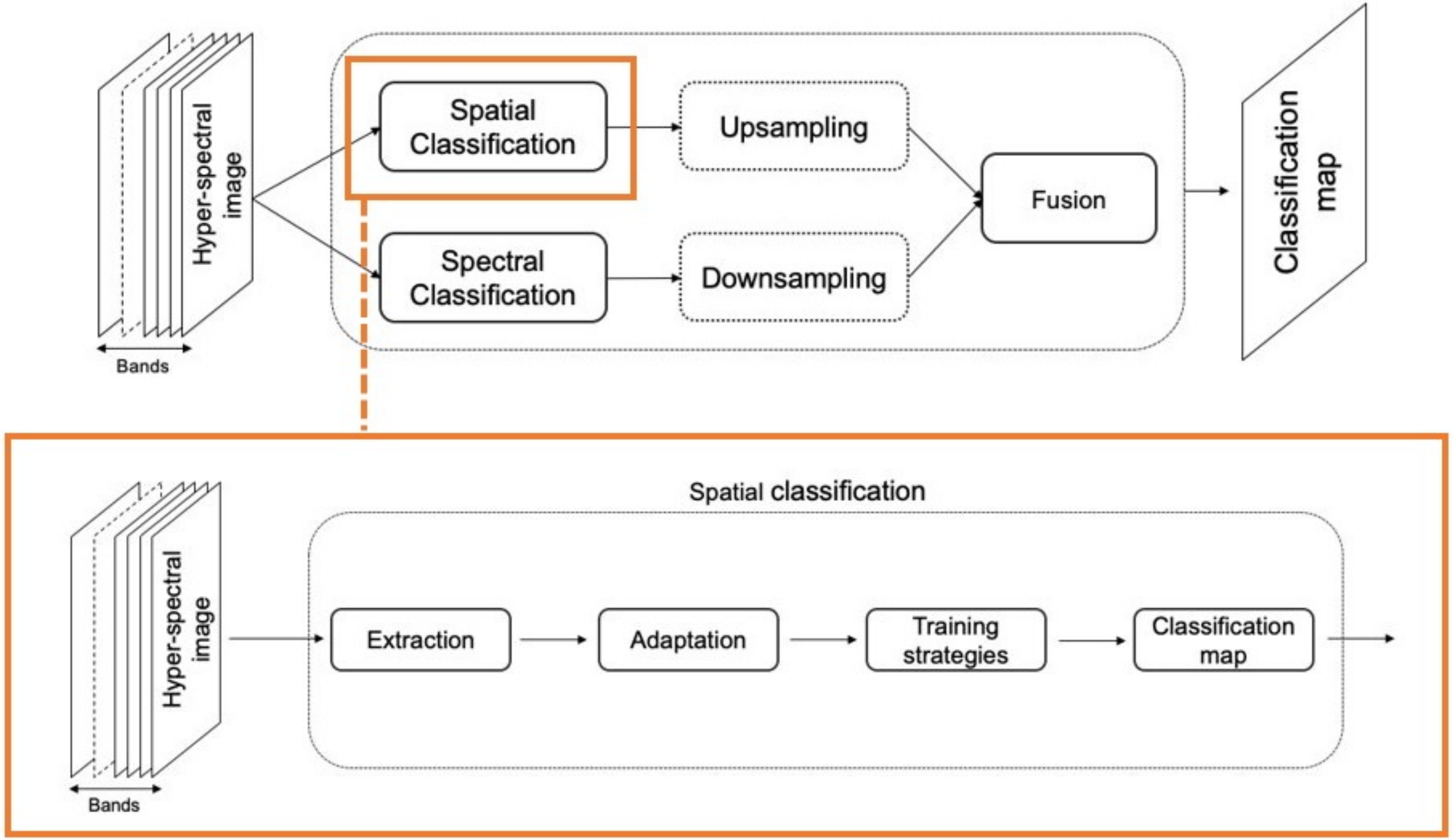
Outline

- (Very) Brief introduction to ML/DL
- Convolutional Neural Networks (CNNs)
- CNNs Architectures
- Computer Vision Applications
- **Segmentation Framework for Hyperspectral Images Classification**

Segmentation Framework for Hyperspectral Images Classification



Segmentation Framework – Spatial Classifier

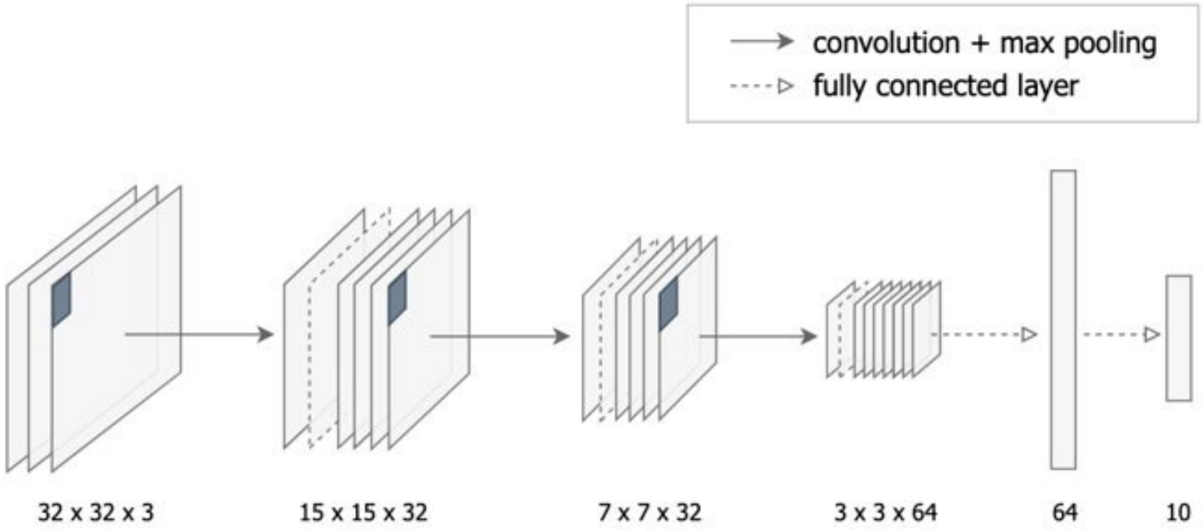


Adapting an existing network

- The adaptation of a generic image classifier involves two phases
  - Structure adaptation:
    - Handling the input of multitude spectral bands and the classification among the categories into consideration
  - Fine-tuning:
    - Exploiting already performed training on a large dataset
    - Utilize knowledge acquired for one task and leverage it to solve another similar task

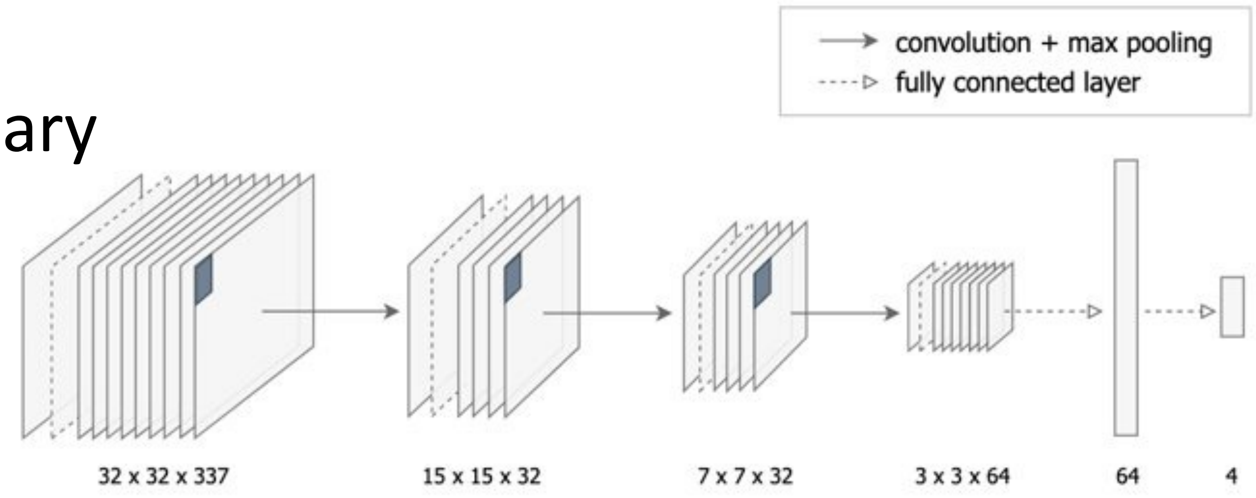
Adapting a General Image Classifier

- CNN general image classifier considered is: Cifar10Net
  - Input: 32 x 32 x 2
  - Output: 10 categories



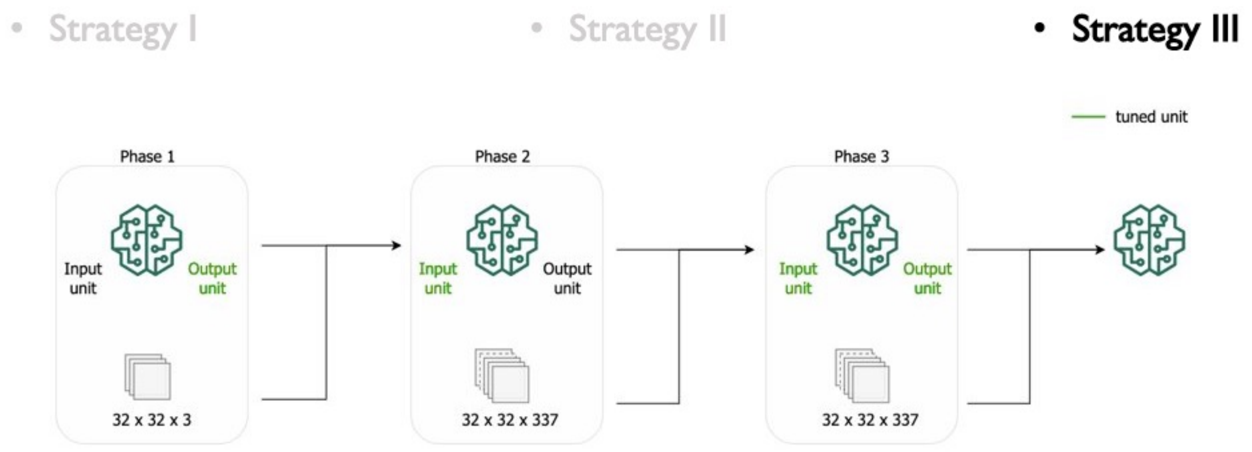
Adapting a General Image Classifier

- CNN general image classifier considered is: Cifar10Net
  - Input: 32 x 32 x 2
  - Output: 10 categories
- Architecture adaptation necessary
  - Input unit
  - Output unit



Adapting a General Image Classifier

- Fine-tuning
  - Different training strategies can be applied
  - The entire network is never trained: input and output units are tuned, with the in-between layers kept frozen



Network Adaptation Example

Thanks!

https://h2i.inf.unibz.it/
h2i-fesr@googlegroups.com