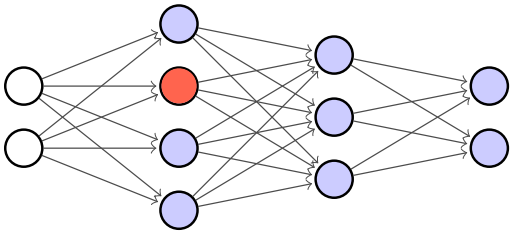
EPFL

Image Processing I



Chapter 6
Convolutional neural networks

Prof. Michael Unser, LIB

December 2022

OUTLINE

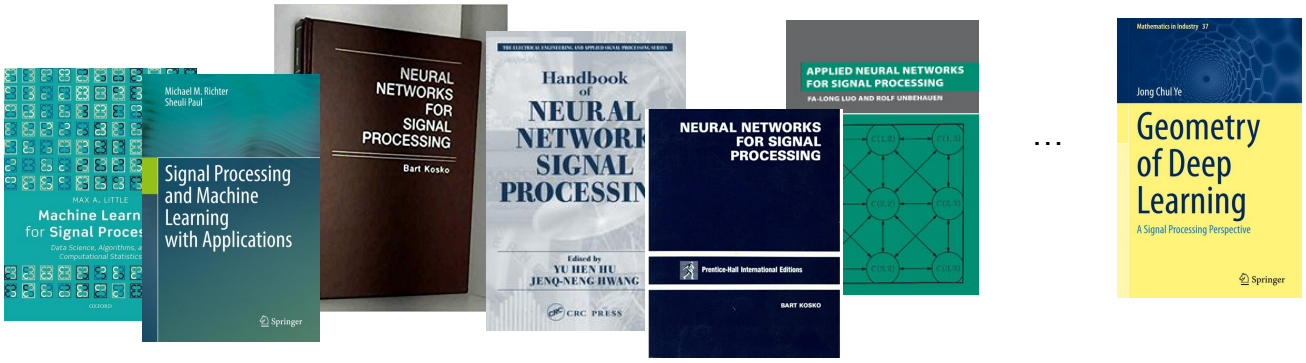
- Introduction
  - The (deep) learning (r)evolution in signal processing
  - Artificial neurons
  - Neural networks architectures for image processing
- Basic Components of CNNs
  - Operator-based formalism
  - Composition properties
  - Pooling
  - Continuity properties
- CNNs in Practice
  - Deep learning pipeline
  - Denoising
  - Segmentation

The (deep) learning (r)evolution in image processing

Special issues

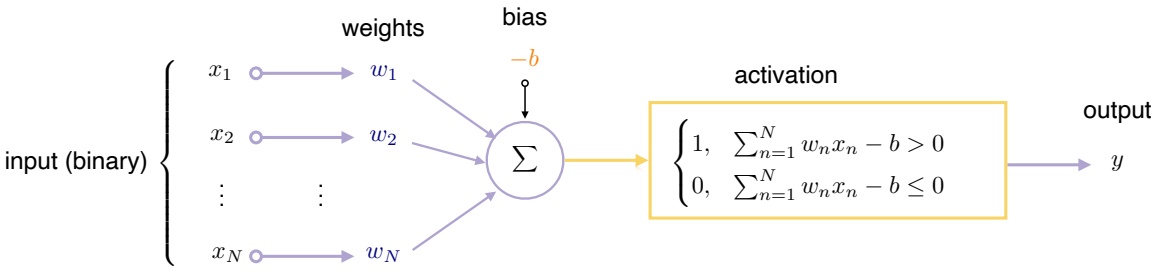
...

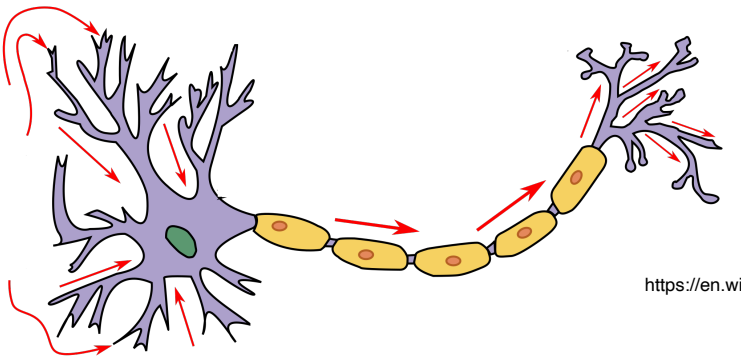
Flurry of new textbooks on neural networks



...

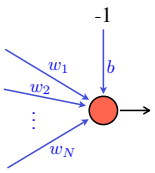
Formal model of neuron (McCulloch & Pitt)





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

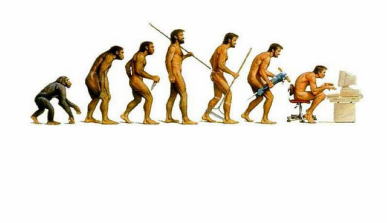
Artificial neurons



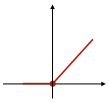
Definition: An artificial neuron with weights $\mathbf{w} = (w_1, \ldots, w_N) \in \mathbb{R}^N$, bias $b \in \mathbb{R}$ and **activation function** $\sigma : \mathbb{R} \to \mathbb{R}$ is defined as the function $f : \mathbb{R}^N \to \mathbb{R}$

$$f(\boldsymbol{x}) = \sigma\left(\mathbf{w}^\mathsf{T}\boldsymbol{x} - b\right) = \sigma\left(\sum_{n=1}^{N} w_n x_n - b\right).$$

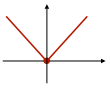
- Examples of activation functions
  - Threshold Logic Unit (Heaviside): $\mathrm{TLU}(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$ (McCullogh & Pitt 1943; Rosenblatt 1957)
  - Sigmoid function: $\sigma(x) = \dfrac{1}{1+\mathrm{e}^{-x}}$ (Rumelhart 1986, …)

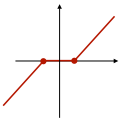


  - Rectified Linear Unit: $\mathrm{ReLU}(x) = x_+ = \max(0, x)$



  - And variants .....



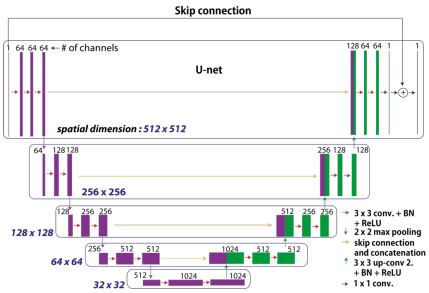


Neural network architectures for image processing

Neural networks are constructed from the **composition of basic modules** that can be chained at will.

Convolutional neural networks (CNN), in particular, are inspired by the structure of the primary visual cortex. They have an architecture that is well suited for image processing.

- Basic modules
  - Multi-channel convolution operators (filters)
  - Pointwise nonlinearities
  - Pooling: linear combination, flattening, sub-sampling, . . .



Some modules—in particular, the filters—are adjustable.
The parameters of the CNN (weights) are set during the training procedure.

- Training (not covered in this chapter)
  - Requires a comprehensive collection of reference input-output pairs. The larger the training set, the better!
  - Formulated as a large-scale optimization problem
  - Solved using some form of stochastic gradient algorithm (ADAM)
  - Requires a lot of computational ressources (GPU)

Basic Components of CNNs

- Operator-based formalism
- Composition
- Pooling
- Continuity and stability estimates

Operator-based formalism

Generic operator $\mathrm{T} : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}$ and $\mathcal{Y}$ are complete normed vector spaces.

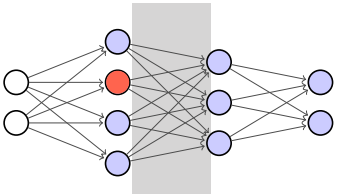
$$\forall x \in \mathcal{X} : \qquad y = \mathrm{T}\{x\} \in \mathcal{Y}$$

- General multivariate nonlinear operator (for image patches)

  $\mathrm{T} : \mathbb{R}^M \to \mathbb{R}^N$

- Pointwise nonlinearity = activation function of neuron

  $\mathrm{T}_{\mathrm{acti}} : \mathbb{R} \to \mathbb{R}$ with $\mathrm{T}_{\mathrm{acti}}\{x\} = \sigma(x + b), \quad b \in \mathbb{R}$



- General multivariate linear operator (for fully connected layer)

  $\mathrm{T}_{\mathrm{lin}} : \mathbb{R}^M \to \mathbb{R}^N$ with $\mathrm{T}_{\mathrm{lin}}\{\mathbf{x}\} = \mathbf{W}\mathbf{x}$ where $\mathbf{W} \in \mathbb{R}^{N \times M}$

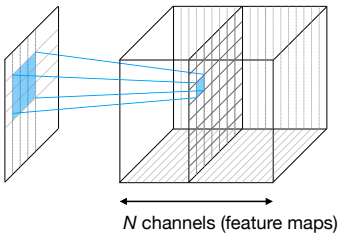
- Image-to-image operator

  $\mathrm{T} : \ell_2(\mathbb{Z}^d) \to \ell_2(\mathbb{Z}^d)$

  Most operators of interest are **shift-invariant**

  Example: $\mathrm{T}_{\mathrm{LSI}}\{f\} = h * f$ with $h = \mathrm{T}_{\mathrm{LSI}}\{\delta[\cdot]\}$ (discrete convolution)

Convolutional layer



- Patch extraction operator: $\ell_2(\mathbb{Z}^d) \times \mathbb{Z}^d \to \mathbb{R}^M$ with $M = \#W$

  $$\mathrm{Vect}(f[\cdot], \boldsymbol{k}) = \boldsymbol{f}_W[\boldsymbol{k}] = (f[\boldsymbol{k} - \boldsymbol{k}_0])_{\boldsymbol{k}_0 \in W}$$

- Convolution layer : $\ell_2(\mathbb{Z}^d) \to \ell_2^N(\mathbb{Z}^d)$ with N channels = feature maps

  Shared operator $\mathrm{T}_{\mathrm{patch}} : \mathbb{R}^M \to \mathbb{R}^N$

  convolution masks — pointwise nonlinearities

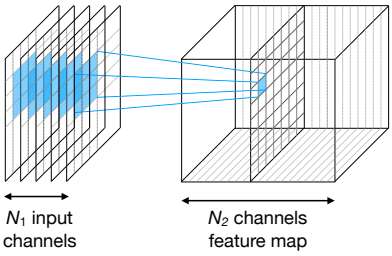
  $$\mathrm{T}_{\mathrm{patch}}(\boldsymbol{f}_W[\boldsymbol{k}]) = \boldsymbol{\sigma}\big(\mathbf{W}\boldsymbol{f}_W[\boldsymbol{k}]\big) \quad \text{where} \quad \mathbf{W} = \begin{pmatrix} \mathbf{w}_1^{\mathsf{T}} \\ \vdots \\ \mathbf{w}_N^{\mathsf{T}} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\sigma} = \begin{pmatrix} \sigma_1 : \mathbb{R} \to \mathbb{R} \\ \vdots \\ \sigma_N : \mathbb{R} \to \mathbb{R} \end{pmatrix}$$

  Typically: $\sigma_n(x) = \mathrm{ReLU}(x - b_n)$

  Implementation of N channel convolution layer:

  $$\begin{pmatrix} \sigma_1\Big((h_1 * f)[\boldsymbol{k}]\Big) \\ \vdots \\ \sigma_N\Big((h_N * f)[\boldsymbol{k}]\Big) \end{pmatrix} = \mathrm{T}_{\mathrm{patch}}(\boldsymbol{f}_W[\boldsymbol{k}]) = \boldsymbol{\sigma}\Big(\mathbf{W}\boldsymbol{f}_W[\boldsymbol{k}]\Big)$$

Vector-valued convolutional layer (tensor)



Input feature map: $\boldsymbol{f}[\cdot] = (f_1[\cdot], \ldots, f_{N_1}[\cdot])$

- Tensor patch extraction: $\ell_2^{N_1}(\mathbb{Z}^d) \times \mathbb{Z}^d \to \mathbb{R}^{M \times N_1}$ with $M = \#W$

  $$\mathrm{Tensor}(\boldsymbol{f}[\cdot], \boldsymbol{k}) = \boldsymbol{F}[\boldsymbol{k}] = (f_i[\boldsymbol{k} - \boldsymbol{k}_0])_{\boldsymbol{k}_0 \in W,\ i \in \{1,\ldots,N_1\}}$$

- Vector-valued convolution layer : $\ell_2^{N_1}(\mathbb{Z}^d) \to \ell_2^{N_2}(\mathbb{Z}^d)$

  Shared operator $\mathrm{T}_{\mathrm{tensor}} : \mathbb{R}^{M \times N_1} \to \mathbb{R}^{N_2}$

  convolution **tensors** — pointwise nonlinearities

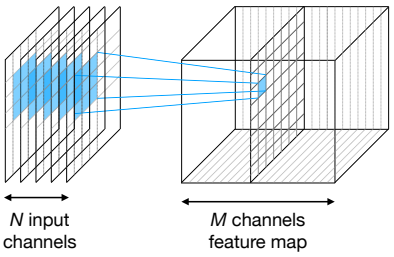
  $$\mathrm{T}_{\mathrm{tensor}}(\boldsymbol{F}[\boldsymbol{k}]) = \boldsymbol{\sigma}\big(\mathbf{W}\boldsymbol{F}[\boldsymbol{k}]\big) \quad \text{where} \quad \mathbf{W} = \begin{pmatrix} \mathbf{w}_1^{\mathsf{T}} \\ \vdots \\ \mathbf{w}_{N_2}^{\mathsf{T}} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\sigma} = \begin{pmatrix} \sigma_1 : \mathbb{R} \to \mathbb{R} \\ \vdots \\ \sigma_{N_2} : \mathbb{R} \to \mathbb{R} \end{pmatrix}$$

  $$= \boldsymbol{\sigma}\Big((\mathbf{H} * \boldsymbol{f})[\boldsymbol{k}]\Big) \quad \text{with} \quad \mathbf{H}[\cdot] = N_2 \times N_1 \text{ array of filters}$$

Convolution of vector-valued images

Input feature map: $\boldsymbol{f}[\cdot] = \begin{pmatrix} f_1[\cdot] \\ \vdots \\ f_N[\cdot] \end{pmatrix}$

Matrix-valued filter: $\mathbf{H}[\cdot] = \begin{pmatrix} h_{1,1}[\cdot] & h_{1,2}[\cdot] & \dots & h_{1,N}[\cdot] \\ \vdots & \vdots & & \vdots \\ h_{M,1}[\cdot] & h_{1,2}[\cdot] & \dots & h_{M,N} \end{pmatrix}$



- Vector-valued filterbank: $\ell_2^N(\mathbb{Z}^d) \to \ell_2^M(\mathbb{Z}^d)$

$$(\mathbf{H} * \boldsymbol{f})[\cdot] = \begin{pmatrix} (h_{1,1} * f_1)[\cdot] + (h_{1,2} * f_2)[\cdot] + \cdots + (h_{1,N} * f_N)[\cdot] \\ \vdots \\ (h_{M,1} * f_1)[\cdot] + (h_{M,2} * f_2)[\cdot] + \cdots + (h_{M,N} * f_N)[\cdot] \end{pmatrix}$$

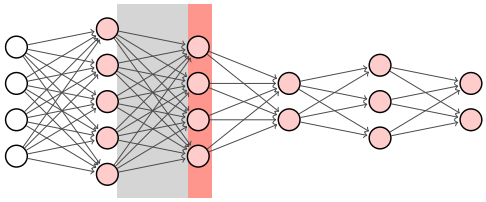
Composition

Series of (nonlinear) operators $\mathrm{T}_\ell : \mathcal{X}_\ell \to \mathcal{Y}_\ell$ where $\mathcal{X}_\ell$ and $\mathcal{Y}_\ell$ are complete normed vector spaces.

Hypothesis: T_ℓ and $\mathrm{T}_{\ell-1}$ have compatible domain and range; i.e., $\mathcal{X}_\ell = \mathcal{Y}_{\ell-1}$

- Composition of $\mathrm{T}_1 : \mathcal{X}_1 \to \mathcal{X}_2$ and $\mathrm{T}_2 : \mathcal{X}_2 \to \mathcal{X}_3$

  $\mathrm{T}_2 \circ \mathrm{T}_1 : \mathcal{X}_1 \to \mathcal{X}_3$ with $\mathrm{T}_2 \circ \mathrm{T}_1\{x\} = \mathrm{T}_2\{\mathrm{T}_1\{x\}\}$



- Deep neural network

  $\mathbf{f}_{\mathrm{deep}}(\boldsymbol{x}) = (\boldsymbol{\sigma}_L \circ \mathrm{A}_L \circ \boldsymbol{\sigma}_{L-1} \circ \cdots \circ \boldsymbol{\sigma}_2 \circ \mathrm{A}_2 \circ \boldsymbol{\sigma}_1 \circ \mathrm{A}_1)(\boldsymbol{x})$

  - Affine layers $\mathrm{A}_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$

    $\mathrm{A}_\ell(\boldsymbol{x}) = \mathbf{W}_\ell \boldsymbol{x} + \mathbf{b}_\ell$ with **trainable** weight matrix $\mathbf{W}_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and bias $\mathbf{b}_\ell \in \mathbb{R}^{N_\ell}$

  - Pointwise nonlinearities $\boldsymbol{\sigma}_\ell : \mathbb{R}^{N_\ell} \to \mathbb{R}^{N_\ell}$

    $\boldsymbol{\sigma}_\ell(\boldsymbol{x}) = \big(\sigma(x_1), \dots, \sigma(x_{N_\ell})\big)$ with common activation fonction $\sigma : \mathbb{R} \to \mathbb{R}$

Composition: Properties

- Preservation of linearity (affiness)

  $\mathrm{A}_1 : \mathbf{x} \mapsto \mathbf{W}_1\mathbf{x} + \mathbf{b}_1$ and $\mathrm{A}_2 : \mathbf{y} \mapsto \mathbf{W}_2\mathbf{y} + \mathbf{b}_2 \quad \Rightarrow \quad \mathrm{A}_2 \circ \mathrm{A}_1 : \mathbf{x} \mapsto (\mathbf{W}_2\mathbf{W}_1)\mathbf{x} + (\mathbf{b}_2 + \mathbf{W}_2\mathbf{b}_1)$

- Preservation of convolutional structure

  T_1 and T_2 are LSI with impulse responses $h_1, h_2 \in \ell_1(\mathbb{Z}^d)$

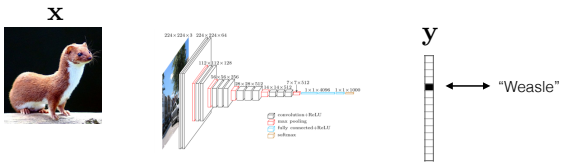
  $\Rightarrow \quad \mathrm{T}_2 \circ \mathrm{T}_1 : f \mapsto h * f \quad$ with $\quad h = h_2 * h_1 \in \ell_1(\mathbb{Z}^d)$

  Application: Construction of larger receptive fields

- Preservation of continuity

  The composition of two continuous functions is continuous

Pooling



- Down-sampling: $\downarrow_m\{f\}[\boldsymbol{k}] = f[m\boldsymbol{k}]$

- Max pooling: $\mathbb{R}^N \to \mathbb{R}$

  $\boldsymbol{u} \mapsto \max(\boldsymbol{u}) = \max(u_1, \ldots, u_N)$

- Softmax: $\mathbb{R}^N \to \mathbb{R}^N$ (transforms output of CNN in "probabilities")

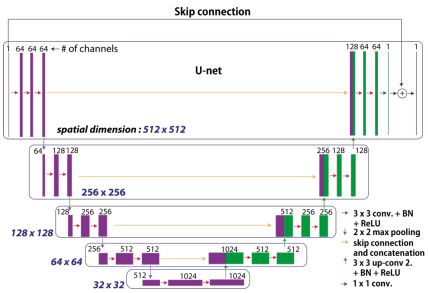
  $$\mathbf{u} \mapsto \left(p_k = \frac{\exp(u_k)}{\sum_{n=1}^{N} \exp(u_n)}\right)$$

- Up-sampling

  $$\uparrow_m\{f\}[\boldsymbol{k}] = \begin{cases} f[\boldsymbol{n}], & \boldsymbol{k} = m\boldsymbol{n} \\ 0, & \text{otherwise} \end{cases}$$

- Up-sampling with repetition

  $\uparrow_m\{f\}[\boldsymbol{k}] = f[\boldsymbol{k}/m]$



U-net

Continuity requirements

Generic operator $\mathrm{T} : \mathcal{X} \to \mathcal{Y}$ where $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ and $(\mathcal{Y}, \|\cdot\|_{\mathcal{Y}})$ are complete normed vector spaces.

Definition

The operator $\mathrm{T} : \mathcal{X} \to \mathcal{Y}$ is **continuous** if $\lim_i \mathrm{T}\{x_i\} = \mathrm{T}\{\lim_i x_i\} = \mathrm{T}\{x\}$ for any sequence $(x_i)_{i\in\mathbb{N}}$ that is converging in $\mathcal{X}$ with $\lim_i x_i = x$

Definition

The operator $\mathrm{T} : \mathcal{X} \to \mathcal{Y}$ is **Lipschitz continuous** if there exists a constant $L > 0$ such that $\|\mathrm{T}\{x_1\} - \mathrm{T}\{x_2\}\|_{\mathcal{Y}} \leq L\|x_1 - x_2\|_{\mathcal{X}}$ for any $x_1, x_2 \in \mathcal{X}$.

- Lipschitz constant

  $\mathrm{Lip}(\mathrm{T}) = L$ where L is the smallest constant such that the Lipschitz inequality holds.

  To avoid instabilities, **all modules** of a CNN should be **Lipschitz continuous**.
  This implies that they are a.e. **differentiable**, which is desirable for training with backpropagation.

Counterexample: TLU networks are discontinuous, and therefore very hard to train unless the architecture is shallow (perceptron).

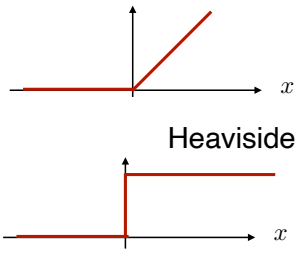
Lipschitz constant of primary modules

- Pointwise nonlinearity

  $\sigma : \mathbb{R} \to \mathbb{R}$ where σ is differentiable

  $$\mathrm{Lip}(\sigma) = \sup_{x\in\mathbb{R}} \left|\frac{\mathrm{d}\sigma(x)}{\mathrm{d}x}\right| = \|\sigma'\|_{L_\infty}$$ (cf. Mean Value Theorem)

  Example: $\mathrm{Lip}(\mathrm{ReLU}) = \sup_{x\in\mathbb{R}} |u(x)| = 1$



- LSI operator (convolution channel)

  $\mathrm{T}_{\mathrm{LSI}}(f) = h * f$ where $h \in \ell_1(\mathbb{Z}^d)$

  $$\mathrm{Lip}(\mathrm{T}_{\mathrm{LSI}}) = H_{\max} = \sup_{\boldsymbol{\omega}\in[0,\pi]^d} \left|H(\mathrm{e}^{\mathrm{j}\boldsymbol{\omega}})\right| \leq \|h\|_{\ell_1}$$

Justification (Parseval)

$$\|h * f - h * g\|_{\ell_2}^2 = \|h * (f - g)\|_{\ell_2}^2 = \frac{1}{(2\pi)^d} \int_{[0,\pi]^d} |H(\mathrm{e}^{\mathrm{j}\boldsymbol{\omega}})|^2 \left|F(\mathrm{e}^{\mathrm{j}\boldsymbol{\omega}}) - G(\mathrm{e}^{\mathrm{j}\boldsymbol{\omega}})\right|^2 \mathrm{d}\boldsymbol{\omega}$$
$$\leq \frac{H_{\max}^2}{(2\pi)^d} \int_{[0,\pi]^d} \left|F(\mathrm{e}^{\mathrm{j}\boldsymbol{\omega}}) - G(\mathrm{e}^{\mathrm{j}\boldsymbol{\omega}})\right|^2 \mathrm{d}\boldsymbol{\omega} = H_{\max}^2 \|f - g\|_{\ell_2}^2$$

Lipschitz constant of primary modules (cont'd)

- Linear (resp. affine) transform

$\mathrm{T}_{\mathrm{lin}} : \mathbb{R}^M \to \mathbb{R}^N$ with $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$ (linear)

or $\mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{b}$ (affine) where $\mathbf{A} \in \mathbb{R}^{M \times N}, \mathbf{b} \in \mathbb{R}^M$

$$\mathrm{Lip}(\mathrm{T}_{\mathrm{lin}}) = \sup_{\|\mathbf{x}\|_2 \leq 1} \|\mathbf{A}\mathbf{x}\|_2 = \rho(\mathbf{A})$$ (spectral norm = largest singular value of $\mathbf{A}$)

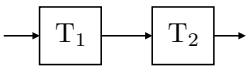
- Imposing Lip-1 layer by spectral normalization

$$\mathrm{T}_{\mathrm{normal}}\{\mathbf{x}\} = \frac{1}{\rho(\mathbf{A})}\mathbf{A}\mathbf{x}$$

> Estimation by power method: For $k = 0, \ldots, K$
>
> $$\mathbf{u}_k = \frac{1}{\|\mathbf{A}^\mathsf{T}(\mathbf{A}\mathbf{u}_k)\|_2}\mathbf{A}^\mathsf{T}(\mathbf{A}\mathbf{u}_k)$$
>
> Upon convergence, $\mathbf{u} = \lim_k \mathbf{u}_k$ is the dominant eigenvector of $\mathbf{A}^\mathsf{T}\mathbf{A}$.
>
> Finally, $\rho(\mathbf{A}) = \sqrt{(\mathbf{A}\mathbf{u})^\mathsf{T}(\mathbf{A}\mathbf{u})}$

Stability and combination of modules

- Composition



$$\mathrm{Lip}(\mathrm{T}_1) = L_1 \ \& \ \mathrm{Lip}(\mathrm{T}_2) = L_2 \quad \Rightarrow \quad \mathrm{Lip}(\mathrm{T}_2 \circ \mathrm{T}_1) \leq L_2 L_1$$

$$\forall f, g \in \mathcal{X}_1 : \|\mathrm{T}_2 \circ \mathrm{T}_1\{f\} - \mathrm{T}_2 \circ \mathrm{T}_1\{g\}\|_{\mathcal{Y}_2} \leq L_2 \|\mathrm{T}_1\{f\} - \mathrm{T}_1\{g\}\|_{\mathcal{Y}_1 = \mathcal{X}_2} \leq L_2 L_1 \|f - g\|_{\mathcal{X}_1}$$

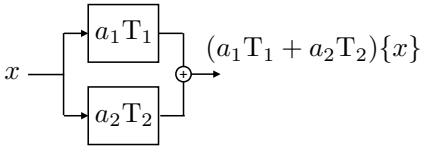
Deep neural network with Lip-1 activations (e.g., ReLU) :

$$\mathbf{f}_{\mathrm{deep}}(\boldsymbol{x}) = (\boldsymbol{\sigma}_L \circ \mathrm{A}_L \circ \boldsymbol{\sigma}_{L-1} \circ \cdots \circ \boldsymbol{\sigma}_2 \circ \mathrm{A}_2 \circ \boldsymbol{\sigma}_1 \circ \mathrm{A}_1)(\boldsymbol{x}) \quad \Rightarrow \quad \mathrm{Lip}(\mathbf{f}_{\mathrm{deep}}) = \prod_{\ell=1}^{L} \left(\overbrace{\mathrm{Lip}(\boldsymbol{\sigma}_\ell)}^{1} \mathrm{Lip}(\mathrm{A}_\ell) \right)$$

➡ Deeper networks tend to be less stable

- Linear combination

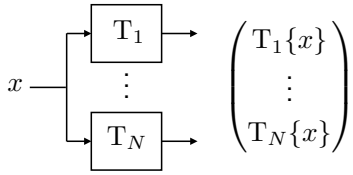
$$\mathrm{Lip}\Big(\sum_{i=1}^{I} a_i \mathrm{T}_i\Big) \leq \sum_{i=1}^{I} |a_i| \mathrm{Lip}(\mathrm{T}_i)$$ (by triangle inequality)



- Parallel feature maps

$$\mathrm{T} = \begin{pmatrix} \mathrm{T}_1 \\ \vdots \\ \mathrm{T}_N \end{pmatrix} : \ell_2(\mathbb{Z}^d) \to \ell_2^N(\mathbb{Z}^d)$$

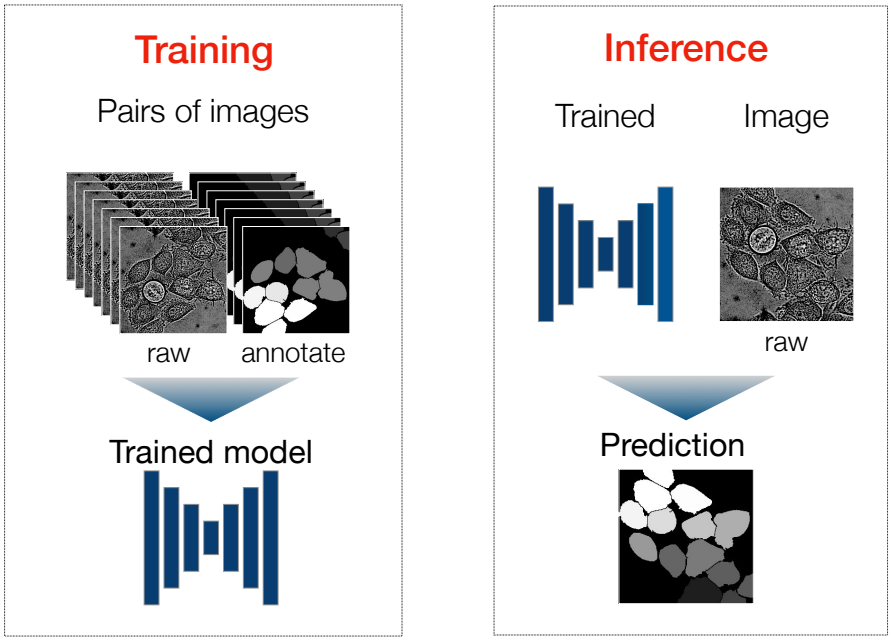
$$\mathrm{Lip}(\mathrm{T}) \leq \sqrt{L_1^2 + \ldots L_N^2} \leq L_1 + \cdots + L_N$$



CNNs in Practice

- Deep learning pipeline
- Denoising
- Segmentation

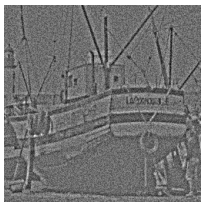
Deep Learning Pipeline



Denoising: “Handcrafted” ancestor of Resnet

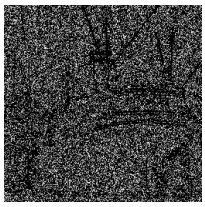
Ground-truth

LP



HP

noise estimator

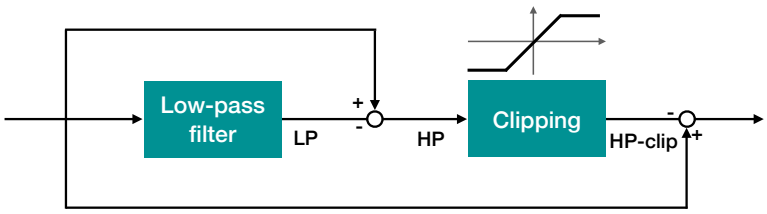


HP-clip

	Input	LP	Denoising
RMSE	17.30	18.19	12.08
PSNR	21.89 dB	21.45 dB	25.01dB

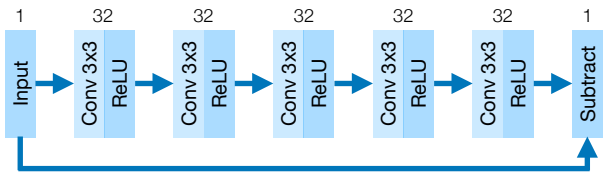
with respect to ground truth

Input
noisy image



Output
denoised image

Deep CNN for residual image denoisng



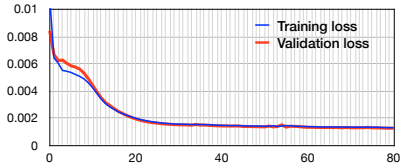
DnCNN architecture (residual)

Datasets

- 80 natural graylevel images 384×384, artificially degraded by adding noise

Training CNN

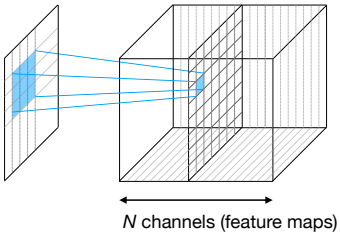
- 20 images (25%) in validation set
- **37’601 parameters** (weights and biais)
- Loss function is the MSE
- 80 epochs, batch_size = 16, lr = 0.001
- Global normalization 0/1



Layer	Shape	Parameters
Input	384 x 384 x 1	0
Conv2D 3x3 + b, ReLU	384 x 384 x 32	320
Conv2D 3x3 + b, ReLU	384 x 384 x 32	9’248
Conv2D 3x3 + b, ReLU	384 x 384 x 32	9’248
Conv2D 3x3 + b, ReLU	384 x 384 x 32	9’248
Conv2D 3x3 + b, ReLU	384 x 384 x 32	9’248
Conv2D 3x3 + b, ReLU	384 x 384 x 1	289
Subtract	384 x 384 x 1	0
Total		**37’601**

DnCNN architecture

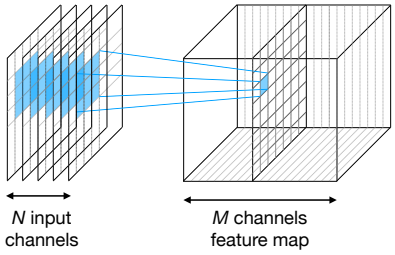
First conv layer:



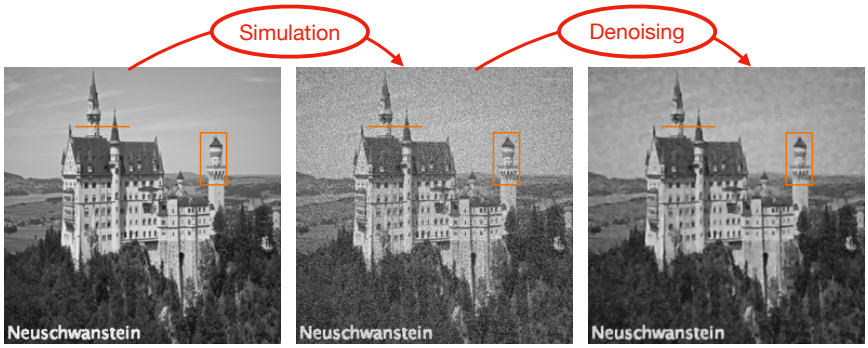
Number of learnable parameters: $(3 \times 3) + 1$ (bias) per output channel

Layer	Shape	Parameters
Input	384 x 384 x 1	0
Conv2D 3x3 + b, ReLU	384 x 384 x 32	320
Conv2D 3x3 + b, ReLU	384 x 384 x 32	9248
Conv2D 3x3 + b, ReLU	384 x 384 x 32	9248
Conv2D 3x3 + b, ReLU	384 x 384 x 32	9248
Conv2D 3x3 + b, ReLU	384 x 384 x 32	9248
Conv2D 3x3 + b, ReLU	384 x 384 x 1	289
Subtract	384 x 384 x 1	0
Total		**37601**

Internal conv layers:

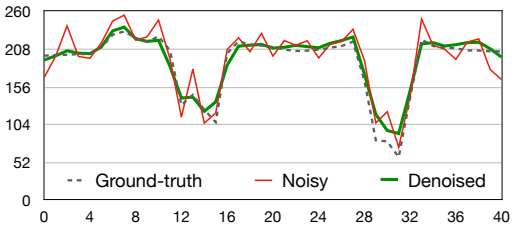


Number of learnable parameters: $(3 \times 3) \times 32 + 1$ (bias) per output channel



Clean image | Noisy image | Prediction

Gaussian noise $\sigma = 20$



Clean image | Noisy image | Prediction | Gaussian Filter | Median Filter | HighPass+ SoftClipping

Clean image
Ground-truth

Noisy image 15.36 dB
Additive Gaussian noise

High-pass filter
Soft Clipping
19.97 dB

Resnet (100 epochs)
3 layers, 16 channels
20.91 dB

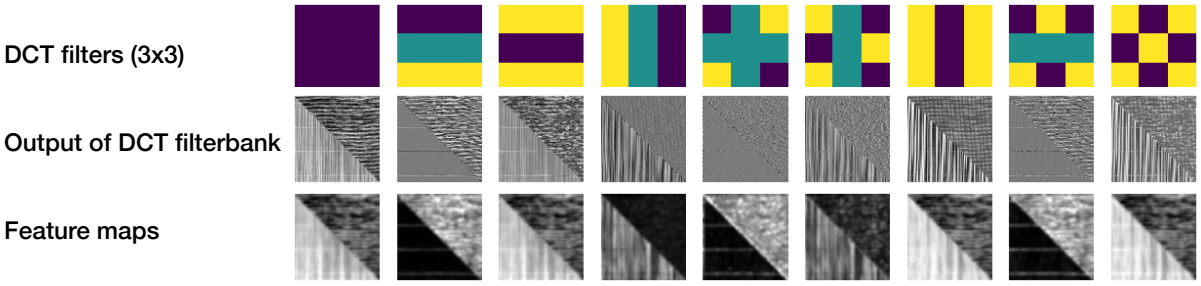
Gaussian filter
$\sigma = 1$
18.23 dB

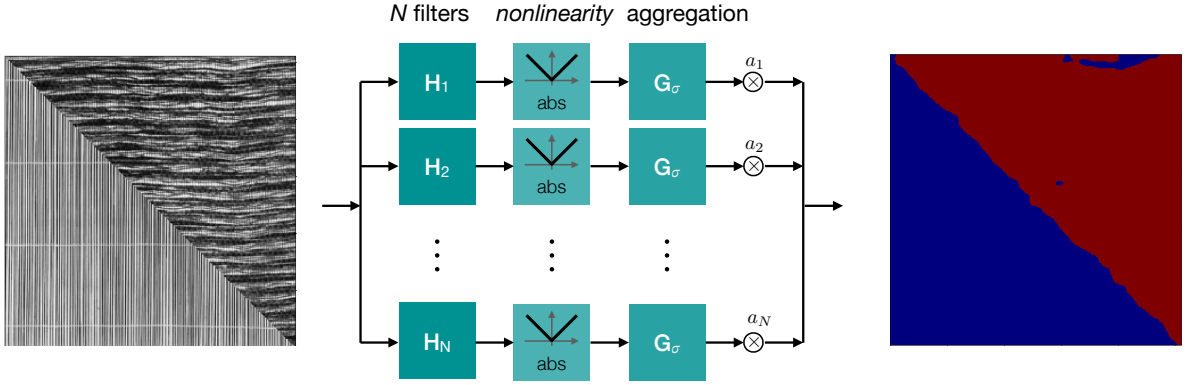
Median filter
radius = 3
18.45 dB

Resnet (200 epochs)
5 layers, 32 channels
21.01 dB

Unet (100 epochs)
3 Pooling steps, 32 channels
20.99 dB

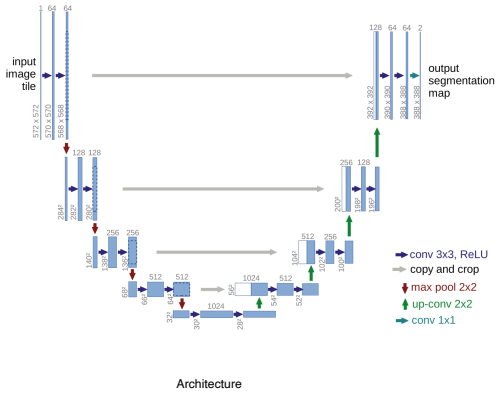
Segmentation: "Handcrafted" texture discriminator



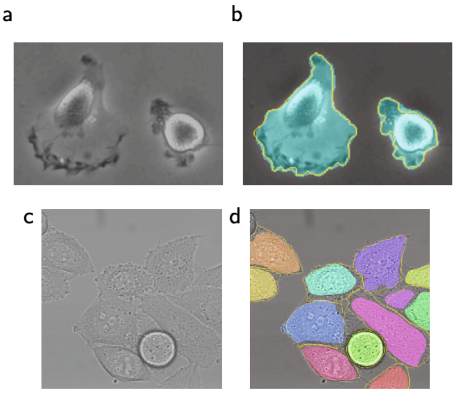


Popular CNN architecture for image segmentation

U-net introduced by Ronneberger in 2015 for biomedical image segmentation

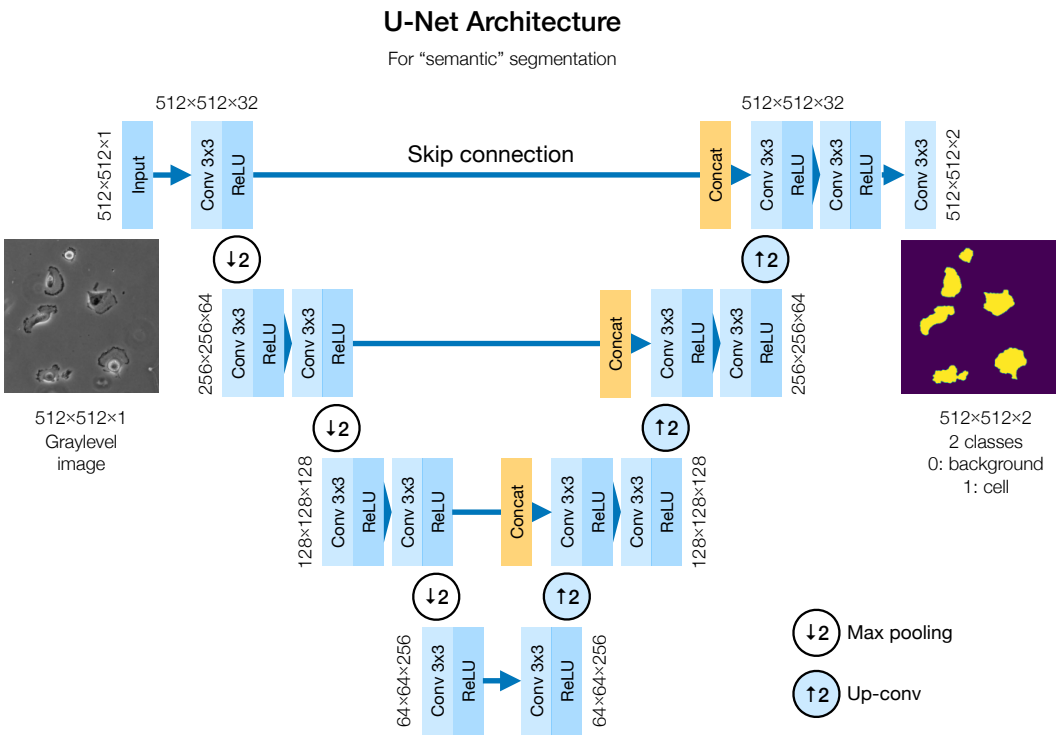


Architecture

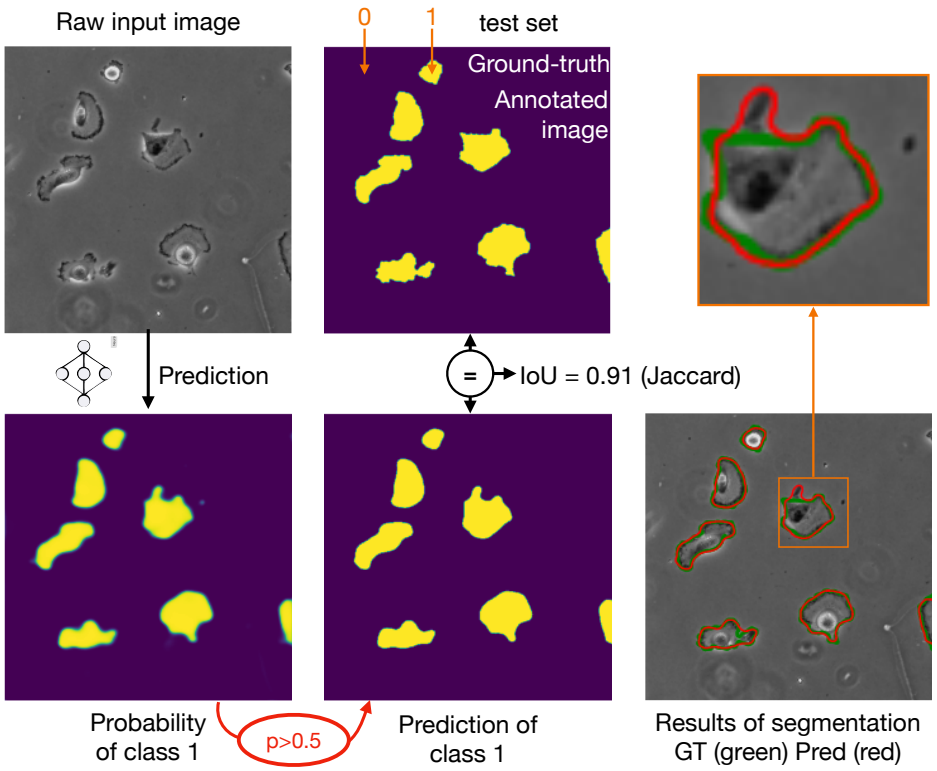


Winner of ISBI 2015 cell tracking challenge

1. https://arxiv.org/abs/1505.04597



Layer	Shape	Params
Input	512x512x1	0
Conv2D 3x3	512x512x32	320
Conv2D 3x3, Activation	512x512x32	9248
MaxPooling2D	256x256x32	0
Conv2D 3x3, Activation	256x256x64	18496
Conv2D 3x3, Activation	256x256x64	36928
MaxPooling2D	128x128x64	0
Conv2D 3x3, Activation	128x128x128	73856
Conv2D 3x3, Activation	128x128x128	147584
MaxPooling2D	64x64x128	0
Conv2D 3x3, Activation	64x64x256	295168
Conv2D 3x3, Activation	64x64x256	590080
Up-conv	128x128x128	131200
Concatenate	128x128x256	0
Conv2D 3x3, Activation	128x128x128	295040
Conv2D 3x3, Activation	128x128x128	147584
Up-conv	256x256x64	32832
Concatenate	256x256x12	0
Conv2D 3x3, Activation	256x256x64	73792
Activation	256x256x64	0
Conv2D 3x3, Activation	256x256x64	36928
Up-conv	512x512x32	8224
Concatenate	512x512x64	0
Conv2D 3x3, Activation	512x512x32	18464
Conv2D 3x3, Activation	512x512x32	9248
Conv2D 3x3	512x512x2	66
Total		**1'925'058**

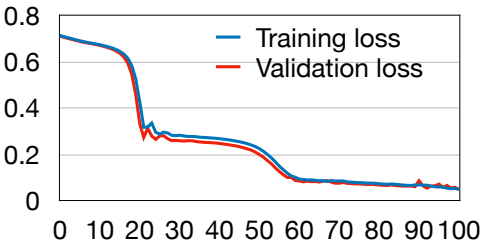


Datasets

- 66 graylevel images (1 channel)
- 66 manually annotated images, 2 classes

Training CNN

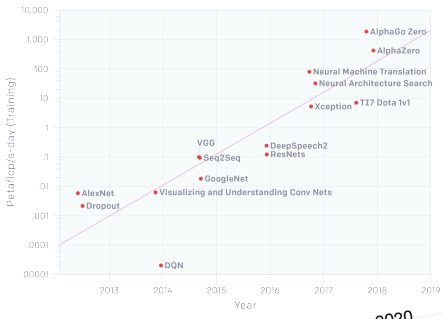
- 66 annotated images / 25% for validation
- **1'925'058 parameters** (weights and bias)
- Loss function is the binary cross entropy
- 100 epochs, batch_size = 16, lr = 0.001



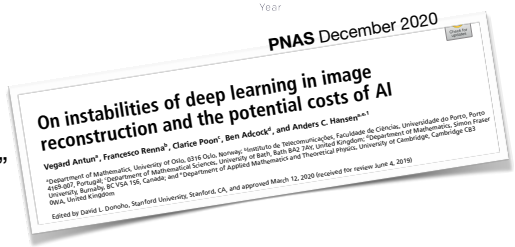
CONCLUSION: The Pros and Cons of CNNs

- Advantages of (deep) CNNs
  - Are sufficiently flexible to implement most image-processing tasks
  - Can benefit from hardware acceleration (GPU)
  - Performance of **deep CNNs** is often **spectacular**
  - Once trained, they are very fast to deploy (inference)

AI Moore's law: doubling every 3.4 month !



- Downsides of CNNs
  - Require **huge amounts** of **data** and **computation** for training
  - While the individual components are simple, the global behaviour is poorly understood ⇒ **lack of guarantees**
  - Training and fine-tuning is fastidious "*Graduate-student descent* …."
  - No free lunch: Trained CNNs are very task/data specific
  - Can **behave erratically** — lack of robustness, subject to adversarial attacks



- How the new trend benefits from the techniques of "traditional" IP
  - **Deeper understanding** of modules
  - Suggestion of **leaner** and **more robust** architectures