Convolutional Neural Networks summary

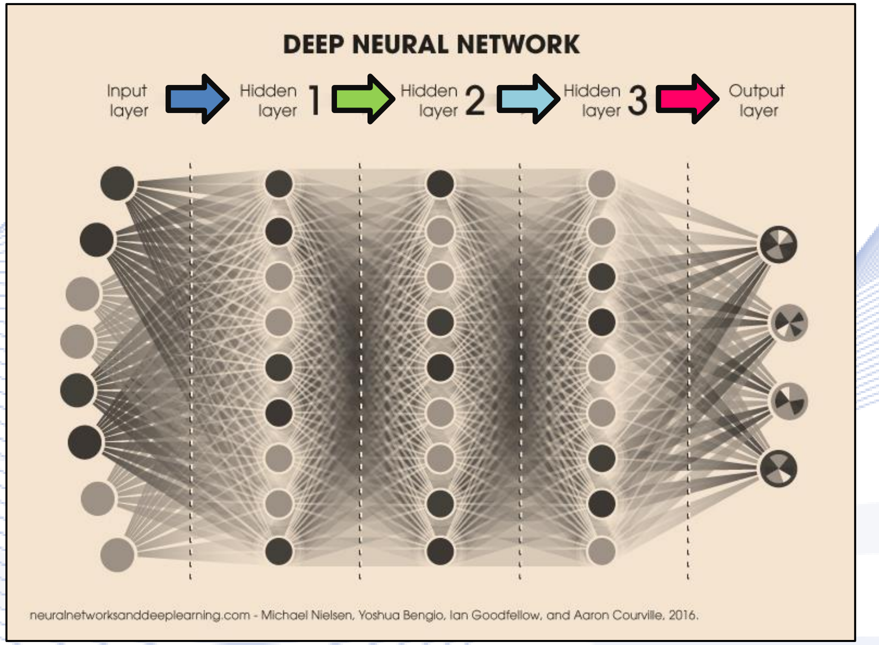
S. Papadopoulos, P. Kaplanoglou, Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr
Version 4.1

Artificial Intelligence &
Information Analysis Lab

Deep Neural Networks

Deep Neural Networks (***DNN***) have a count of layers (depth) $L \geq 3$ (typically $L \gg 3$):

- Typically, there are very many hidden layers.

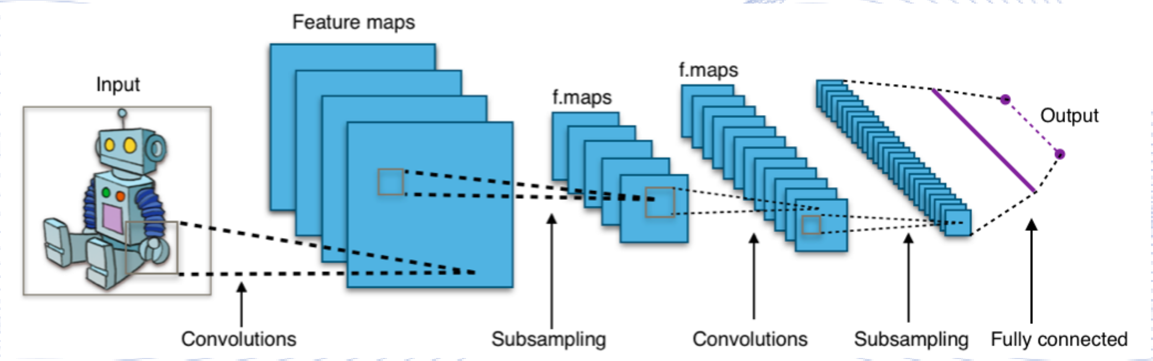


Deep Neural Network with $L = 4$

Deep Neural Networks

Convolutional Neural Networks (***DNN***):

- employ sparse connectivity (image convolutions) in the first layers.
- They may employ fully connected MLps in the last layers.



CNN structure [WIKIPEDIA].

Reference CNN Classifier

The AlexNet CNN

Convolutional Module 1
- Convolutional Layer 1
- Max Pooling 1
- Local Response Normalization 1

Convolutional Module 2
- Convolutional Layer 2
- Max Pooling 2
- Local Response Normalization 2

Convolutional Module 3
- Convolutional Layer 3
- Local Response Normalization 3

Convolutional Module 4
- Convolutional Layer 4
- Local Response Normalization 4

Convolutional Module 5
- Convolutional Layer 5
- Max Pooling 5
- Local Response Normalization 5

Fully Connected Module 1
- FC Layer 6
- Dropout

Fully Connected Module 2
- FC Layer 7
- Dropout

Softmax Output Module
- FC Layer 8
- Softmax Function

- There are several *convolutional layers.*
- There is a classifier that consists of 3 *fully connected* layers.
- Class prediction are given by *softmax* functions.

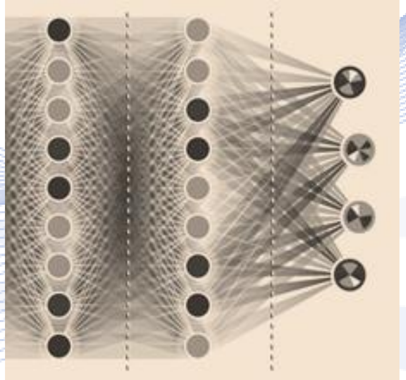


Image Convolution

2D Image convolution of a $M_1 \times M_2$ ***convolution kernel*** (or 2D filter) w with an image x of size $N_1 \times N_2$, is defined by:

$$y(i,j) = \sum_{k_1=0}^{M_1-1} \sum_{k_2=0}^{M_2-1} w(k_1,k_2)x(i-k_1, j-k_2).$$

- It describes the output of a ***2D Finite Impulse Response*** (***FIR***) ***filter***.
- If filter window has odd size $(M_1 = 2\nu_1 + 1,\ M_2 = 2\nu_2 + 1)$ and is centered around $(0,0)$, 2D convolution takes the form:

$$y(i,j) = \sum_{k_1=-\nu 1}^{\nu_1} \sum_{k_2=-\nu 2}^{\nu_2} w(k_1,k_2)x(i-k_1, j-k_2).$$

Convolutional Neural Networks

* ***Zero padding*** the input image edges by a ν pixel wide border ribbon allows convolution operator to operate on the entire input image domain.
* Padding is arbitrary and can be done by any other pixel value, e.g., the ones of the outermost image rows and column pixels.
* If no padding is performed, the output image has reduced size $(N_1 - 2\nu) \times (N_2 - 2\nu)$.

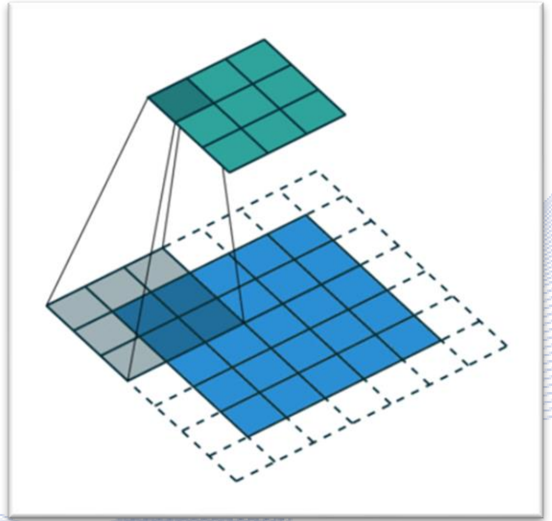


Image Convolution

- Image blurring:

Original image

$$\mathbf{W} = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}.$$

Convolution output

Image Convolution

- Edge detection:

Original image

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}.$$

Convolution output

Artificial Intelligence & Information Analysis Lab

Convolutional Layer

- The 2D matrix $\mathbf{X}_{ij}(r)$ is a slice of $\mathbf{X}_{ij}$that contains the values of channel r (e.g. intensity of red/green/blue) for the $M_1 \times M_2$ image block centered at $[i, j]^T$:

$$\mathbf{X}_{ij}(r) = [x(k_1, k_2, r): k_1 = i - \nu_1, \dots, i + \nu_1, k_2 = j - \nu_2, \dots, j + \nu_2].$$

- Typically, small $M_1 \times M_2$ image blocks, e.g., $3 \times 3, \dots, 11 \times 11$, are employed.
- We can define convolutions operating on any of the $r = 1, \dots, d_{in}$ input feature channels and producing any of the $o = 1, \dots, d_{out}$ output feature channels:

$$y(i, j, o) = \sum_{k_1=0}^{M_1-1} \sum_{k_2=0}^{M_2-1} w(k_1, k_2, r, o) x(i - k_1, j - k_2, r).$$

Convolutional Layer

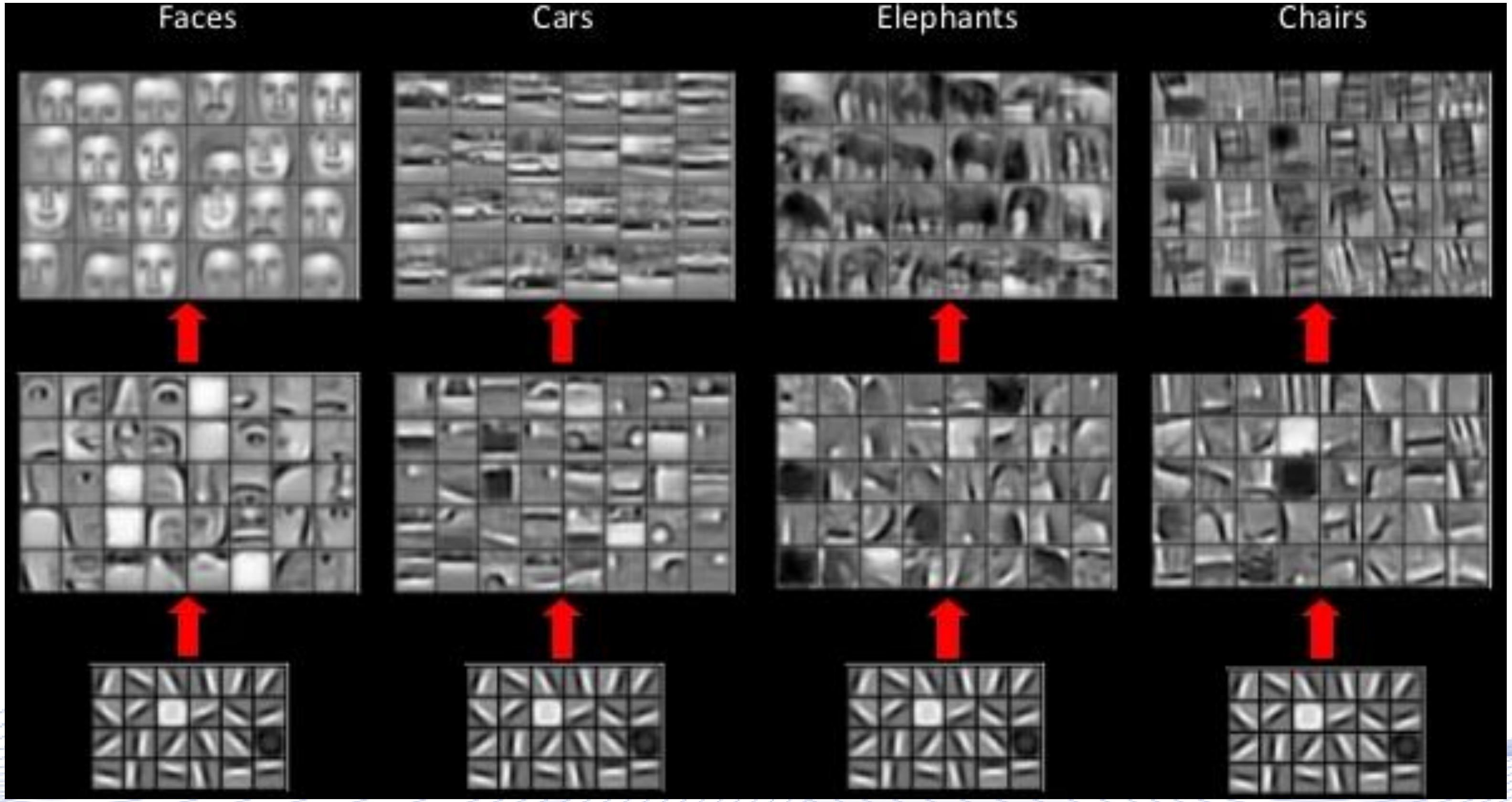
Multichannel input-output:

- For a convolutional layer l with an activation function $f_l(\cdot)$, multiple incoming features d_{in} and one single output feature o:

$$y^{(l)}(i,j,o) = f_l\left(b^{(l)} + \sum_{r=1}^{d_{in}} \sum_{k_1=-v_1^{(l)}}^{v_1^{(l)}} \sum_{k_2=-v_2^{(l)}}^{v_2^{(l)}} w^{(l)}(k_1,k_2,r,o)\, x^{(l)}(i-k_1, j-k_2, r) \right).$$

- ***Contrary to claims even by senior scientists, this is a 2D convolution and NOT a 3D one!***
- It includes a weighted summation across the input feature channels.

Neural Image Features

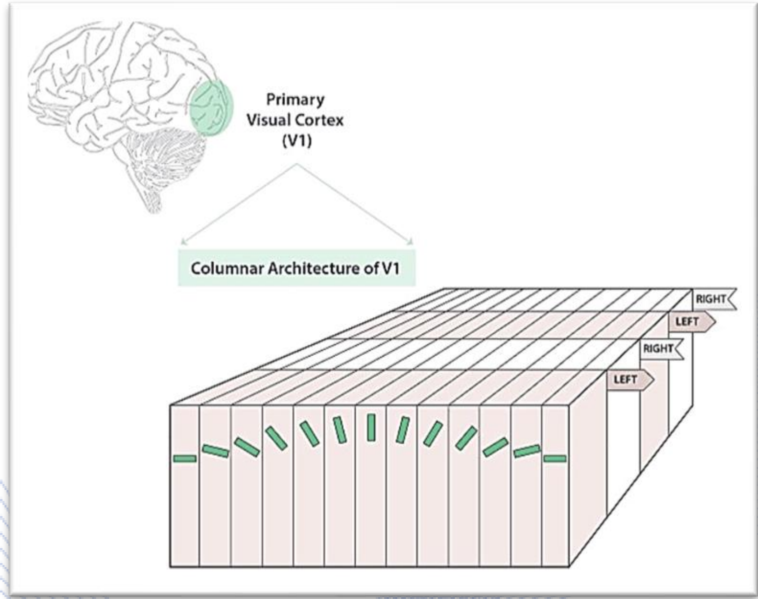


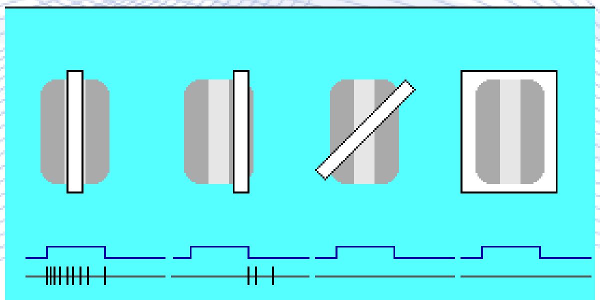
Early CNN layers can capture local and simple features, such as edges. Last CNN layers are able to learn much more complex concepts, such as entire objects.

Convolutional Layer

Biological motivation:

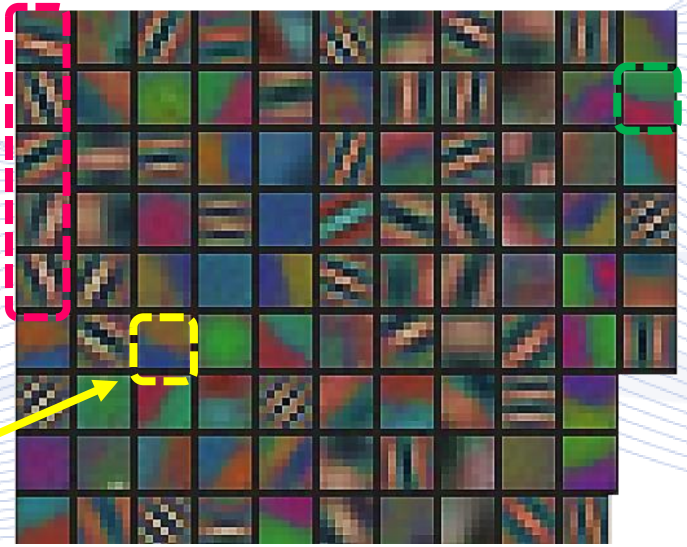
- CNNs were inspired by brain neurons in the mammalian primary visual cortex (V1).
- V1 cells are mapped to the same local region of the retina, forming ***hypercolumns***.
- V1 simple cells detect image lines and are sensitive to orientation.





Neural Image Features

- Visualizing the $d_{out} = 96$ features learned from RGB pixels in the 1st layer of the Zeiler & Fergus CNN (ZFNet) shows characteristics of biological vision.
  - ZFNet is an improvement of AlexNet with 1st layer: $[7 \times 7/2|\, 3 \rightarrow 96]$
  - Feature visualization indicated poorly trained convolutional kernels in AlexNet.



Orientation selectivity found in V1 simple cells

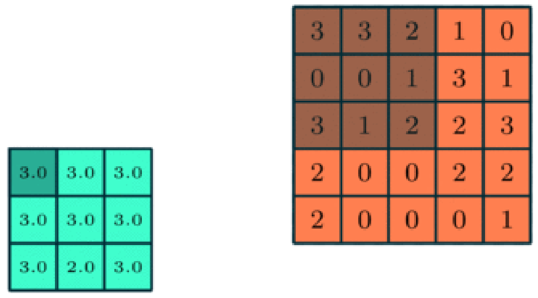
Green/red color opponency observed in retinal neurons and human visual perception

Blue/yellow color opponency observed in retinal neurons and human visual perception

Pooling Layers

Pooling layers are added inside a CNN architecture primarily for downsampling, aiming to reduce the computational cost. Secondarily helps on translation invariance.

- The pooling window is moved over an activation map $\mathbf{A}_{ij}^{(l)}(o)$ along i, j with stride s.
- Typical pool window sizes $2 \times 2,\ 3 \times 3$.
- Downsampling usually with $s = 2$.
- Pools could overlap, e.g., [3×3 / 2]



- Ad-hoc decision to use pooling or not.
- No formal justification for the effect of overlapping on pooling regions.

Activation Functions

- ***Sigmoid*** and ***hyperbolic tangent*** function are not proper for CNNs, because they lead to the *vanishing gradients* problem.

- ***Rectifiers*** are more suitable for activation functions.
  - **ReLU** - Rectified Linear Unit

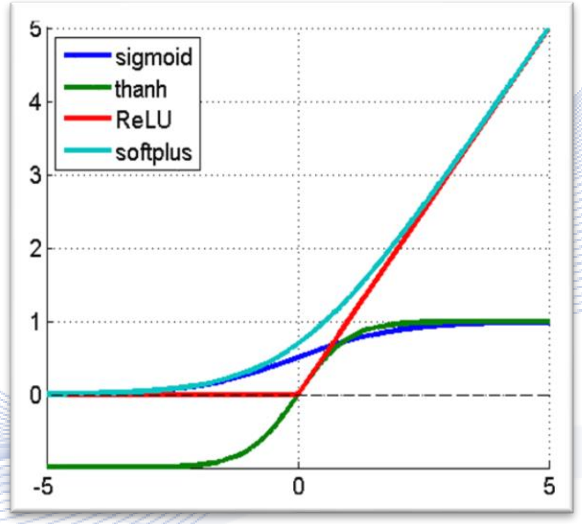
    $$y = ReLU(u) = \max\{u, 0\} \quad : \mathbb{R} \to [0, +\infty)$$

  - **ReLU6** - Rectified Linear Unit Bounded by 6

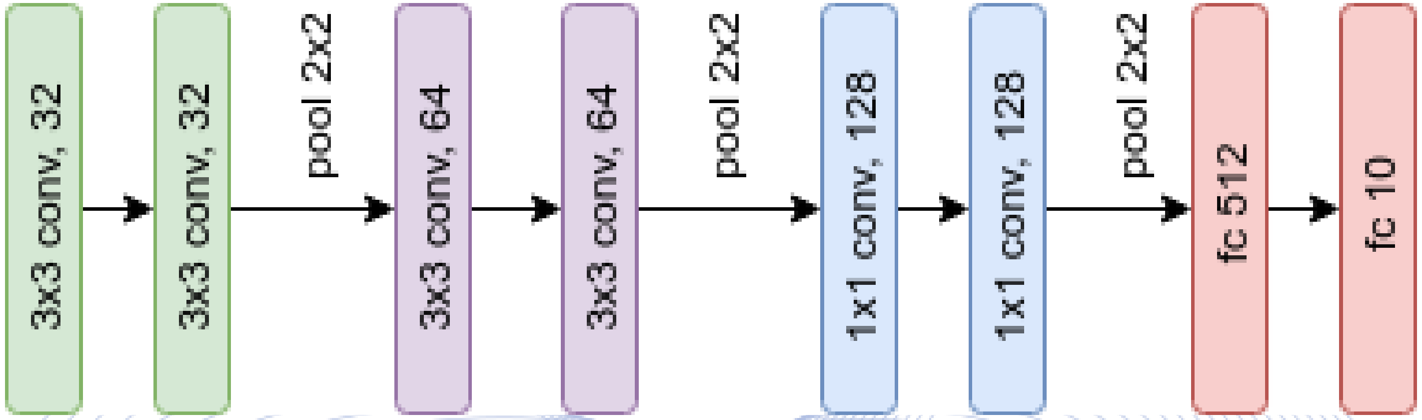
    $$y = min\ \{ReLU(u), 6\} = \min\{\max\{u, 0\}, 6\} \quad : \mathbb{R} \to [0,6]$$

  - **Softplus**

    $$y = softplus(u) = log(1 + e^u) \quad : \mathbb{R} \to [0, +\infty)$$



Convolutional Neural Networks

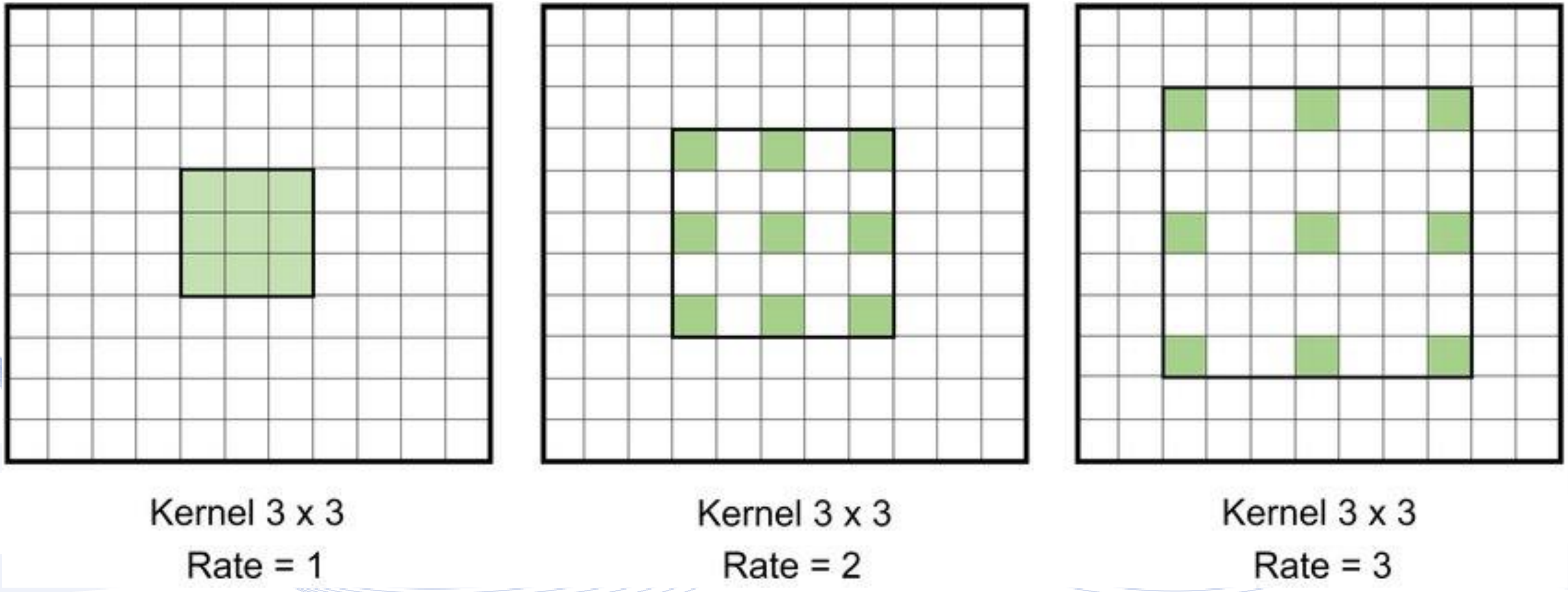


Multilayer CNN architecture.

Artificial Intelligence &
Information Analysis Lab

Special convolution types

Atrous (Dilated) Convolution:



[MOR2018]

Special convolution types

1×1 *convolution*:

- In general, a convolutional layer l with an activation function $f_l(\cdot)$, multiple incoming features d_{in} are convolved with $M_1 \times M_2$ convolution maks and summed to produce one single output feature o:

$$y^{(l)}(i,j,o) = f_l\left(b^{(l)} + \sum_{r=1}^{d_{in}} \sum_{k_1=1}^{M_1} \sum_{k_2=1}^{M_2} w^{(l)}(k_1,k_2,r,o)\, x^{(l)}(i-k_1, j-k_2, r)\right).$$

- In case that $M_1 = M_2 = 1$, a 1×1 convolution results in:

$$y^{(l)}(i,j,o) = f_l\left(b^{(l)} + \sum_{r=1}^{d_{in}} w^{(l)}(1,1,r,o)\, x^{(l)}(i,j,r)\right).$$

CNN Training

- ***Mean Square Error (MSE)***:

$$J(\boldsymbol{\theta}) = J(\mathbf{W}, \mathbf{b}) = \frac{1}{N}\sum_{i=1}^{N}\|\hat{\mathbf{y}}_i - \mathbf{y}_i\|^2.$$

  - It is suitable for regression and classification.

- ***Categorical Cross Entropy Error***:

$$J_{CCE} = -\sum_{i=1}^{N}\sum_{j=1}^{m} y_{ij}\log(\hat{y}_{ij}).$$

  - It is suitable for classifiers that use softmax output layers.

Backpropagation

- The most widespread algorithm for supervised training of CNNs is the Backpropagation algorithm.
- Unlike typical MLPs, the summation that occurs inside each convolutional neuron during forward pass uses convolution instead of normal multiplication.
- Training is performed as a problem of minimization of a cost function.
- MSE is the most common cost function.

CNN Training

- CNNs are trained with the same gradient descent methods as multilayer perceptron.
  - Convolution is a differentiable operation.
  - Mini-batch Stochastic Gradient Descent methods are used for image batches.
- Optimization methods:
  - *Learning rate decay* are scheduled changes to the learning rate at the various training epochs. For non-adaptive mini-batch SGD methods, e.g. *Momentum*.
  - *ADAM* is an optimization method with an adaptive learning rate.
- Large scale datasets are needed to adequately train a CNN.
  - CNNs are prone to over-fitting.
  - Training images count in the magnitude of 10 or 100 thousands.

CNN Training

Data Augmentation:

- It is used to avoid overfitting.
- The training image set is augmented during training with label-preserving transformation of the samples:
    - Image translations and random image crops.
    - Photometric distortions, i.e. altering the intensities of RGB channels.
    - Scaling and rotation, e.g. at $\leq 90^o$
    - Vertical reflections, e.g. mirror.
    - Addition of Salt and Pepper noise.
- Data augmentation can be done with minimal computation cost inside the training process.

CNN Training

Softmax Layer:

- It is the last layer in several neural network classifier.
- The response of neuron i in the softmax layer L is calculated with regard to the value of its activation function $a_i = f(z_i)$:

$$\hat{y}_i = g(a_i) = \frac{e^{a_i}}{\sum_{k=1}^{k_L} e^{a_k}} : \mathbb{R} \to [0,1], \qquad i = 1, .., k_L,$$

$$\sum_{i=1}^{k_L} \hat{y}_i = 1.$$

- The responses of softmax neurons sum up to one:
- Better representation for mutually exclusive class labels.

CNN Training

- Batch Normalization:

$$BN_{\gamma,\beta,\varepsilon}(x_{ij}) = \gamma \frac{(x_{ij} - \overline{x_j})}{\sqrt{s_j^2 + \varepsilon}} + \beta, \qquad i = 1,..,N_B, j = 1,\dots,N_1 \times N_2.$$

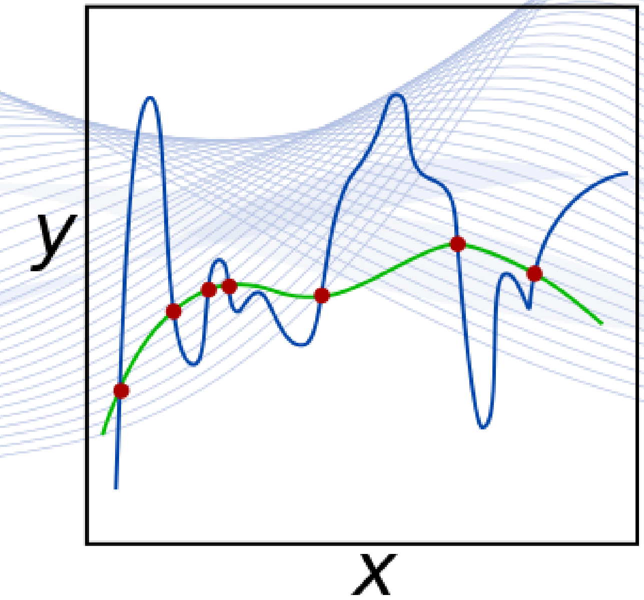
- Mean and Sample Standard Deviation of each of the pixels in each of the minibatches:

$$\overline{x_j} = \frac{1}{N_B}\sum_{i=1}^{N_B} x_{ij},$$

$$s_j^2 = \frac{1}{N_B}\sum_{i=1}^{N_B} (x_{ij} - \overline{x_j})^2.$$

CNN Training

- Depending on the functional form of $\Omega(\cdot)$, the effect on the model parameters is different:
    - L_2 regularization: $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||^2 = \sum_i \theta_i^2$.
    - L_1 regularization: $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{\theta}|| = \sum_i |\theta_i|$.



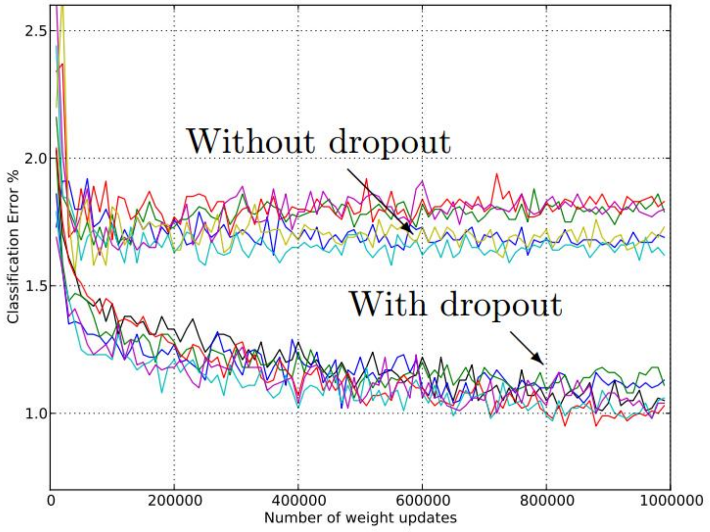
CNN Training

Dropout randomly excludes a set of neurons from a certain training epoch with a constant keep out probability p_{keep}.

- Activations of dropped out neurons are set to zero.
  - They do not participate in the loss, thus excluded from back-propagation.
  - Dropout was initially used in AlexNet after each fully connected layer.
  - During testing a trained model, all neurons are used with their already learned weights.

- Induces dynamic sparsity during training.

- Prevents complex co-adaptations of the synaptic weights, that may lead to correlated activations of neurons.

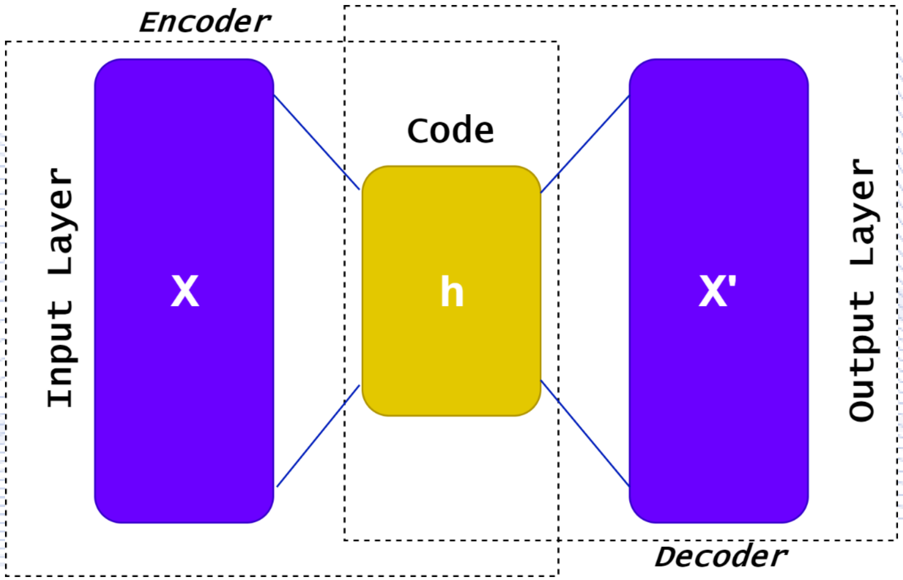


Artificial Intelligence & Information Analysis Lab

Deep Autoencoders

Given a sample $\mathbf{x} \in \mathbb{R}^n$ and a function $\mathbf{y} = \boldsymbol{f}(\mathbf{x}; \boldsymbol{\theta})$, the model output $\mathbf{y}$ should be equal to the model input $\mathbf{x}$:

- **Training**: Given N pairs of training examples $\mathcal{D} = \{\mathbf{x}_i, i = 1, \dots, N\}$, where $\mathbf{x}_i = \mathbf{y}_i \in \mathbb{R}^n$, estimate $\boldsymbol{\theta}$ by minimizing a loss function: $\min_{\boldsymbol{\theta}} J(\mathbf{x}, \hat{\mathbf{y}})$.



Autoencoder structure.

CNN architectures

Residual Convolutional Module:

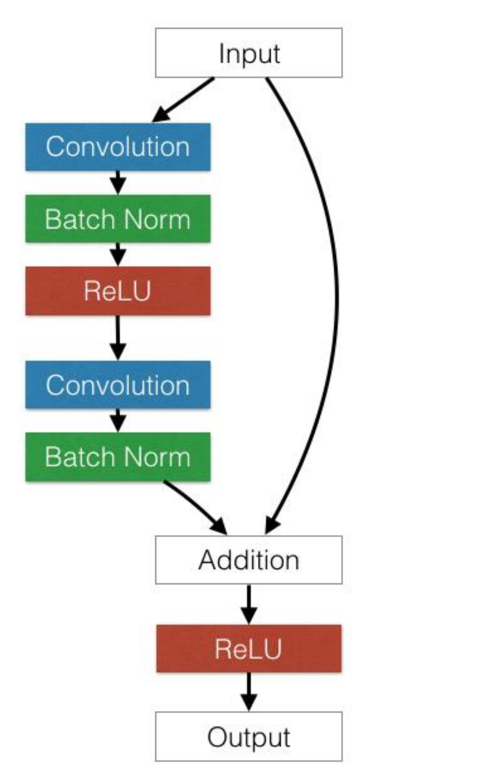
- The basic module of *ResNets.*

- A *shortcut connection* bypasses layers. BN is used before the activation function. These implement *identity mapping*.

$$\mathbf{A}^{(l+2)} = f_{l+2}(\mathbf{A}^{(l)} + BN_{l+2}\left(\mathbf{b}^{(l+2)} + \mathbf{W}^{(l+2)} ** f_{l+1}(BN_{l+1}(\mathbf{b}^{(l+1)} + \mathbf{W}^{(l+1)} ** \mathbf{A}^{(l)}))\right)$$

  - BN_l is the batch normalization that is applied in layer l

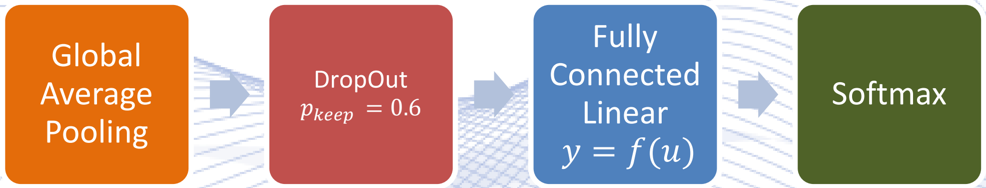
- *Residual learning* makes possible to train extremely deep CNNs up to 192 layers.



CNN architectures

Inception Output Classifier

- It contains a global average pooling, 40% dropout and a fully connected (FC) layer of $d_{out} = 1024$ neurons with a linear activation function before the Softmax layer.
  - It replaces the memory demanding classifier that uses two FC layers of $d_{out} = 4096$ each.



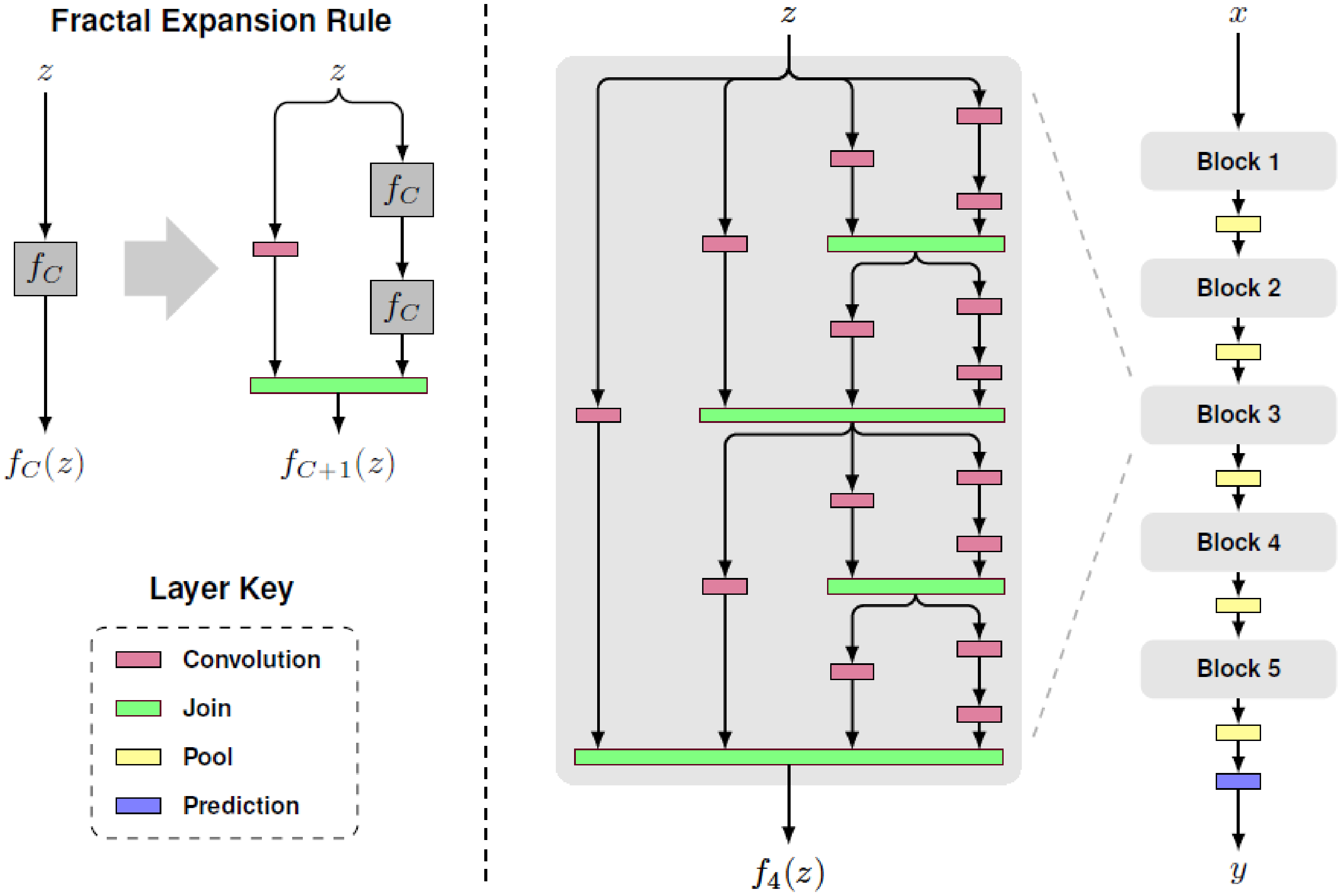
- There are two auxiliary classifiers in the Inception v1 (GoogleLeNet) CNN for two additional gradient flows.

CNN architectures

FRACTALNET:

- For $C = 1$, $f_1(\mathbf{z}) = conv(\mathbf{z})$.
- Successive fractals are formed as: $f_{C+1}(\mathbf{z}) = [(f_C \circ f_C)(\mathbf{z})] \oplus [conv(\mathbf{z})]$,

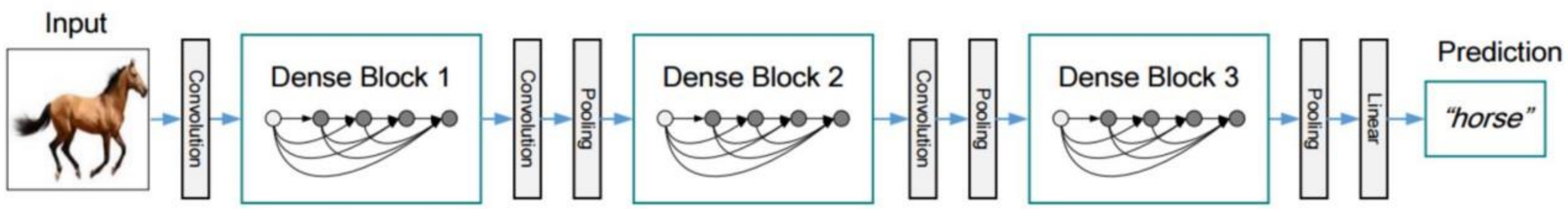
- $\circ$ denotes composition
- $\oplus$ denotes element-wise mean.



CNN architectures

DenseNet:

- It expands ResNet shortcut connection logic.
- At ***dense-block level***, the outputs of all preceding layers are used as inputs for each layer and its output is fed to all successive layers.

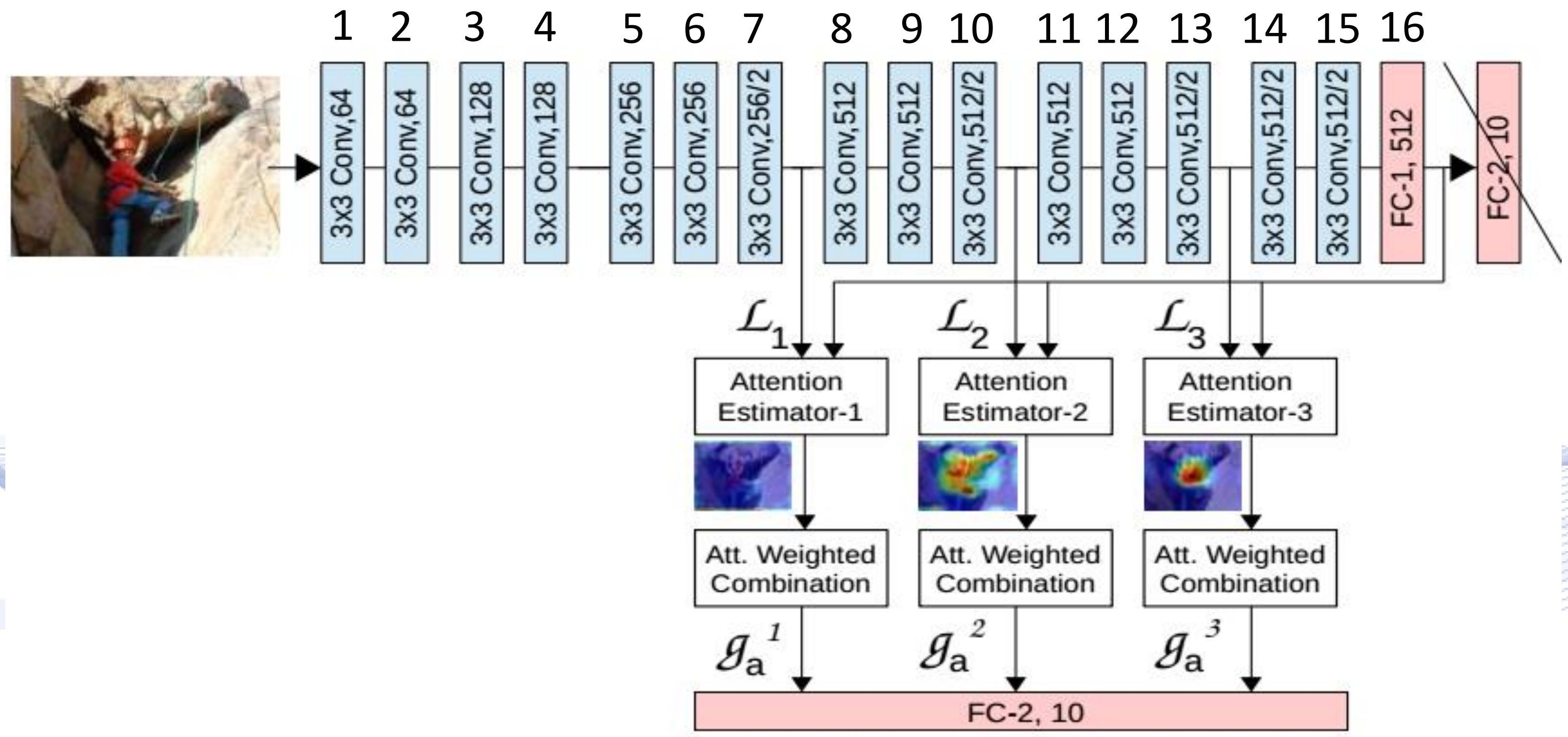


- Since each layer receives feature maps from all preceding layers, the network can be thinner and more compact.

CNN architectures

Visual Attention:

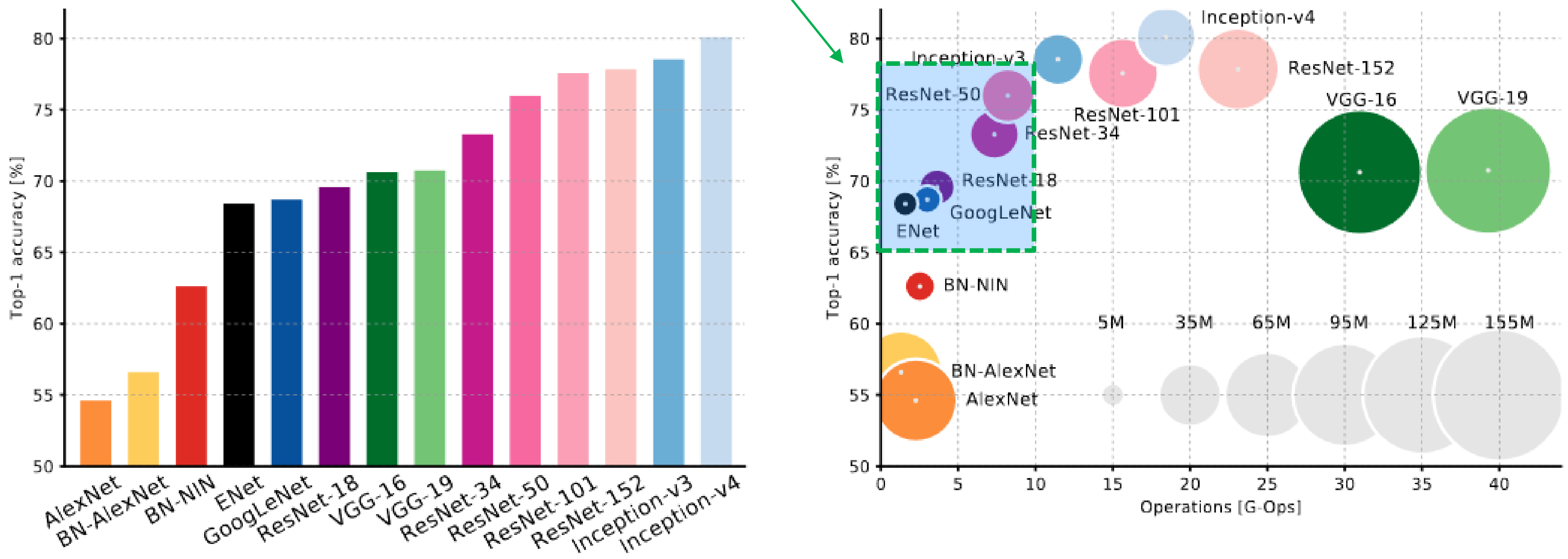
- Attention mechanism applied on a VGG-16.



CNN Computational Issues

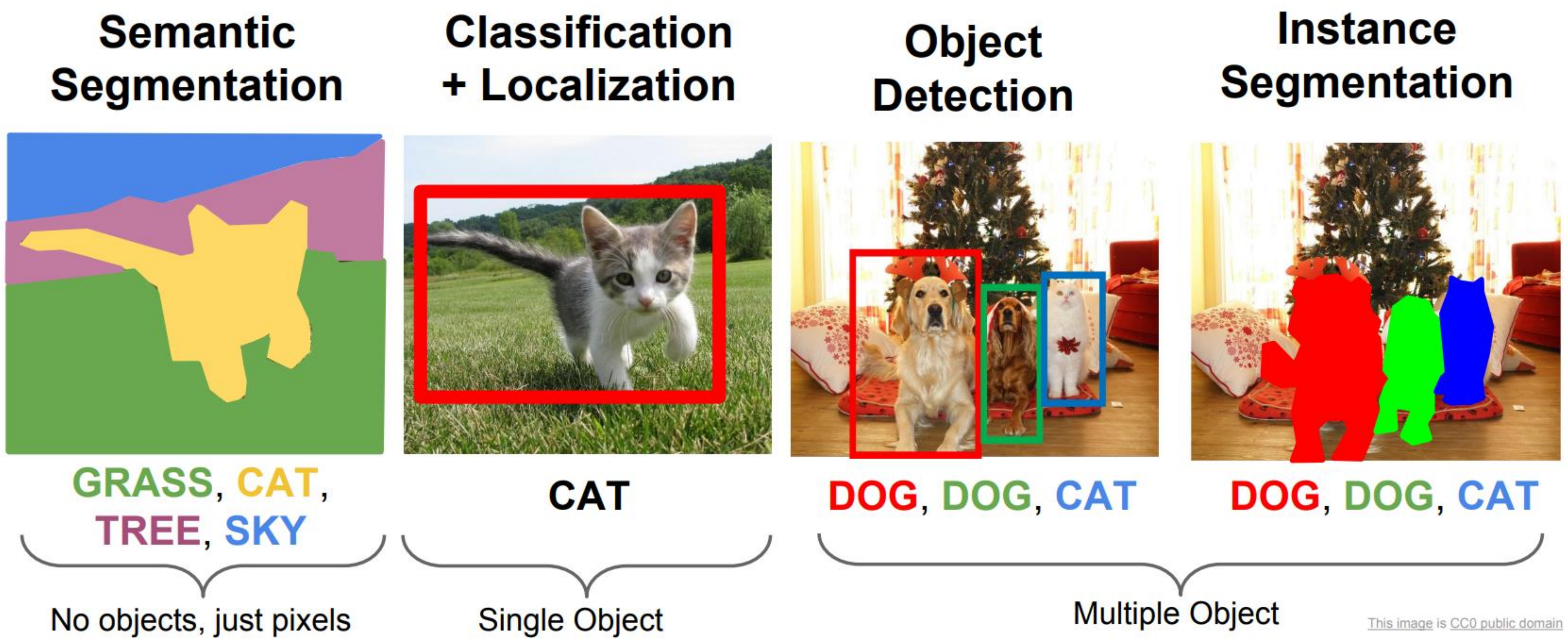
Candidates for real-world applications

“Goldilocks” zone of CNN models



A. Canziani, A. Paszke, and E. Culurciello, “An Analysis of Deep Neural Network Models for Practical Applications,” *arXiv:1605.07678 [cs]*, May 2016.

CNN Use Cases

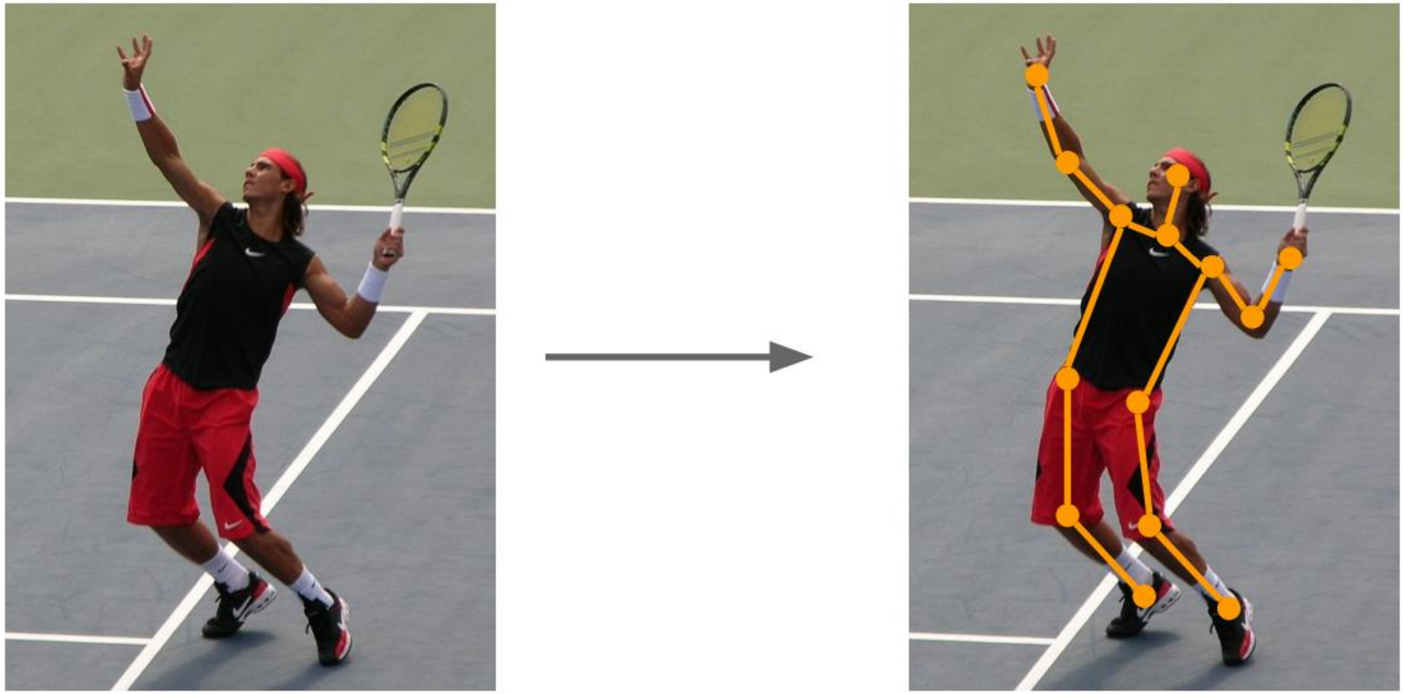


[LI2017]

Artificial Intelligence & Information Analysis Lab

CNN Use Cases

Human Pose Estimation



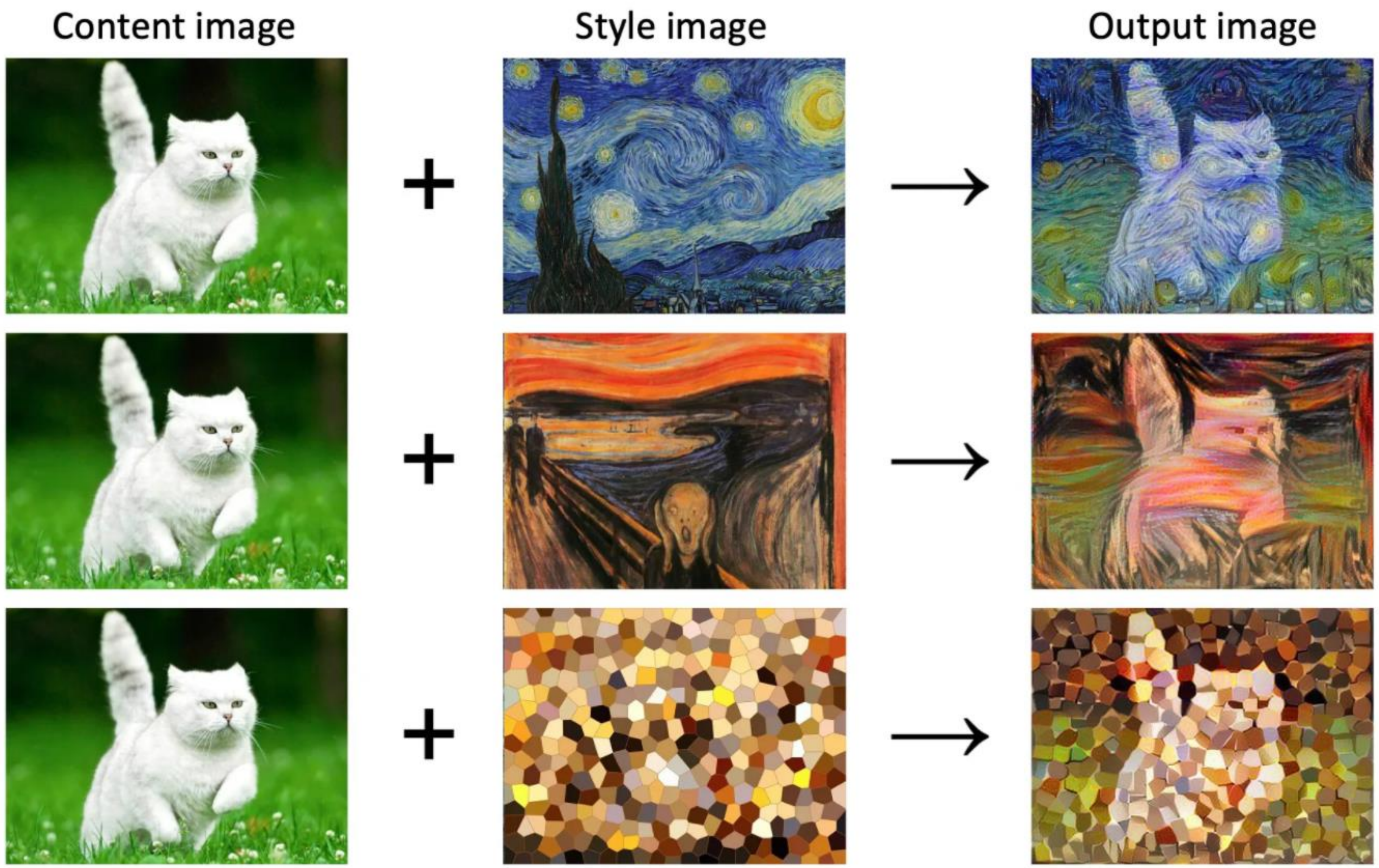
Represent pose as a set of 14 joint positions:

Left / right foot
Left / right knee
Left / right hip
Left / right shoulder
Left / right elbow
Left / right hand
Neck
Head top

[JOH2010]

CNN Use Cases

Style Transfer



[GDD2019]

Q & A

Thank you very much for your attention!

Contact: Prof. I. Pitas
pitas@csd.auth.gr