Towards Deeper Understanding of Deep Learning

Stéphane Ayache (with credits by Thierry Artières)

Equipe QARMA d'apprentissage automatique
Pole Science des Données - LIS lab
Ecole Centrale Marseille, Polytech Marseille

June 26, 2019

Outline

1. Intro
   - Context
   - Breakthrough
   - Explainability
2. MLPs
3. Architectures
4. Understanding Convolution
5. Conclusion

History of Neural Networks

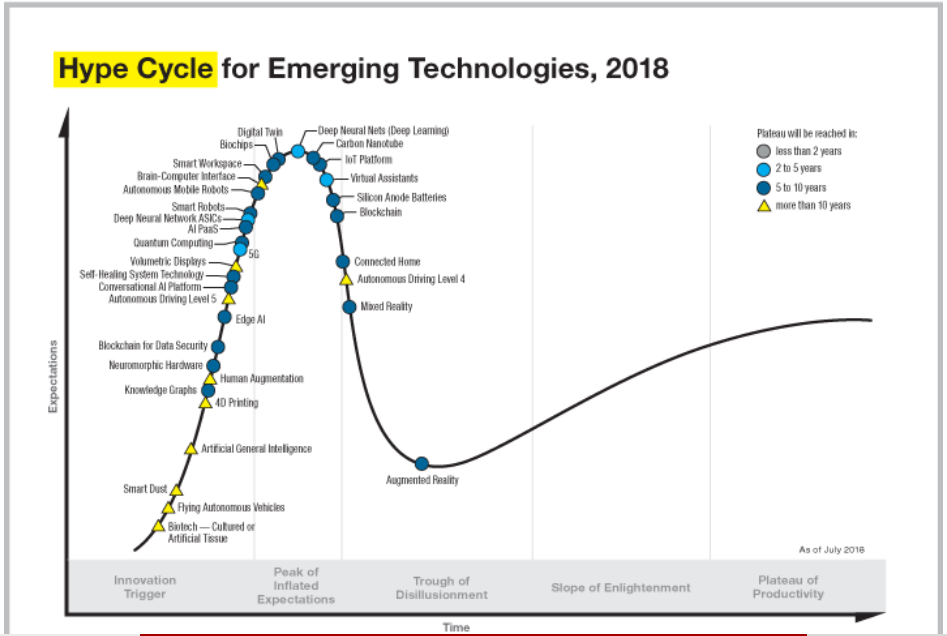
Key dates

- 1943 : Formal neuron [McCuloch-Pitts]
- 1950 : Oragnization of neurons and learning rules [Hebb]
- 1960 : Perceptron [Rosenblatt]
- 1960 : Update rule [Widrow Hoff]
- 1969 : Limitations of the Perceptron [Minsky]
- 1980s : Back-propagation [Rumelhart and Hinton]
- 1990s : Convolutional Networks [LeCun and al.]
- 1990s: Long Short Term Memory networks [Hochreiter and Schmidhuber]
- 2006 : Paper on Deep Learning in Nature [Hinton and al.]
- 2012 : Imagenet Challenge Win [Krizhevsky, Sutskever, and Hinton]
- 2013 : First edition of ICLR
- 2013 : Memory networks [Weston and al.]
- 2014 : Adversarial Networks [Goodfelow and al.]
- 2014 : Google Net [Szegedy and al.]
- 2015 : Residual Networks [He et al.]
- ...

AI today

Where are we?

- The original AI (Dartmouth workshop in 1956) was General AI
- Today 60 years after Dartmouth
  - We have achieved impressive AI results in some fields
  - We can start thinking more seriously about "Thinking Machine"
  - These are just the premises



Deep Learning today

Spectaculary breakthroughs - fast industrial transfer

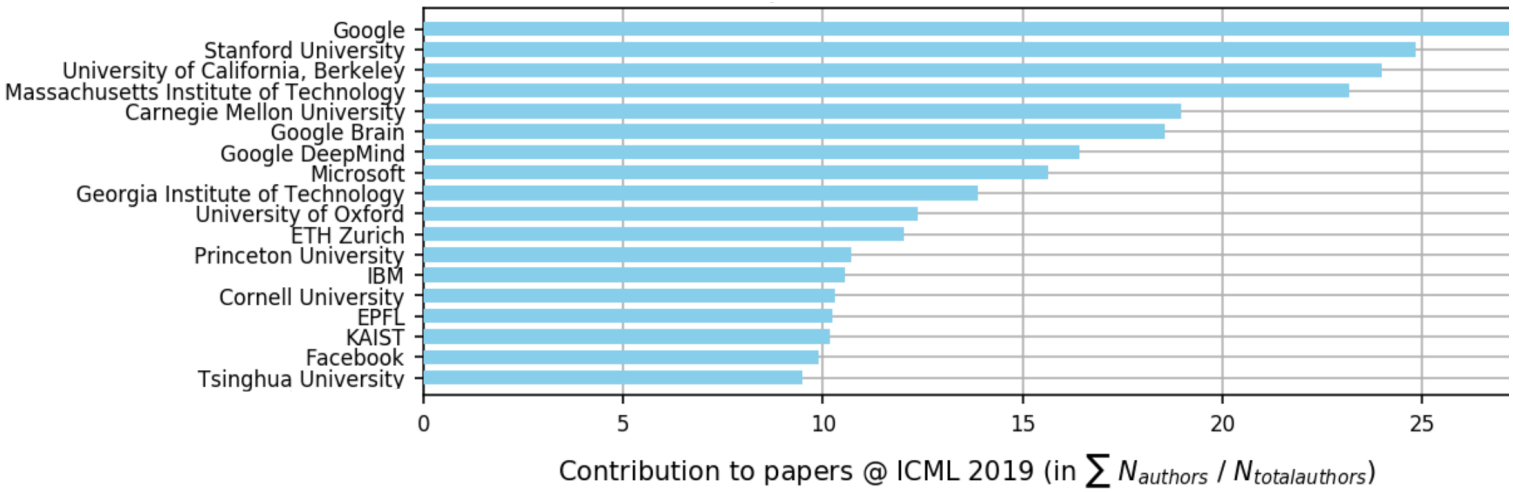
- Images, Videos, Audio, Speech, Texts
- Successful setting
  - Structured data (temporal, spatial...)
  - Huge volumes of datas
  - Huge models (millions of parameters)
  - Huge storage and computing resources (GPU, TPU)

	VGGNet	DeepVideo	GNMT
Used For	Identifying Image Category	Identifying Video Category	Translation
Input	Image	Video	English Text
Output	1000 Categories	47 Categories	French Text
Parameters	140M	~100M	380M
Data Size	1.2M Images with assigned Category	1.1M Videos with assigned Category	6M Sentence Pairs, 340M Words
Dataset	ILSVRC-2012	Sports-1M	WMT'14

Machine Learning and Deep Learning today

Topics, trends and who's who?

- Mix between academics and companies
- Extreme popularity of Deep Learning topics



Top contributors by affiliation @ ICML 2019

Machine Learning and Deep Learning today

Topics, trends and who's who?

- Mix between academics and companies
- Extreme popularity of Deep Learning topics

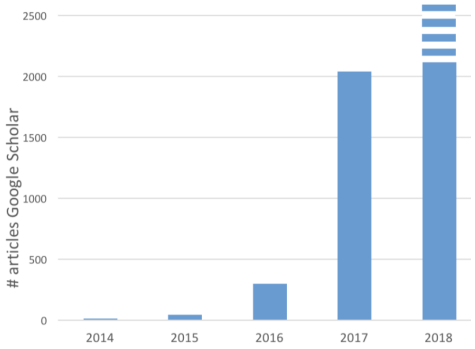
Rank	Day	Name	Marked	Like
1	1	Tutorials Hall A	2789	287
2	2	Deep Learning, Applications	2364	289
3	3	Deep Learning	1831	163
4	3	Reinforecment Learning, Deep Learning	1592	140
5	1	Optimization	1522	130
6	1	Tutorials Hall C	1344	135
7	1	Algorithms	1307	137
8	2	Theory	1288	83
9	2	Algorithms Optimization	1223	107
10	4	Deep Learning, Algorithms	1202	113
11	4	Deep Reinforcement Learning	1202	43
12	2	Invited talk: Kate Crawford: The Trouble with Bias	1162	71
13	3	Reinforcement Learning, Algorithms, Applications	1156	134
14	3	Invited talk: Pieter Abbeel: Deep Learning for Robotics	1087	61
15	1	Tutorials Grand Ballroom	1082	132

Most Popular Tracks @ NIPS 2017

DL research is going very fast !!

Example of an emerging topic: Generative Adversarial Networks

- First publication : 2014 by Ian J. Goodfellow, and al.
- Hundreds of publications (close to a thousand) papers since



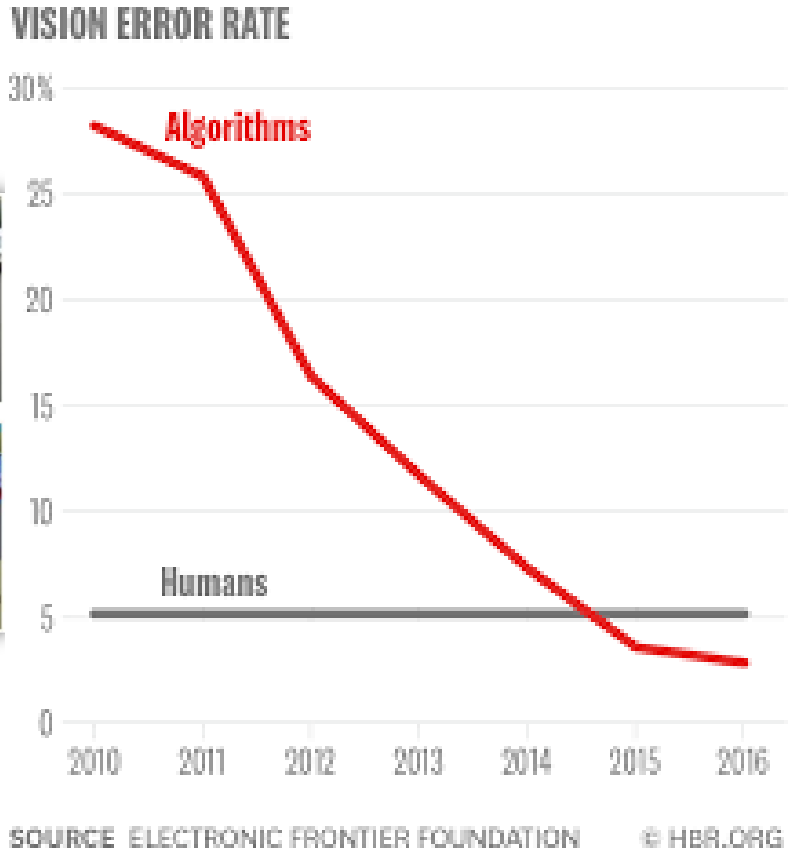
New publication mode

- Wasserstein GANs, Martin Arjovsky and al.
  - Published on arXiv : Jan 2017
  - Published at ICML in Aout 2017
- Improved Training of Wasserstein GANs, by Ishaan Gulrajani and al.
  - Published on arXiv : March 2017
  - Published at NIPS in December 2017
- Improving the Improved Training of Wasserstein GANs: A Consistency Term and Its Dual Effect, by Xiang Wei and al.
  - Published on Openreview : Oct 2017
  - Accepted as poster at ICLR in 2018 (April 2018)

Spectaculary breakthrough

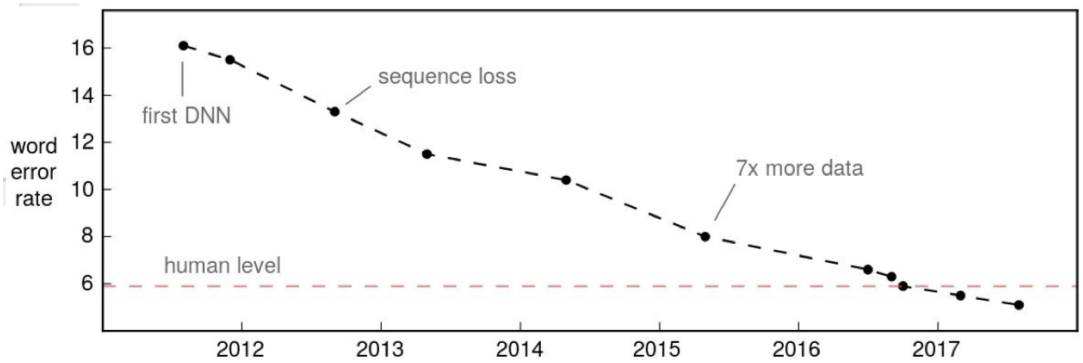
Computer vision

Real time Object recognition



Spectaculary breakthrough

Speech recognition



Improvements in word error rate over time on the Switchboard conversational speech recognition benchmark. The test set was collected in 2000. It consists of 40 phone conversations between two random native English speakers.

Source: https://awni.github.io/speech-recognition/

Spectaculary breakthrough

Games

- BackGammon, Chess, Go

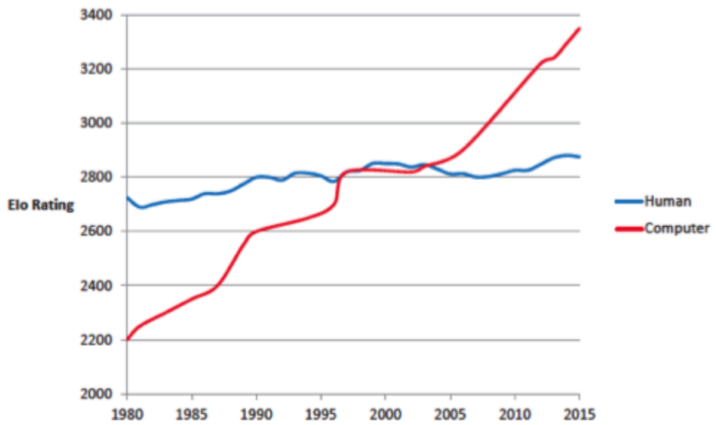
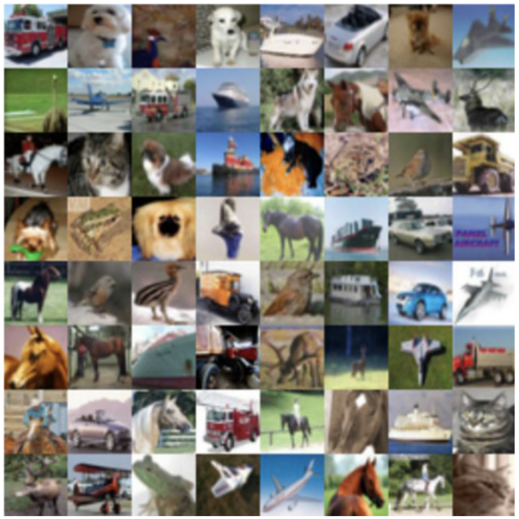


FIGURE 2.3 Elo scores—a measure of competency in competitive games—showing the chess-playing competency of humans and machines, measured over time. SOURCE: Courtesy of Murray Campbell.

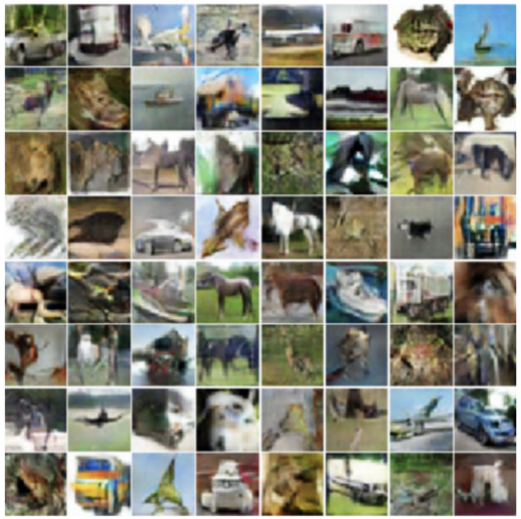
Spectaculary breakthrough

Image generation

Recent Nvidia results



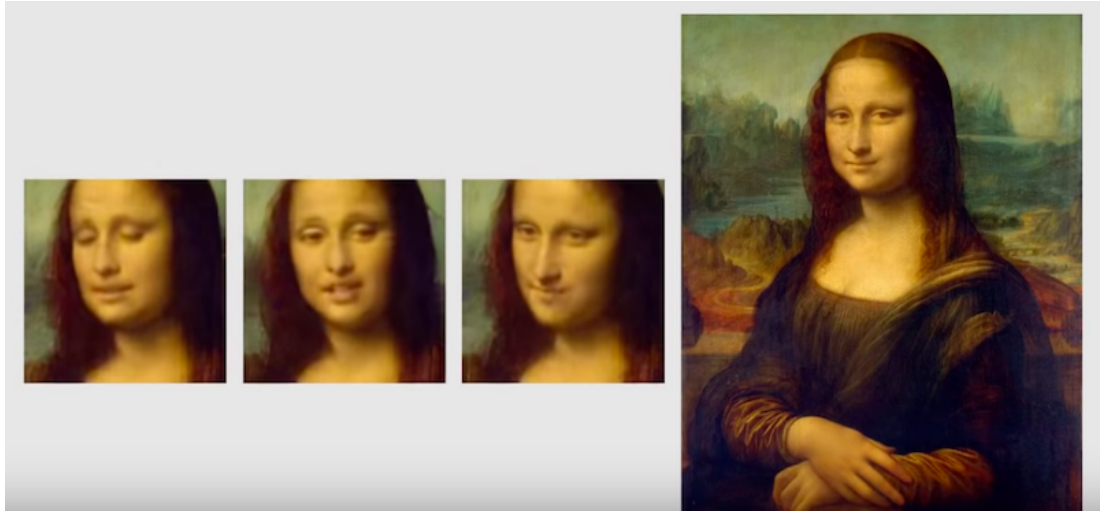
Real images (CIFAR-10)



Generated images

Spectaculary breakthrough

Should we still trust what we see?



Why now ?

Huge training resources for huge models

- Huge volumes of training data
- Huge computational ressources (clusters of GPUs)

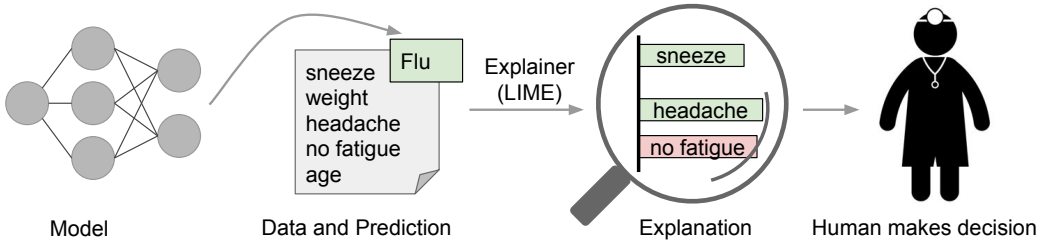
Advances in understanding optimizing NNs

- Regularization (Dropout...)
- Making gradient flow (ResNets, LSTMs, ...)

Faster diffusion than ever

- Softwares
- Results
  - Publications (arxiv publication model) + codes
  - Architectures, weights (3 python lines for loading a state of the art computer vision model!)

Next Challenges: Robustness, Explainabilty



[Ribeiro et al., KDD 2016]

Black box models

- Machine Learning models are black box models
- Urgent needs for trustability with the increasing performance in real-life tasks (automatic cars, army, health etc)
- Several levels of understanability / explainability: Explain a decision on an input / the whole process
- Various kind of methods: Agnostic, model-based ...

Explainabilty

Linear models

- $f(x) = w^T x = \sum_{j=1}^{d} x_j w_j$
- Learning with a regularization term : $C(W) = \sum_{i=1}^{N} l(x^i, y^i, w) + ||w||^2$
- Useless weights go to 0
- Relevance of feature j measured as $|w_j|$

Nonlinear models

- More difficult to disentangle the relevance of all the features
- Popular measure: Gradient of class output wrt input feaures: $|\frac{\partial y^c}{\partial x_j}|$
- Other derived measures

Outline

1. Intro
2. MLPs
   - Basics
   - Deeper in MLPs
3. Architectures
4. Understanding Convolution
5. Conclusion

Where DL comes from

F. Rosenblatt 1958 : The perceptron

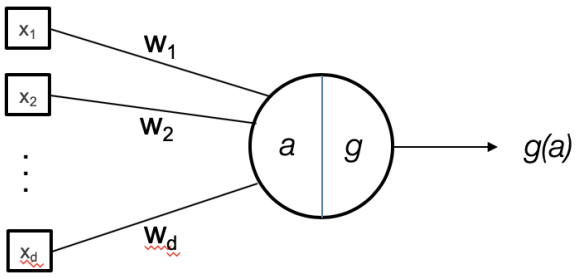
A single Neuron

One Neuron

- Elementary computation

$$\text{activation} = w^T.x = \sum_j w_j x_j + w_0$$

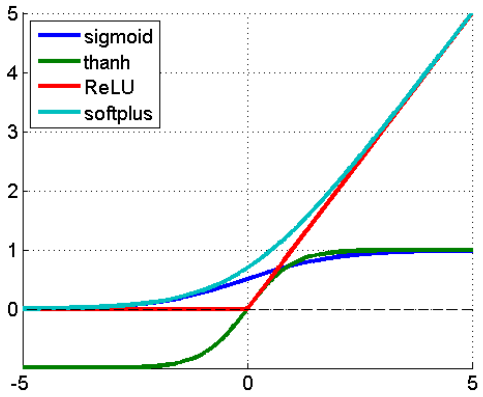
$$\text{output} = g(a(x))$$



Non linearity : g

- Sigmoide, Hyperbolic tangent, Gaussian
- Rectified Linear Unit (RELU)

$$f(x) = 0 \text{ if } x \leq 0$$
$$= x \text{ otherwise}$$



Multi Layer Perceptron (MLP)

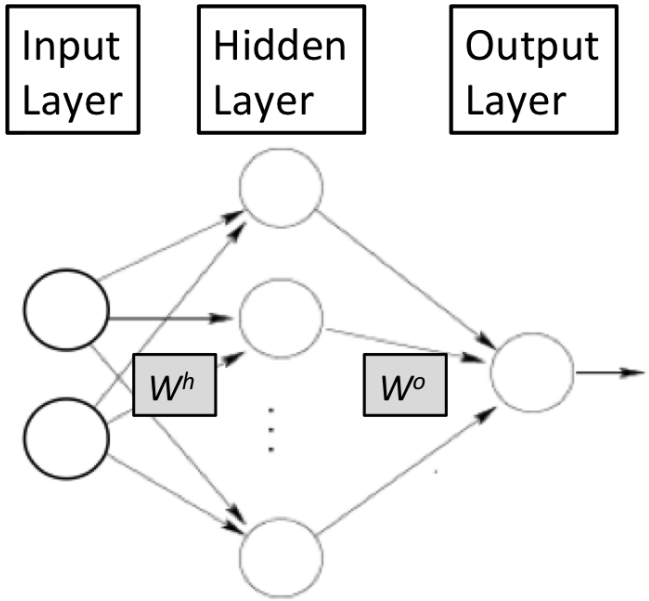
Structure

- Organization in successive layers
  - Input layer
  - Hidden layers
  - Output layer

Function implemented by a MLP

$$g(W^o.g(W^h x))$$

- Inference: Forward propagation from input to output layer
  - Fill the input layer with x: $h_0 = x$
  - Iterate from the first hidden layer to the last one
    - $h^l = W^l \times h^{l-1}$
    - $h^l = g(h^l)$



Learning a MLP

Learning as an optimization problem

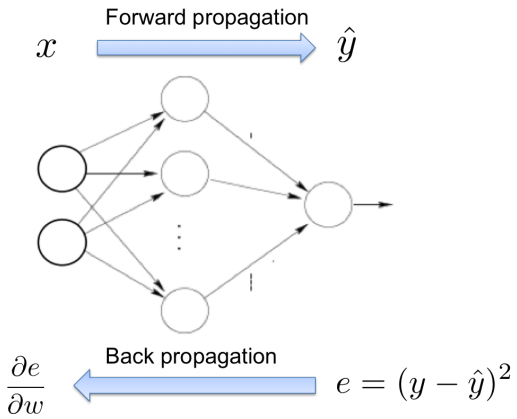
- Objective function of parameters set w for a given training set T

$$\begin{aligned} C(w) =& F(w) + R(w) \\ =& \sum_{(x,y)\in T} L_w(x, y, w) + ||w||^2 \end{aligned}$$

- Gradient descent optimization: $w = w - \epsilon \frac{\partial C(w)}{\partial w}$

Backpropagation

- Use chain rule for computing derivative of the loss with respect to all weights in the NN



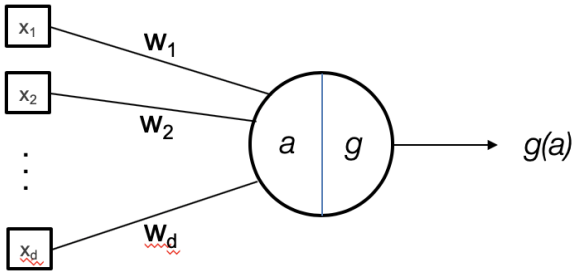
A single ReLU Neuron

One Neuron

- Elementary computation

$$\text{activation} = w^T.x = \sum_j w_j x_j + w_0$$

$$\text{output} = ReLU(a(x))$$



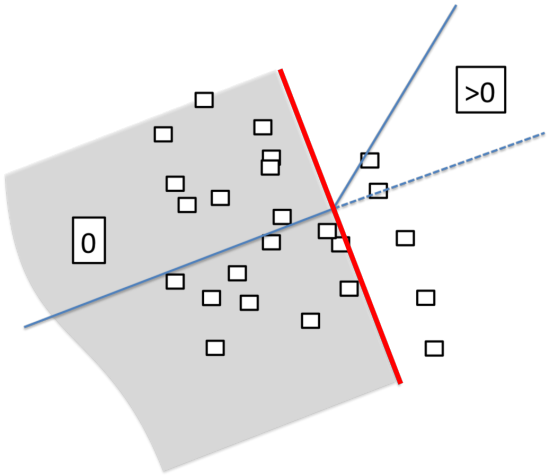
A single ReLU Neuron

One Neuron

- Elementary computation

$$\text{activation} = w^T.x = \sum_j w_j x_j + w_0$$

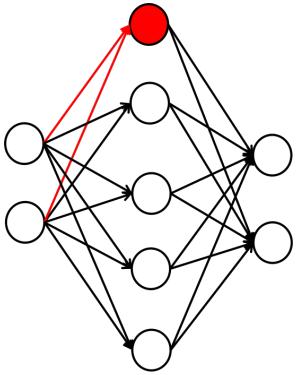
$$\text{output} = ReLU(a(x))$$

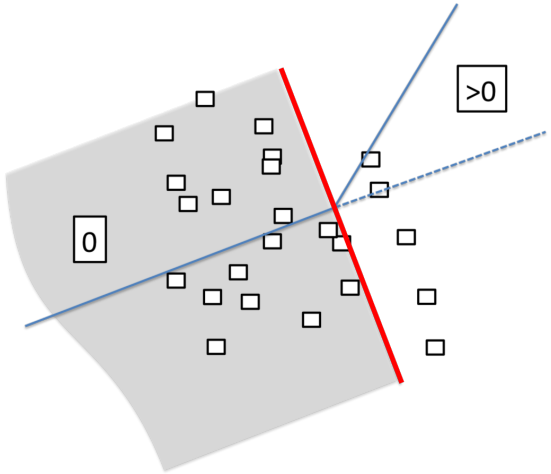


What a MLP may compute

What does a hidden neuron

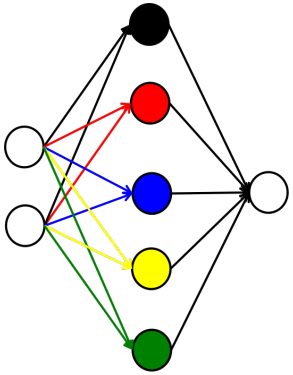
- Divides the input space in two

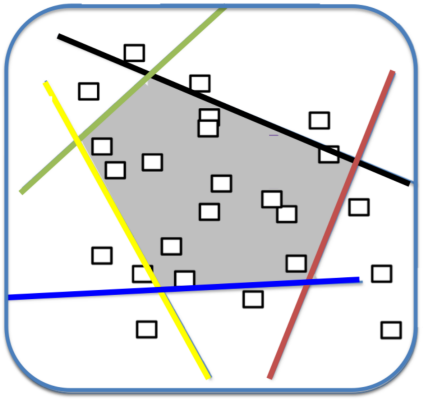




Combining multiple hidden neurons

- Allows identifying complex areas of the input space
- New (distributed) representation of the input

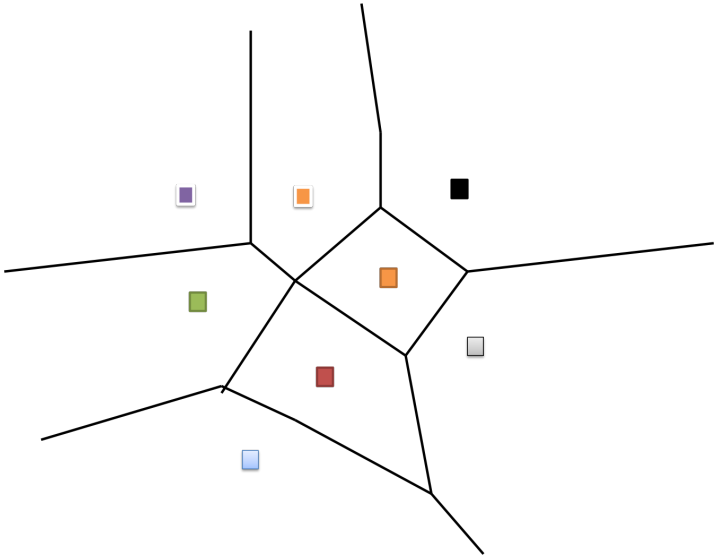


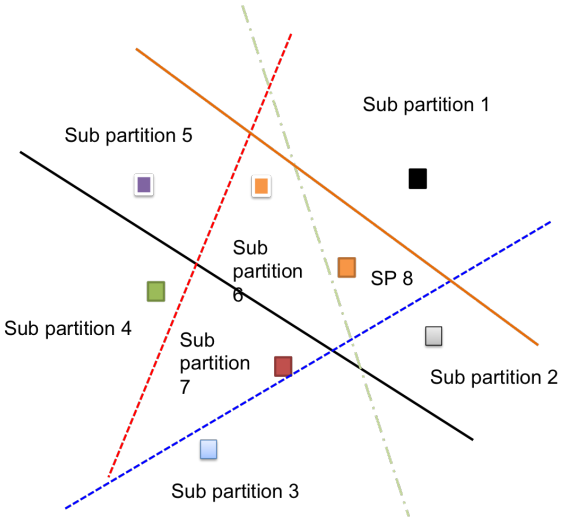


MLP compute distributed representations

Might be much more efficient than non distributed ones

Somehow the number of regions in which a NN architecture may divide the input space is a measure of its capacity





MLP = Universal approximators

One layer is enough !

- Theorem [Cybenko 1989]: Let $\phi(\cdot)$ be a nonconstant, bounded, and monotonically-increasing continuous function. Let I_m denote the m-dimensional unit hypercube $[0,1]^m$. The space of continuous functions on I_m is denoted by $C(I_m)$. Then, given any $\epsilon > 0$, there exists an integer N, such that for any function $f \in C(I_m)$, there exist real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^m$, where $i = 1, \cdots, N$, such that we may define:

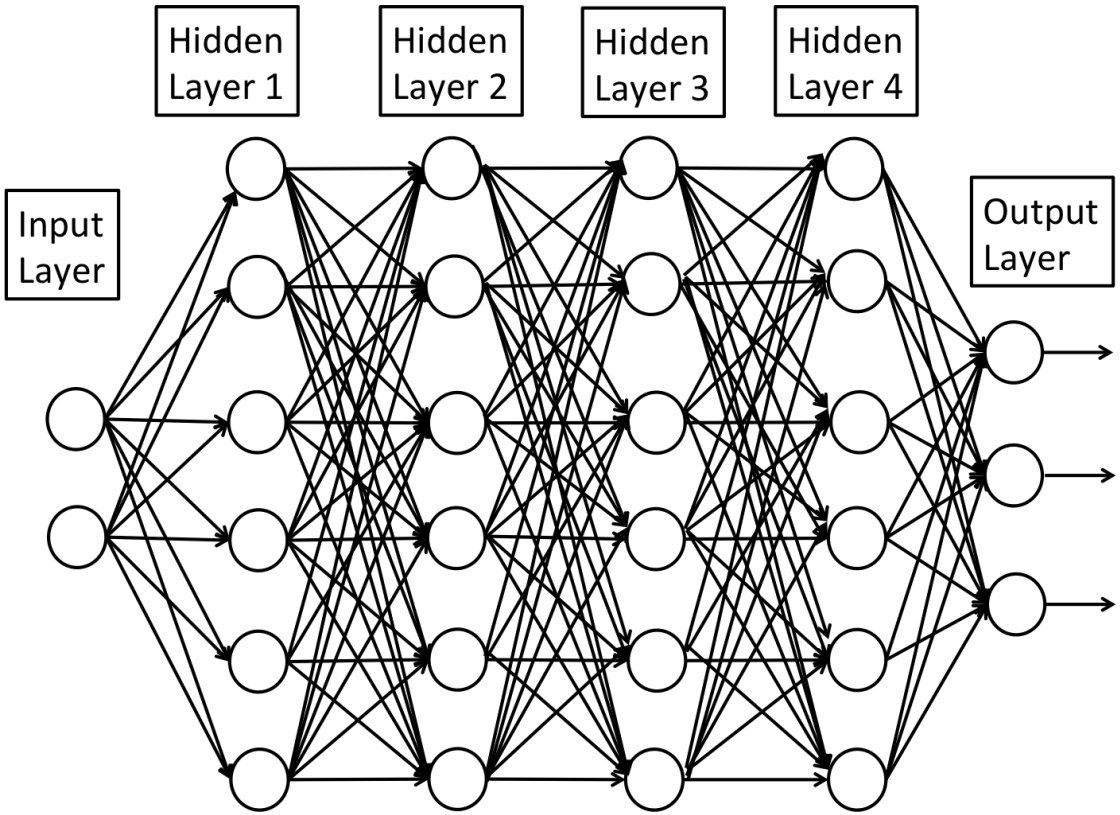
$$F(x) = \sum_{i=1}^{N} v_i \phi\left(w_i^T x + b_i\right)$$

  as an approximate realization of the function f where f is independent of ϕ ; that is : $|F(x) - f(x)| < \epsilon$ for all $x \in I_m$. In other words, functions of the form $F(x)$ are dense in $C(I_m)$.
- Existence theorem only
- Many reasons for not getting good results in practice

Outline

1. Intro
2. MLPs
3. Architectures
   - Dense
   - Autoencoders
   - Convolution
4. Understanding Convolution
5. Conclusion

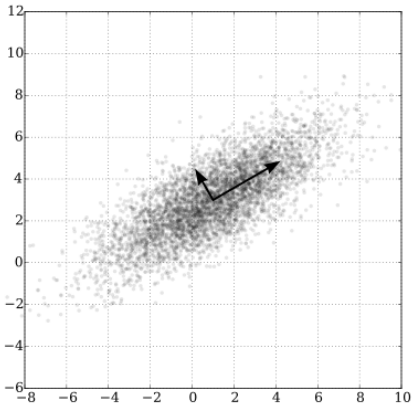
Dense architecture



Autoencoders

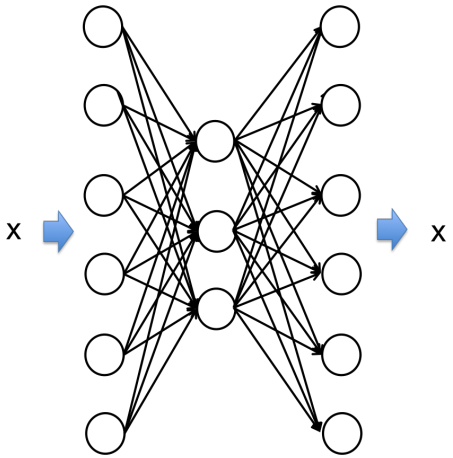
Principal Component Analysis

- Unsupervised standard (Linear) Data Analysis technique
  - Visualization, dimension reduction
- Aims at finding principal axes of a dataset



NN with Diabolo shape

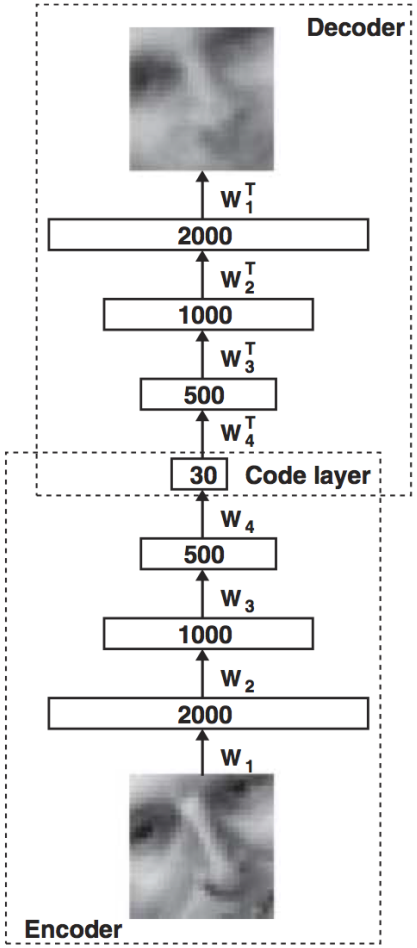
- Reconstruct the input at the output via an intermediate (small) layer
- Unsupervised learning
- Non linear projection, distributed representation
- Hidden layer may be larger than input/output layers



Deep autoencoders

Deep NN with Diabolo shape

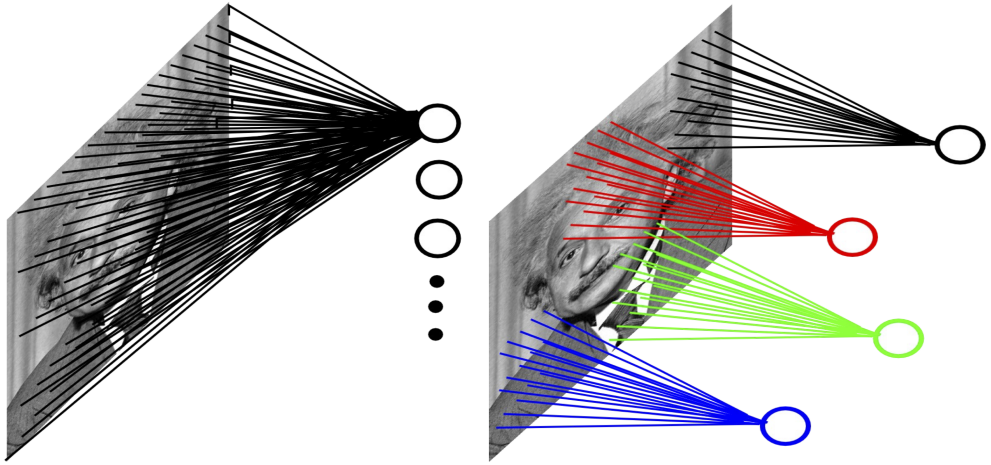
- Extension of autoencoders (figure [Hinton et al., Nature 2006])
- Pioneer work that started the Deep Learning wave



Convolutional architectures

Convolutional layers

- Exploit a structure in the data
  - Images : spatial structure
  - Texts, audio ; temporal structure
  - videos : spatio-temporal structure
- Use shared weigths

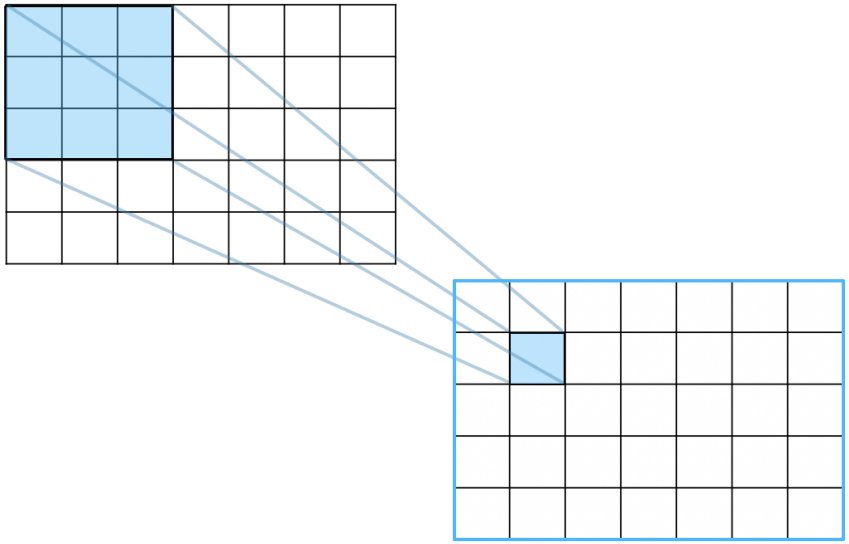


Dense vs. Locally connected

[LeCun and Ranzato Tutorial, DL, 2015]

Convolution operator

Kernel Convolution $\Longleftrightarrow$ Sliding Window

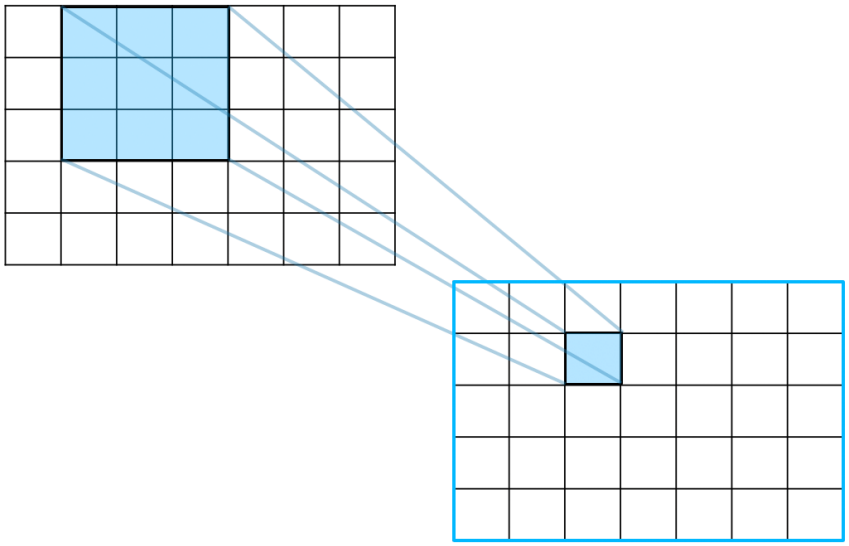


Example for 3x3 kernel

$$(I * f)(x, y) \equiv \sum_{u=-1}^{+1} \sum_{v=-1}^{+1} I(x+u, y+v) f(u, v)$$

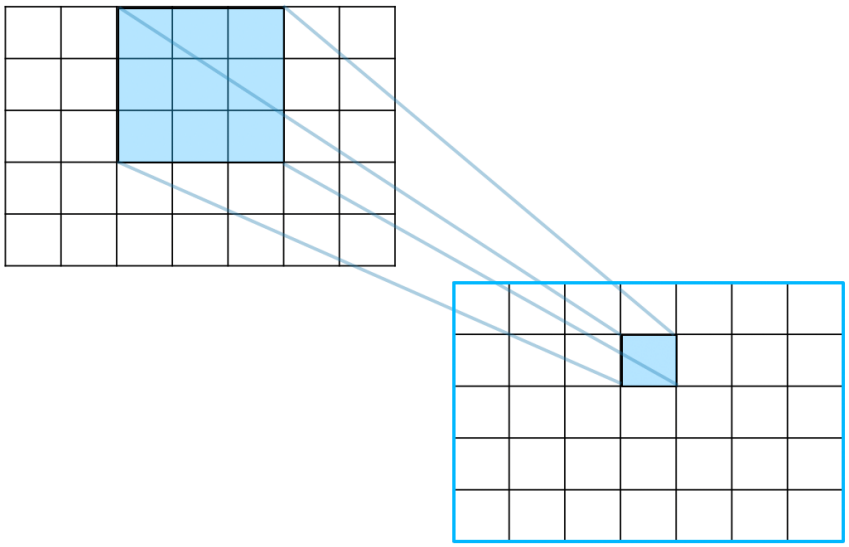
Convolution operator

Kernel Convolution $\Longleftrightarrow$ Sliding Window



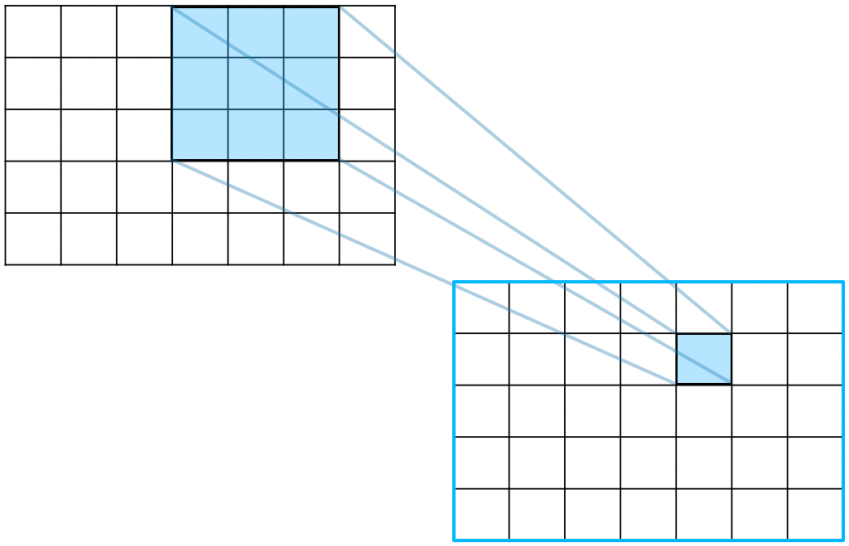
Convolution operator

Kernel Convolution $\Longleftrightarrow$ Sliding Window



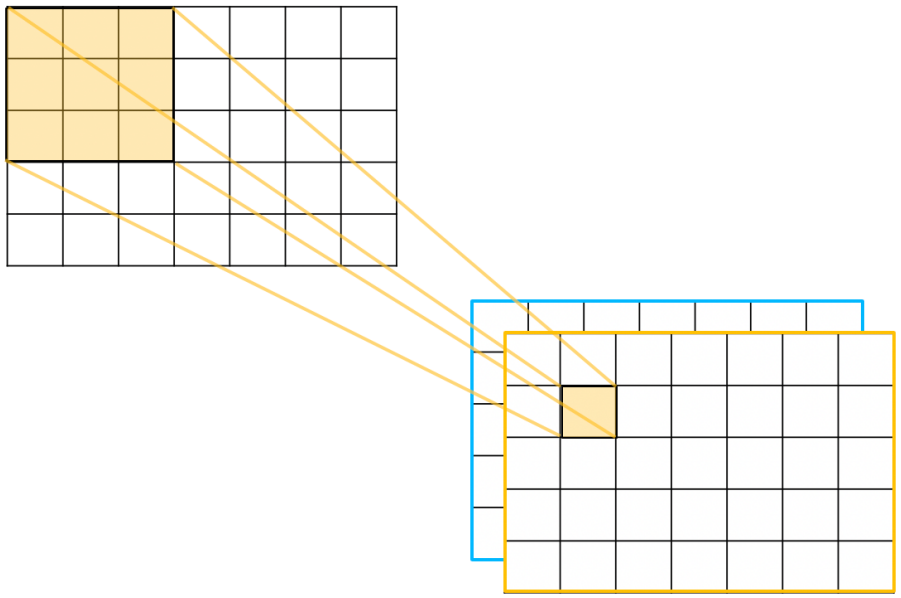
Convolution operator

Kernel Convolution $\Longleftrightarrow$ Sliding Window



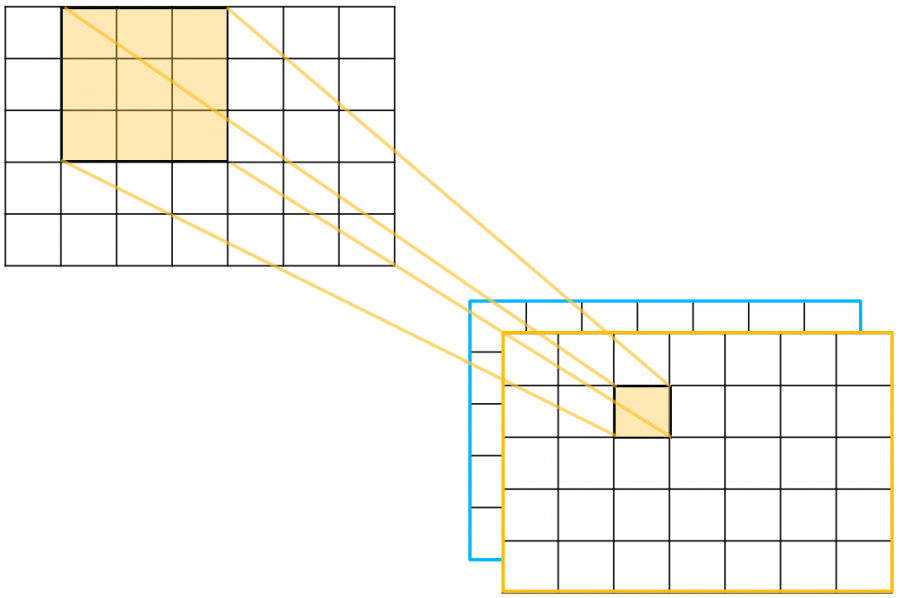
Convolution operator

Kernel Convolution $\Longleftrightarrow$ Sliding Window



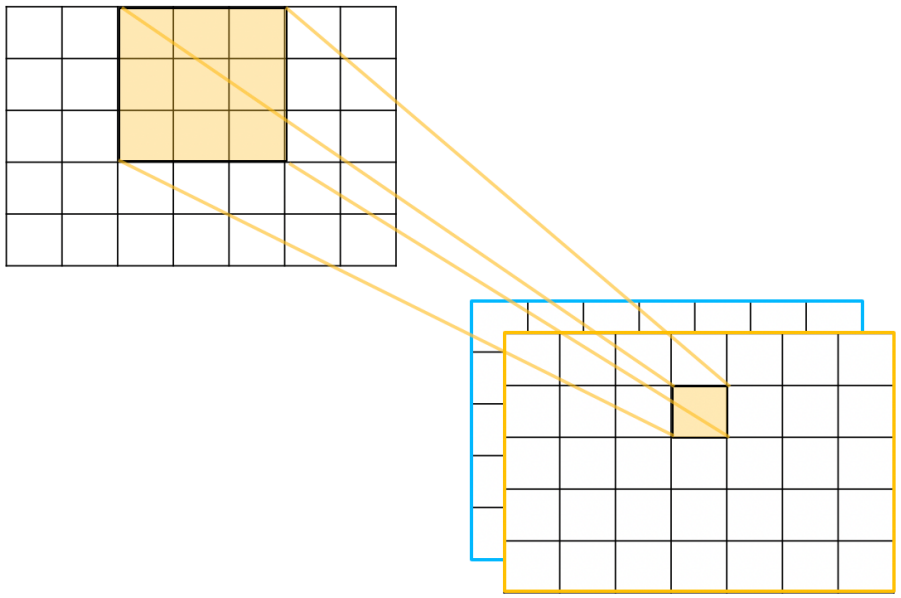
Convolution operator

Kernel Convolution $\Longleftrightarrow$ Sliding Window



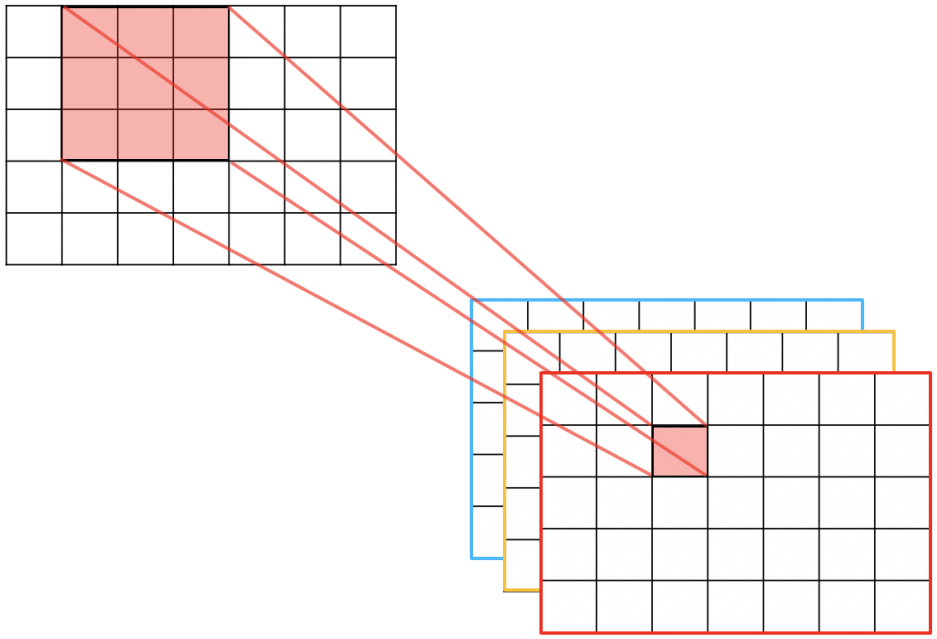
Convolution operator

Kernel Convolution $\Longleftrightarrow$ Sliding Window



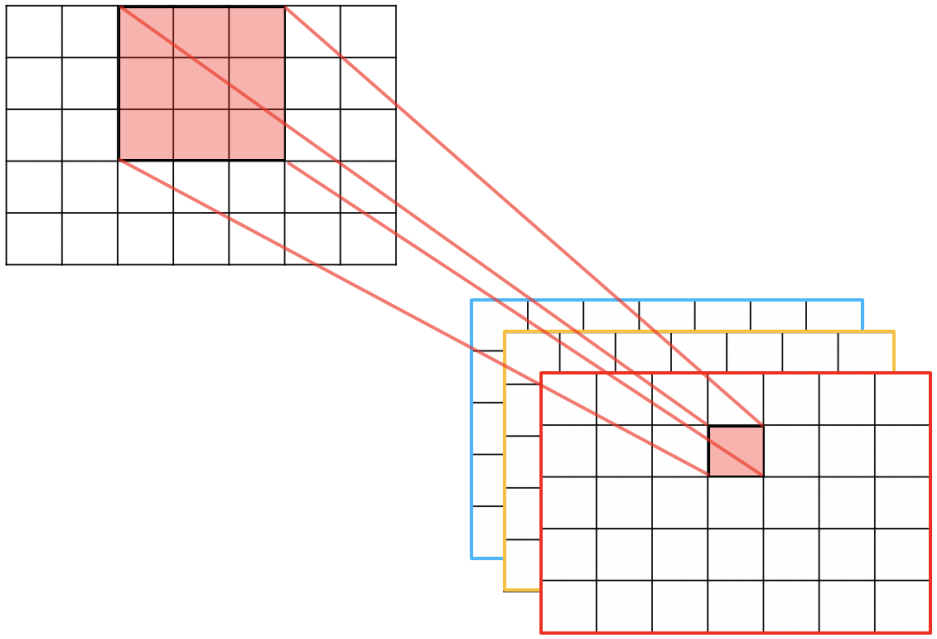
Convolution operator

Kernel Convolution $\Longleftrightarrow$ Sliding Window



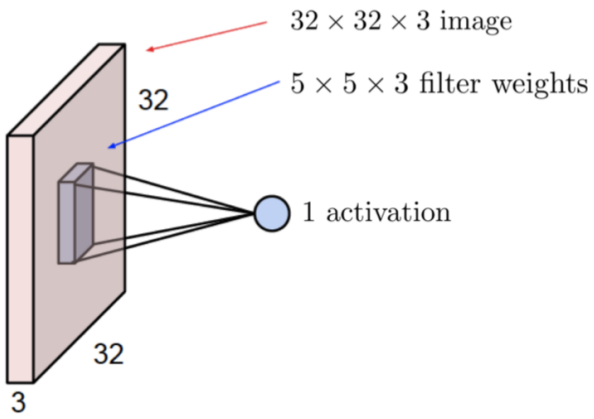
Convolution operator

Kernel Convolution $\Longleftrightarrow$ Sliding Window



Convolution layer

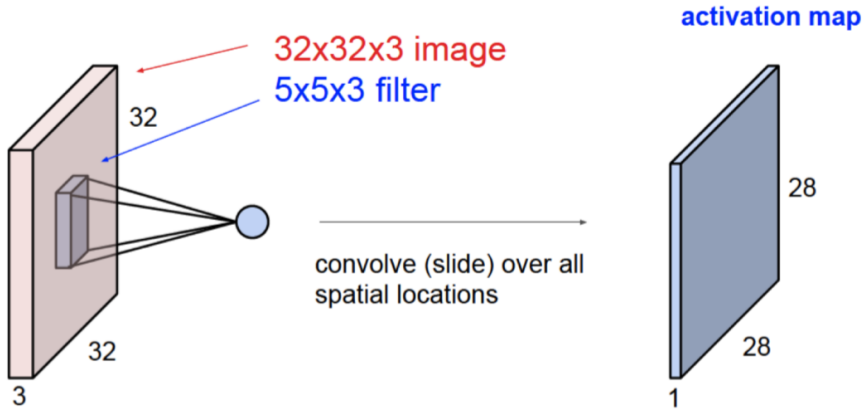
Convolution of multiple inputs with several small filter yields several activation maps



[From Fei Feil Li slides]

Convolution layer

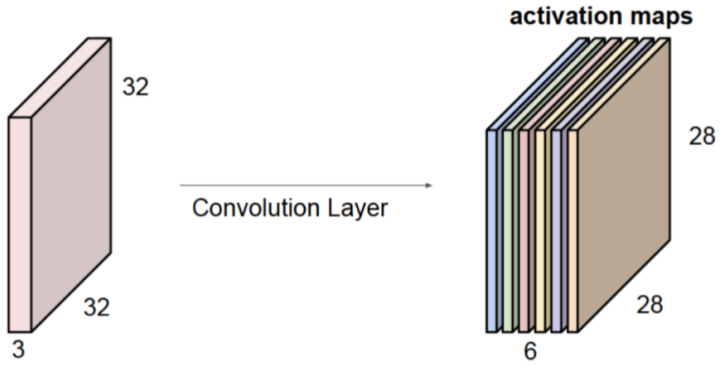
Convolution of multiple inputs with several small filter yields several activation maps



[From Fei Feil Li slides]

Convolution layer

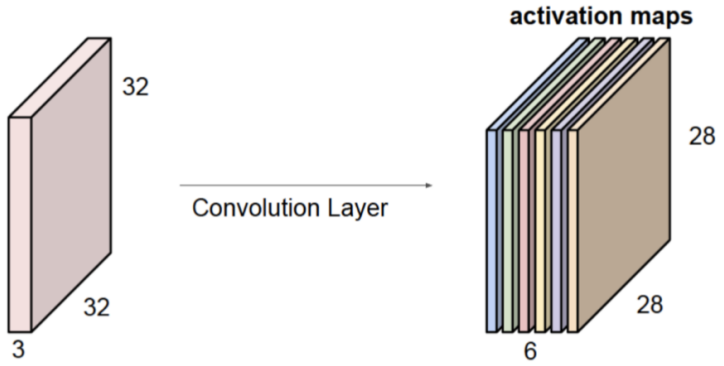
Convolution of multiple inputs with several small filter yields several activation maps



[From Fei Feil Li slides]

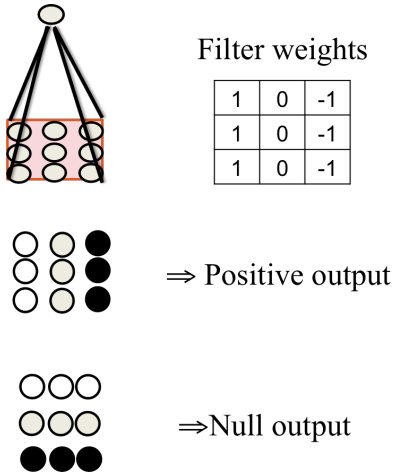
Convolution layer

Convolution of multiple inputs with several small filter yields several activation maps



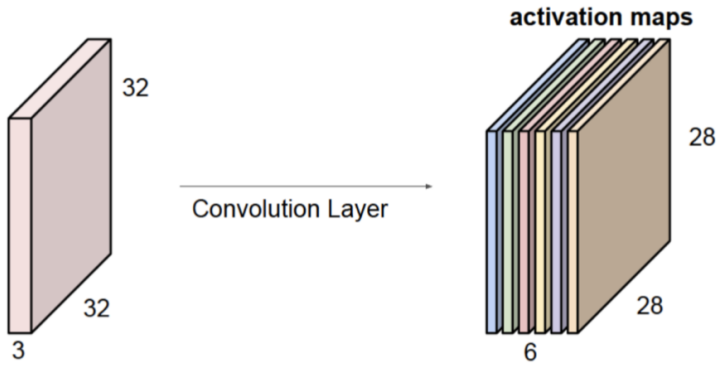
[From Fei Feil Li slides]

Example of a filter



Convolution layer

Convolution of multiple inputs with several small filter yields several activation maps



[From Fei Feil Li slides]

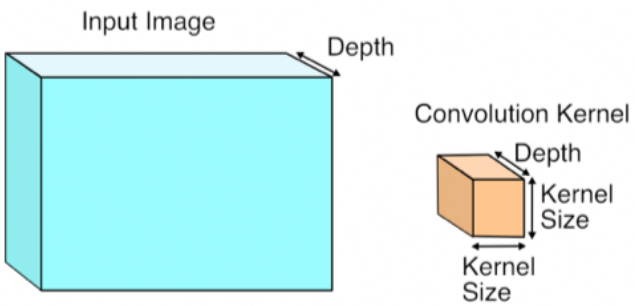
Very few free parameters but intensive computation

- Input : N_m input maps of size $H \times W$
- Output N_f filters with filter size $= h \times w$
- Parameters : $h \times w \times N_m \times N_f$
- Fwd computation : $\approx H \times W \times N_f \times N_m \times h \times w$
- For instance (very small case) :
  - From 3 32×32 input maps $\rightarrow$ 6 filters with filter size 3×3
  - $3 \times 3 \times 3 \times 6 = 48$ parameters
  - 165 888 operations

Convolution layer implementation

How is it implemented? Why is it fast on GPUs?

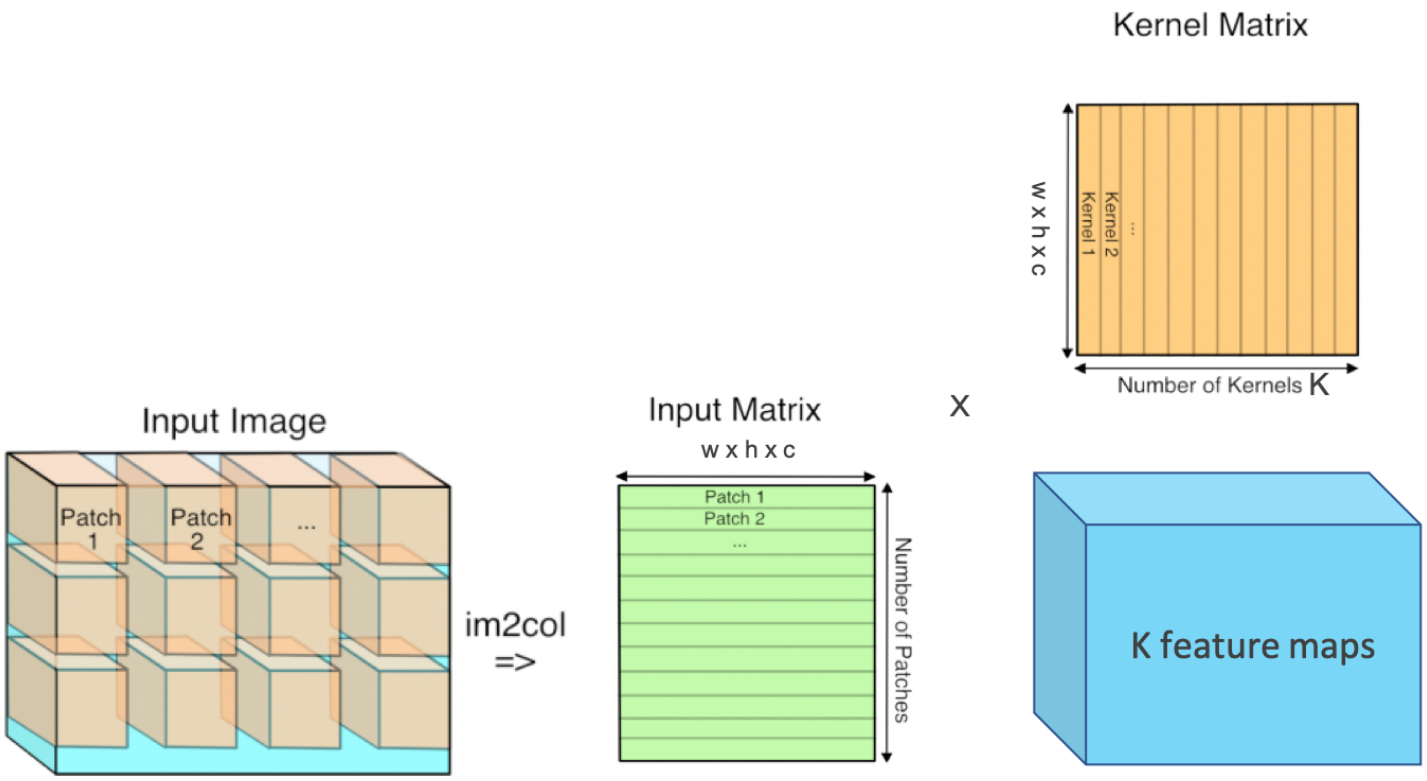
Given an image and some convolution parameters, learn K convolution kernels:



Convolution layer implementation

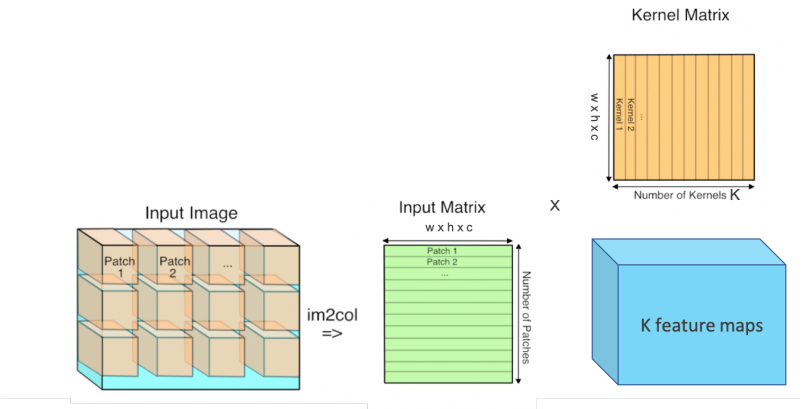
How is it implemented? Why is it fast on GPUs?

Because it's using matrices multiplications!



Convolution layer implementation

How is it implemented? Why is it fast on GPUs?

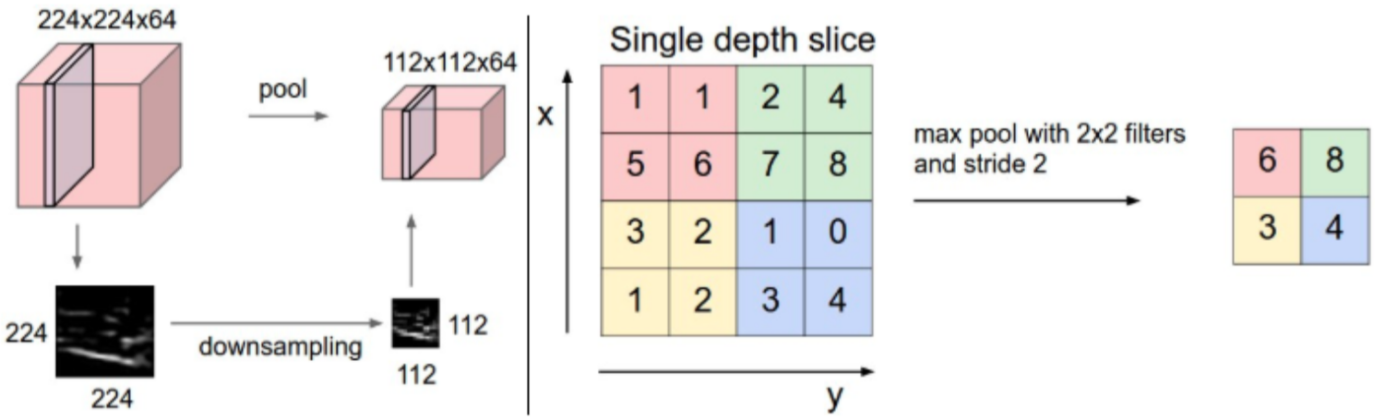


- Given an image A, the K feature maps U are obtained by reshaping the result of $im2col(A) \times V$, with V the matrix that contains vectorized convolution kernels $\in \mathbb{R}^{whc \times K}$
- Sizes of feature maps obviously depend of stride of convolution

Convolution layer

Aggregation layer

- Subsampling layer (one per activation map) with aggregation operator
- Max pooling → brings invariance and robustness



Complexity

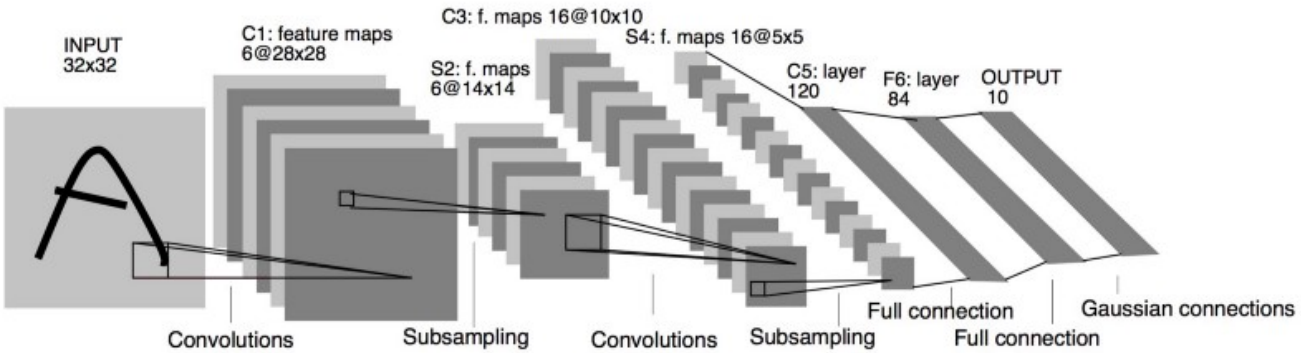
- No parameters
- Moderate computation

Old and new convolutional architectures

Convolution architectures

- Most often a mix of (convolutional + pooling) layers followed by dense layers
- Most computation effort are in propagating through convolution layers
- Most parameters are in final fully connected layers

LeNet [LeCun and al., 1997]

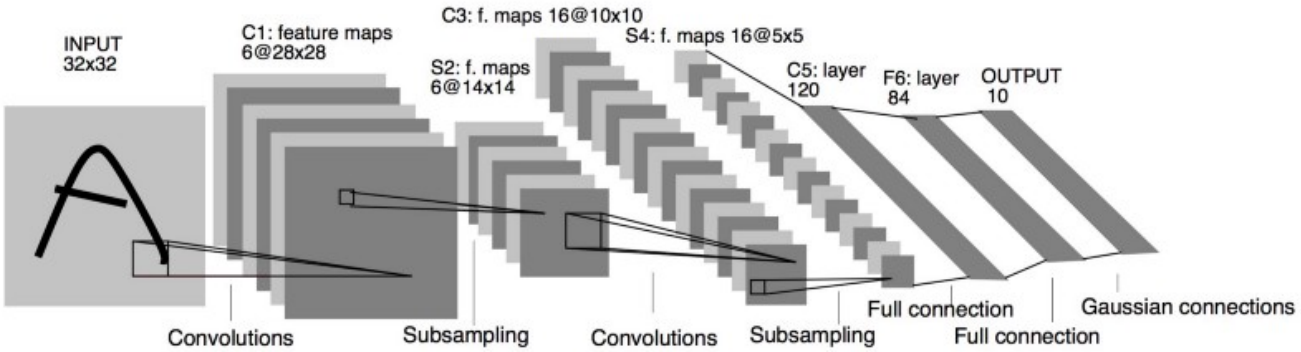


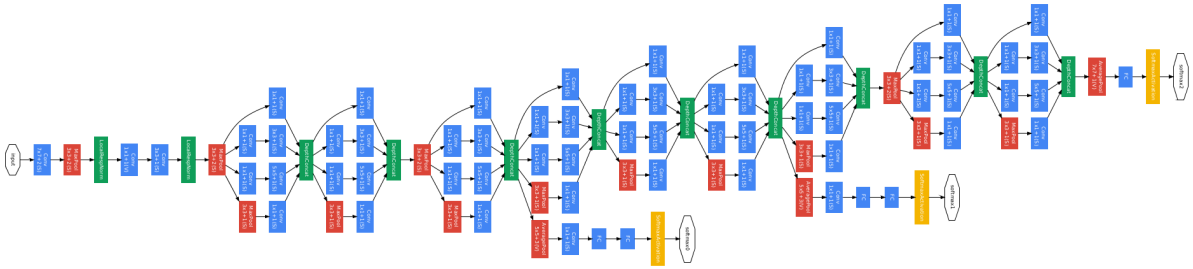
Old and new convolutional architectures

Convolution architectures

- Dit it change so much ?

LeNet [LeCun and al., 1997]



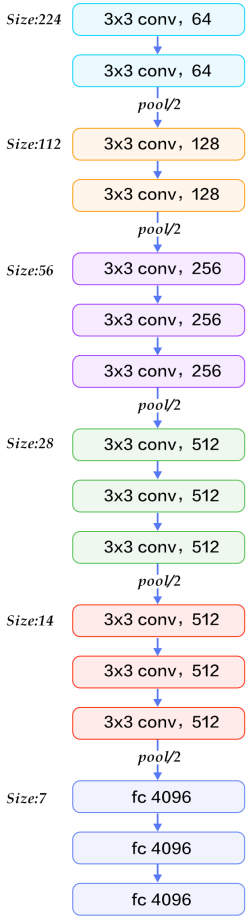


GoogleNet 2014

Old and new convolutional architectures

VGG idea

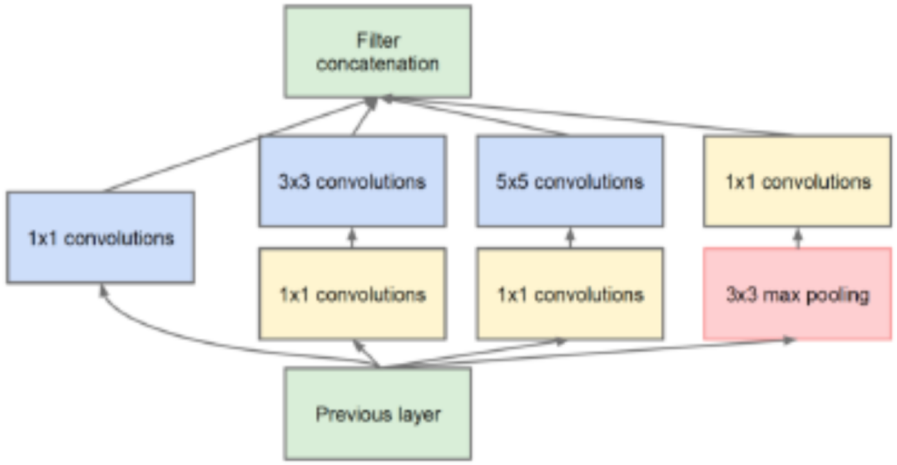
- Modular design
  - 3x3 conv as basis
  - Stack the same module
  - Same computation for each module (1/2 spatial size => 2x filters)

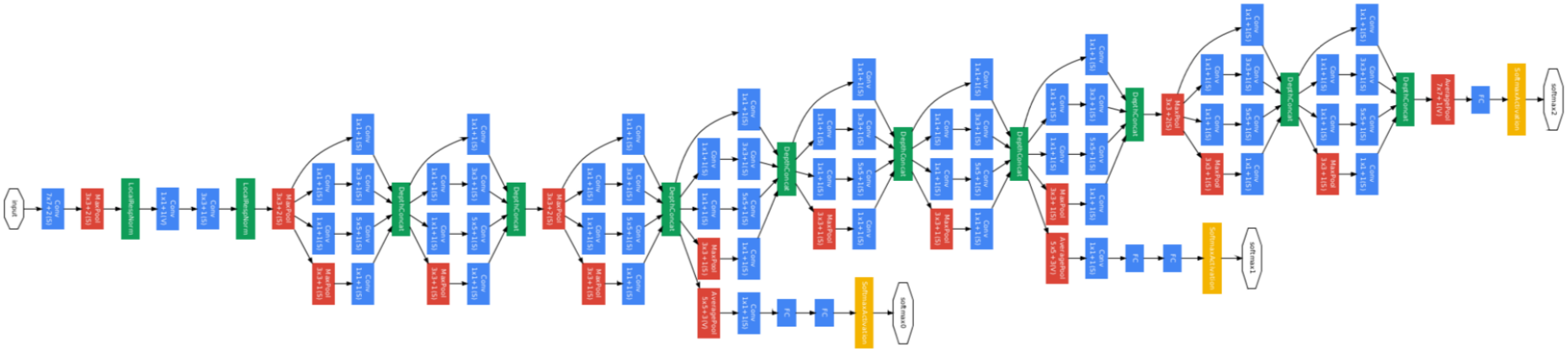


Old and new convolutional architectures

Inception idea (GoogleNet)

- Inception modules
  - Multiple branchs (1x1, 3x3, 5x5, pool)
  - Shortcuts (stand alone 1x1, merged by concat)
  - Bottleneck (reduce dim by 1x1 before expensive 3x3/5x5 convs)

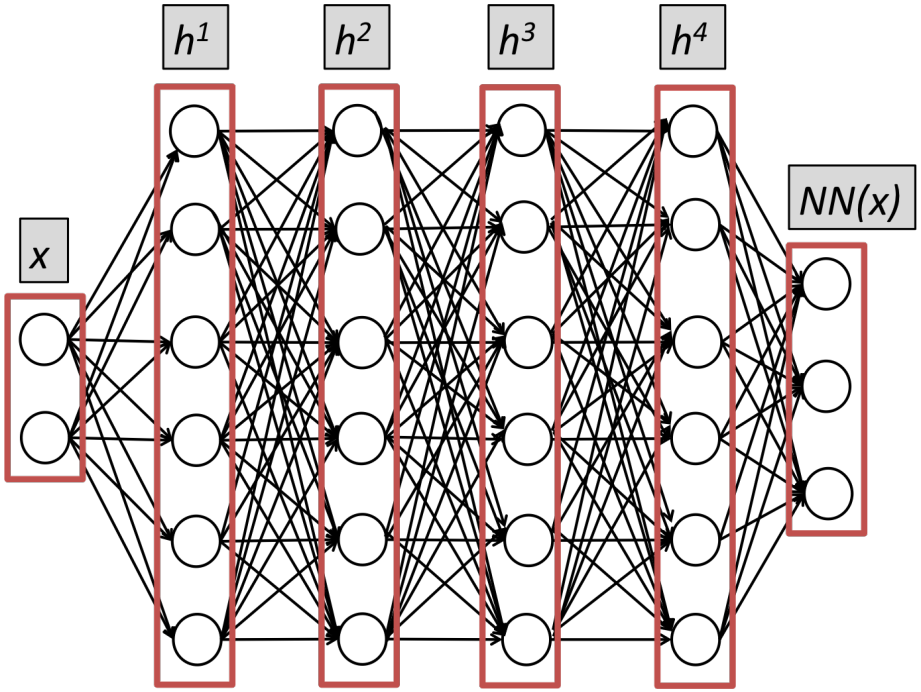




Deep Learning = Representation Learning

Hierarchy of representation spaces by successive hidden layers

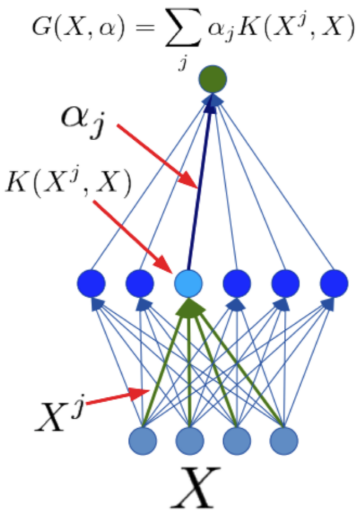
$$h^i(x) = g(W^i \times h^{i-1}(x))$$



Shallow vs deep models

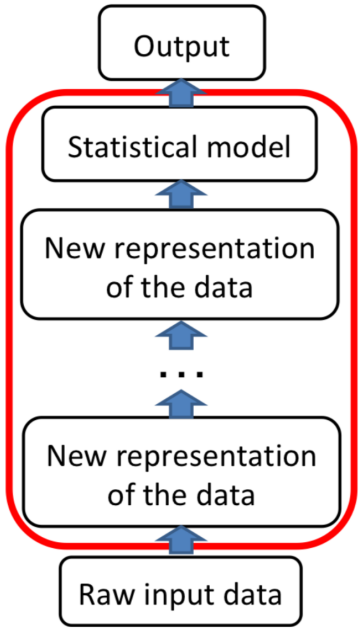
From [LeCun tutorial Statlearn]

- Neural Networks with one hidden layer are shallow models
- SVMs are shallow models



DL

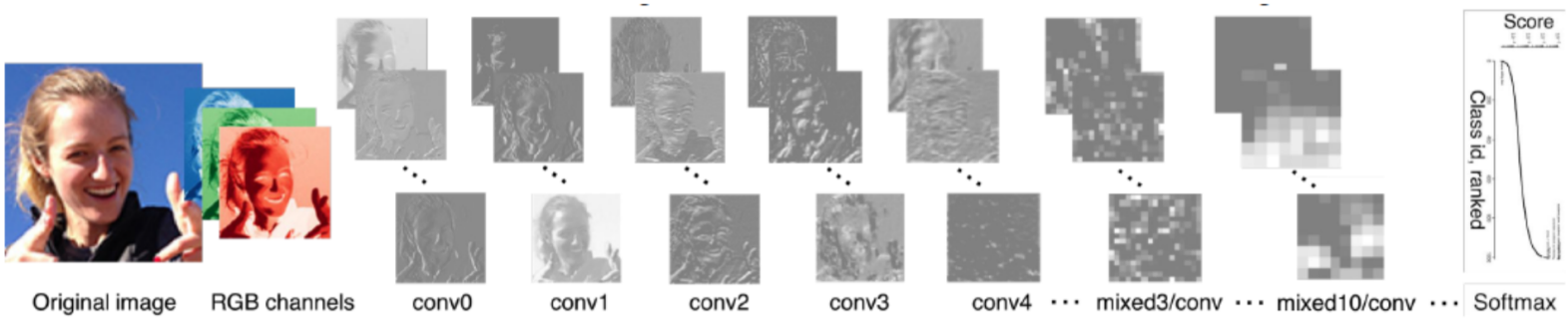
- Joint learning of a hierarchy of representations and of a prediction model



Feature hierarchy : from low to high level

What feature hierarchy means ?

- Low-level features are shared among categories
- High-level features are more global and more invariant



[From Taigman et al., 2014]

Outline

1. Intro
2. MLPs
3. Architectures
4. Understanding Convolution
   - Kronecker Product Factorization
   - Convolution Layer and Complete Kronecker Product
   - CKP for transfert learning
5. Conclusion

Kronecker Product

If A is a bloc matrix, such that $A_{i,j} = B_{i,j} \times C$, then $A = B \otimes C$

$$\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \otimes \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

$$=$$

$$\left[\begin{array}{ccc|ccc} b_{11}c_{11} & b_{11}c_{12} & b_{11}c_{13} & b_{12}c_{11} & b_{12}c_{12} & b_{12}c_{13} \\ b_{11}c_{21} & b_{11}c_{22} & b_{11}c_{23} & b_{12}c_{21} & b_{12}c_{22} & b_{12}c_{23} \\ b_{11}c_{31} & b_{11}c_{32} & b_{11}c_{33} & b_{12}c_{31} & b_{12}c_{32} & b_{12}c_{33} \\ \hline b_{21}c_{11} & b_{21}c_{12} & b_{21}c_{13} & b_{22}c_{11} & b_{22}c_{12} & b_{22}c_{13} \\ b_{21}c_{21} & b_{21}c_{22} & b_{21}c_{23} & b_{22}c_{21} & b_{22}c_{22} & b_{22}c_{23} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \end{array}\right]$$

Nearest Kronecker Product

Given A, a matrix $\in \mathbb{R}^{m \times n}$, with $m = m1 \times m2$ et $n = n1 \times n2$, how to get $B \in \mathbb{R}^{m1 \times n1}$ et $C \in \mathbb{R}^{m2 \times n2}$, such that $A \approx B \otimes C$?

Theorem (CF Van Loan - 1993)

Assume that $A \in \mathbb{R}^{m \times n}$ with $m = m1m2$ and $n = n1n2$. If $B \in \mathbb{R}^{m1 \times n1}$ and $C \in \mathbb{R}^{m2 \times n2}$, then

$$\|A - B \otimes C\|_F = \|\Re(A) - vec(B)vec(C)^T\|_F$$

where $\Re$ is a $\Re$eshape operator, and $vec()$ is matrix vectorization.

$\Rightarrow$ B and C can be computed by SVD of $\Re(A)$!!

Nearest Kronecker Product

$$\|A - B \otimes C\|_F = \|\Re(A) - vec(B)vec(C)^T\|_F$$

Reshape and Vectorization example

$$\left\| \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \\ \hline a_{51} & a_{52} & a_{53} & a_{54} \\ a_{61} & a_{62} & a_{63} & a_{64} \end{array}\right] - \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \otimes \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \right\|_F = \left\| \begin{bmatrix} a_{11} & a_{21} & a_{12} & a_{22} \\ a_{31} & a_{41} & a_{32} & a_{42} \\ a_{51} & a_{61} & a_{52} & a_{62} \\ a_{13} & a_{23} & a_{14} & a_{24} \\ a_{33} & a_{43} & a_{34} & a_{44} \\ a_{53} & a_{63} & a_{54} & a_{64} \end{bmatrix} - \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{22} \\ b_{32} \end{bmatrix} \begin{bmatrix} c_{11} & c_{21} & c_{12} & c_{22} \end{bmatrix} \right\|_F$$

$\Rightarrow$ B and C are solutions of rank-1 SVD with highest eigenvalue

$$vec(B) = \sqrt{\sigma 1} U(:;1) \qquad vec(C) = \sqrt{\sigma 1} V(:;1)$$

Complete Kronecker Product

Rank-k Generalization

$$\Re(A) \approx \sum_{i=1}^{k} \sigma_i U(:;i) V(:;i)^T = U_k \Sigma_r V_k^T$$

with U_k, Σ_k and V_k the truncated U, σ and V at rank k

Rank-k Kronecker Product

$$A = \sum_{i=1}^{k} \sigma_k B_k \otimes C_k$$

with $B_k = reshape(U_k, (m1, n1))$ and $C_k = reshape(V_k, (m2, n2))$

Remarks

Remark 1

Finding both B and C consists in a bilinear factorization of $\Re(A)$, then reshaping U_k and V_k

Remark 2

Integrating $\sqrt{\Sigma_k}$ in U_k and V_k, allows to write $\Re(A) \approx U_k \times V_k^T$

$$\min_{U_k} ||\Re(A) - U_k V_k^T|| \Rightarrow U_r = \Re(A) \times V_k \times (V_k^T V_k)^{-1}$$

Remark 3

The $\Re$eshape operator is

$$A' = \Re(A) \in \mathbb{R}^{m1n1 \times m2n2} \Rightarrow A'(i + (j-1)m1; :) = vec(A_{ij})T$$

with i *in* $[1..m1]$ and j *in* $[1..n1]$
$\Rightarrow$ Similar to *im2col* transformation
$\Rightarrow$ Equivalent when stride = shape(C) in Convolution layers

Convolutions and Kronecker Product

Consider the following:

- a convolution layer setup to learn k filters, with $stride = shape(filters)$
- V_k the matrix that contains k vectorized kernels convolution in column
- A a given multichannels image (ie: RGB image or feature maps)
- U_k the matrix that contains k vectorized feature maps in row
- $\Re$ the reshape operator corresponding to *im2col*

Convolution Layer versus Complete Kronecker Product

- Convolution $\Rightarrow U_k = \Re(A) V_k$
- CKP $\Rightarrow U_k = \Re(A) V_k (V_k^T V_k)^{-1}$
- with V_k learnt by backpropagation

Are they equivalent??

$(V_k^T V_k)^{-1} = I$ when kernels are orthogonal

$\Rightarrow$ Yes, when $k < rank(\Re(A))$

Learning CNN with Kronecker Product

Disclaimer: Work in progress!

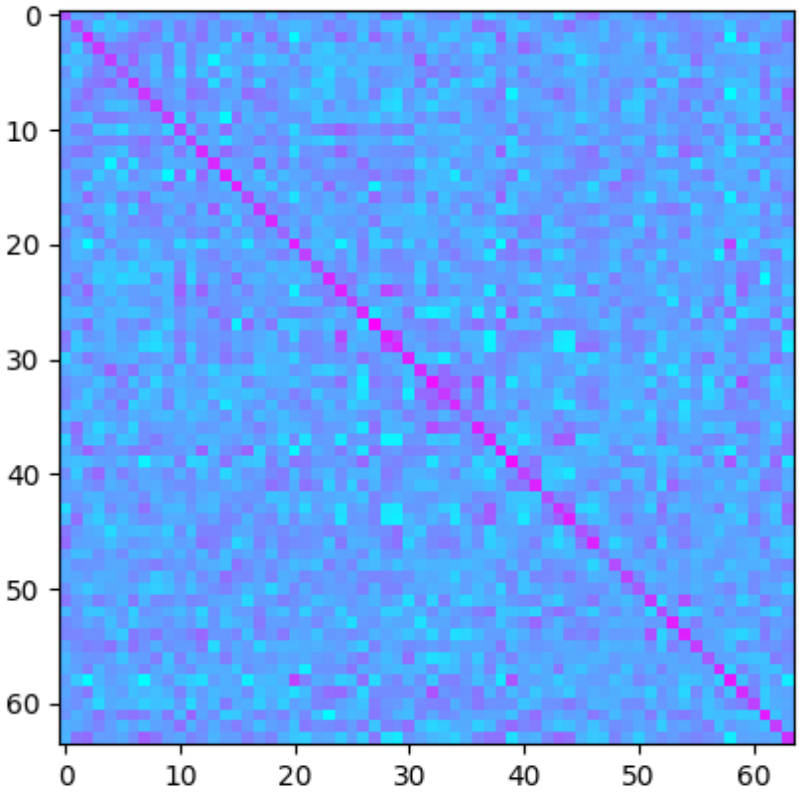
Convolutions and CKP are mostly equivalent, except for bottom layers

- First layer
  - input tensor of shape $W \times H \times 3$
  - learns k_1 3x3 filters $\Rightarrow \Re(A) \in \mathbb{R}^{P \times 27}$, with $P >> 27$ the number of patches according to a given stride
  - In usual classification architectures $k_1 > 27$, hence filters are not orthogonal.
- Second layer
  - input tensor of shape $W \times H \times k_1$
  - learns k_2 3x3 filters $\Rightarrow \Re(A) \in \mathbb{R}^{P \times 9k_1}$, with $P >> 9k_1$.
  - If $k_2 > 9k_1$, then filters are not orthogonal.

Covariances of filters by layer

Disclaimer: Work in progress!

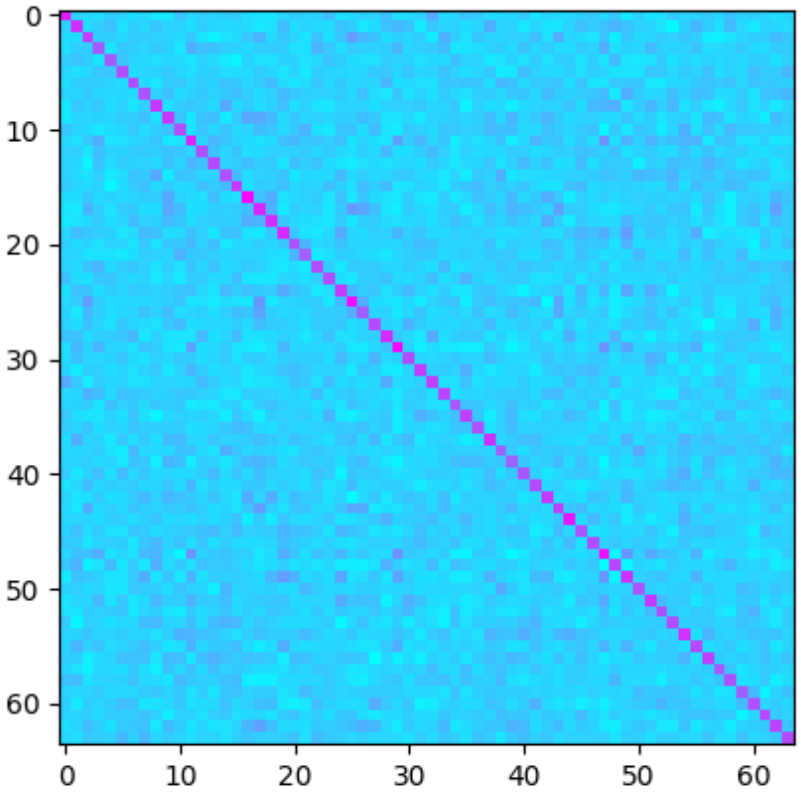
First conv layer



Covariances of filters by layer

Disclaimer: Work in progress!

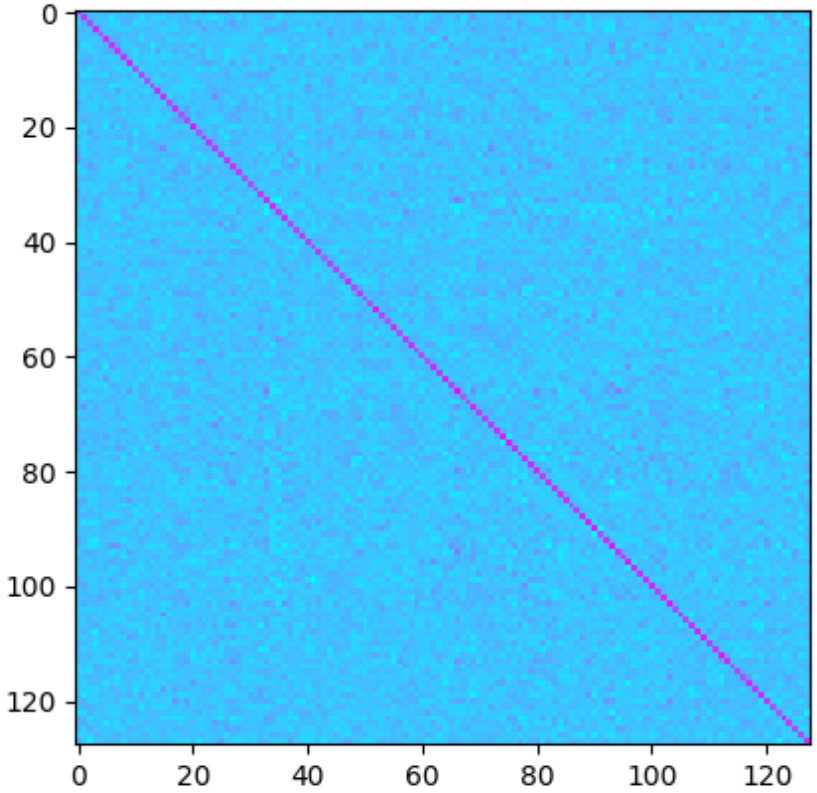
Second conv layer



Covariances of filters by layer

Disclaimer: Work in progress!

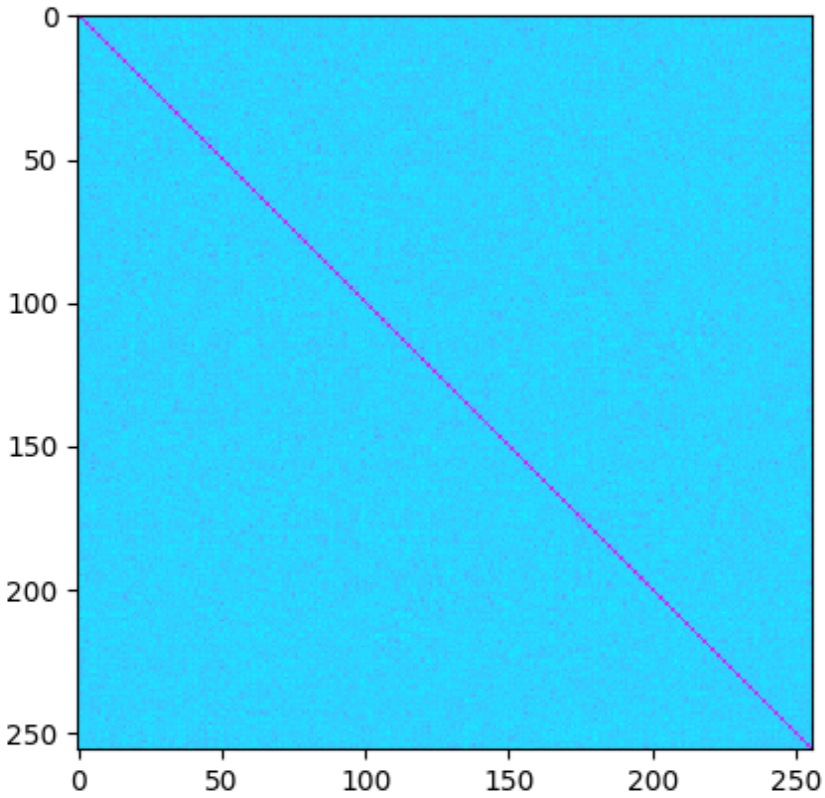
Third conv layer



Covariances of filters by layer

Disclaimer: Work in progress!

Fourth conv layer



Impact on classification accuracy?

Seems not much, still to be analyzed...

- ... maybe because ...
  - omitting covariance adds variability in NN
  - regularization effect?
- ... might help to ...
  - understand how to design CNN architectures
  - compression / distillation
  - transfert learning...

CKP for transfert learning

Having CNN_A, a pretrained model on Task 1, how to transfert its knowledge to Task 2?

- Finetuning VGG, Inception $\Rightarrow$ freeze/finetune (bottom/upper) convolution layers ; lean/finetune new dense layers
- Finetuning ResNet $\Rightarrow$ finetune (highest) convolution layers
- Domain Adaptation

Rank-k Kronecker Product and Weighted Convolutions

- $A = \sum_{i=1}^{k} \sigma_k B_k \otimes C_k$
- $\Re(A) = U_k \Sigma_k V_k$, with $\Sigma_k = \begin{bmatrix} \sigma_1 & \sigma_2 & \dots & \sigma_k \end{bmatrix}$
- $U_k = \Re(A) \Sigma_k V_k$, with $\Sigma_k = \begin{bmatrix} \sigma_1 & \sigma_2 & \dots & \sigma_k \end{bmatrix}$

$\Rightarrow$ Learn both V_k and Σ_k by backpropagation
$\Rightarrow$ Freeze V_k, learn Σ_k (initialized to ones)

Experiments on Weighted Convolutions

Trained with reduced trained set, avg accuracy over 3 replicates

VGG on CIFAR100

	Conv	WConv
Avg	0.447	**0.454**
Std	0.006	0.0064

ResNet on CIFAR100

	Conv	WConv
Avg	**0.612**	0.609
Std	0.003	0.001

ResNet on Tiny ImageNet

	Conv	WConv
Avg	0.476	**0.479**
Std	0.003	0.006

Finetuning with reduced trained set, avg accuracy over 3 replicates

Inception on CIFAR100

	Conv	WConv
Avg	0.633	**0.636**
Std	0.001	0.001

ResNet on Tiny ImageNet

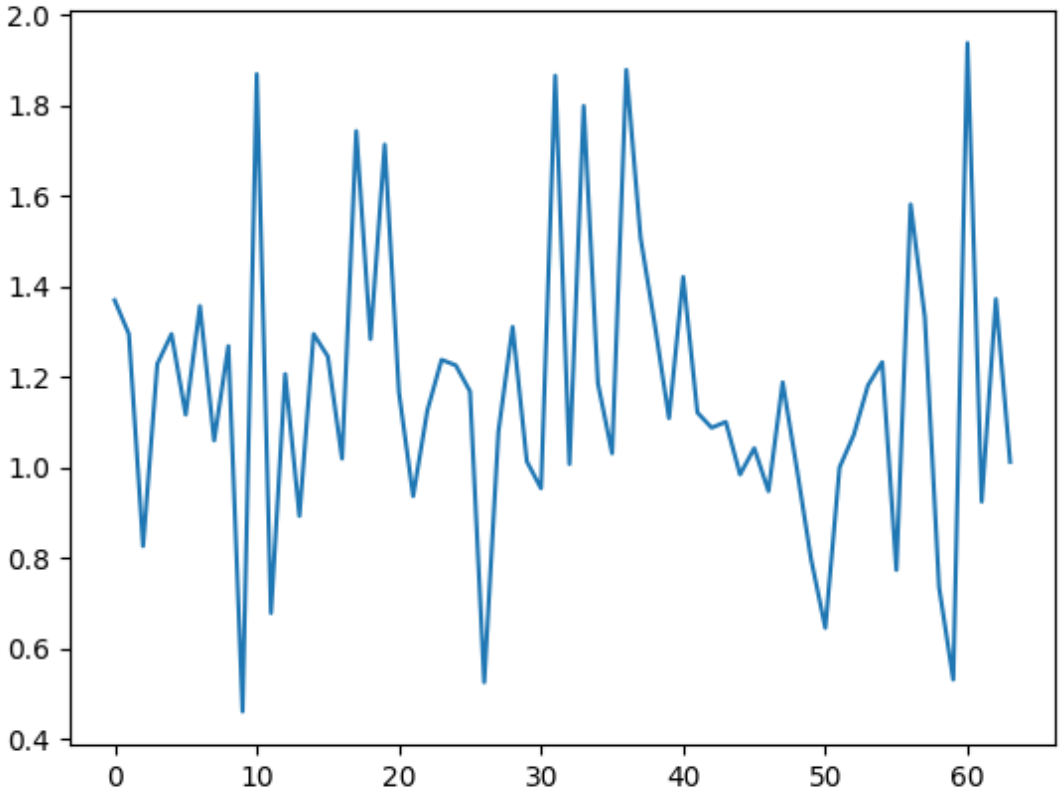
	Conv	WConv
Avg	0.678	**0.683**
Std	0.001	0.001

Autoencoder on CIFAR
Avg MSE over 3 replicates

	Conv	WConv	WConv2
Avg	0.0062	**0.0055**	0.0062
Std	0.0002	0.0003	0.0024

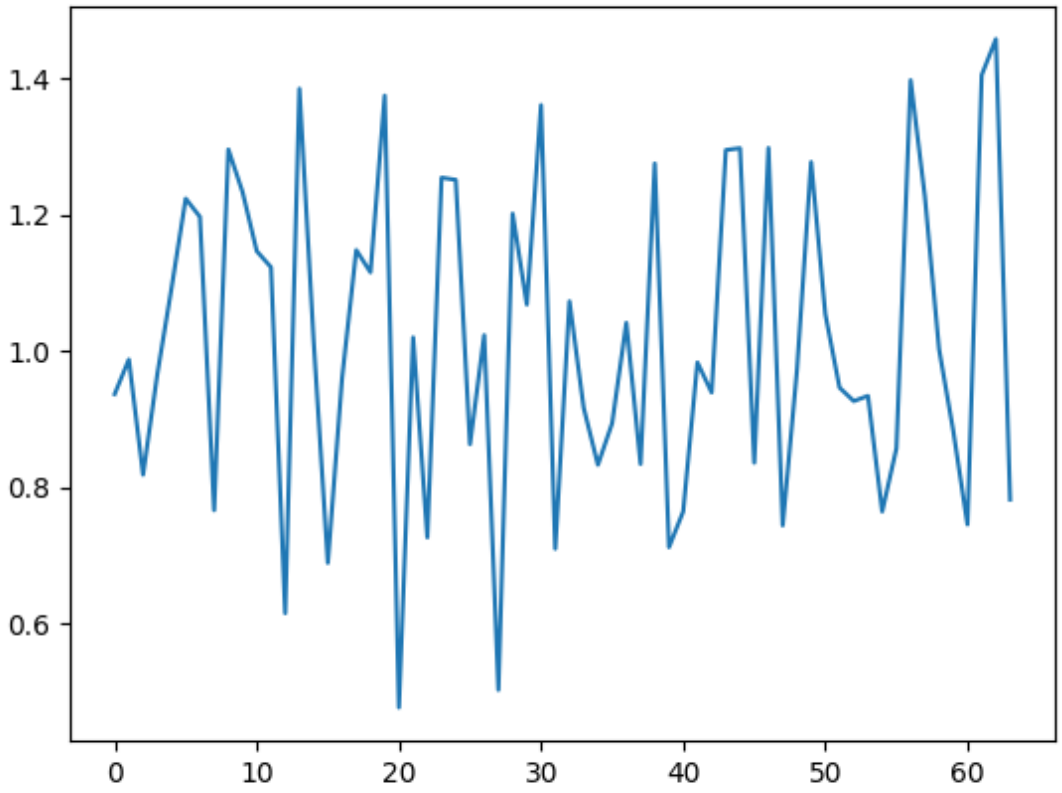
Weights of filters by layer

First conv layer



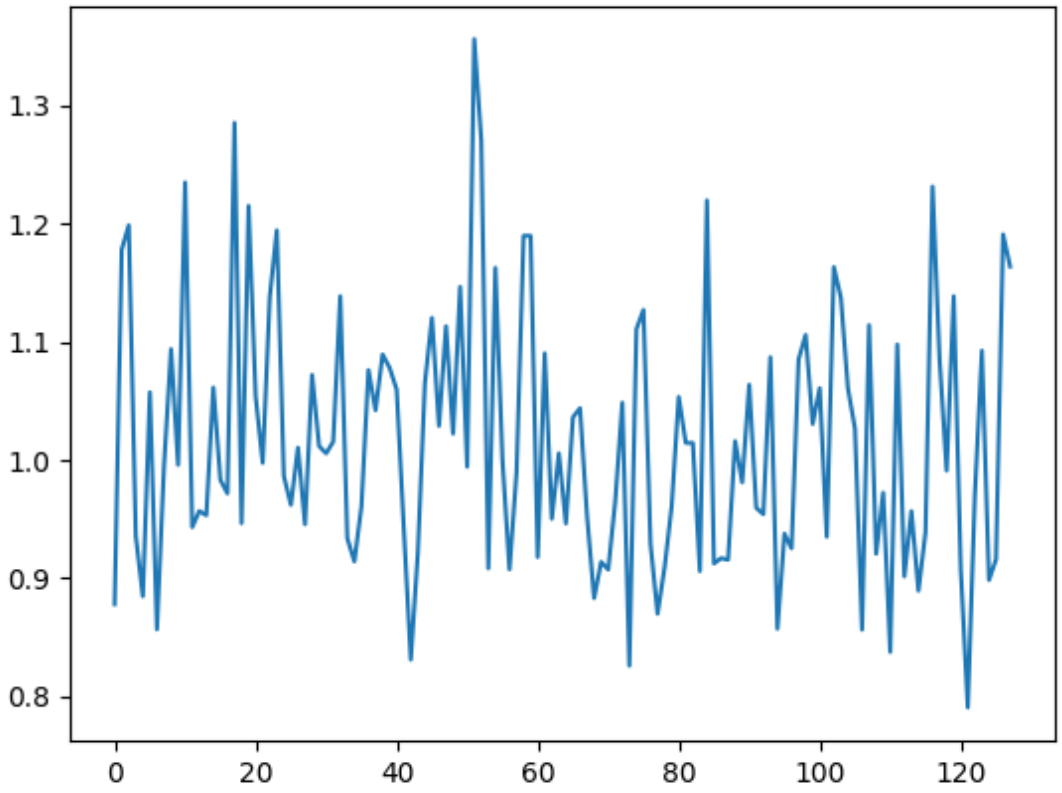
Weights of filters by layer

Second conv layer



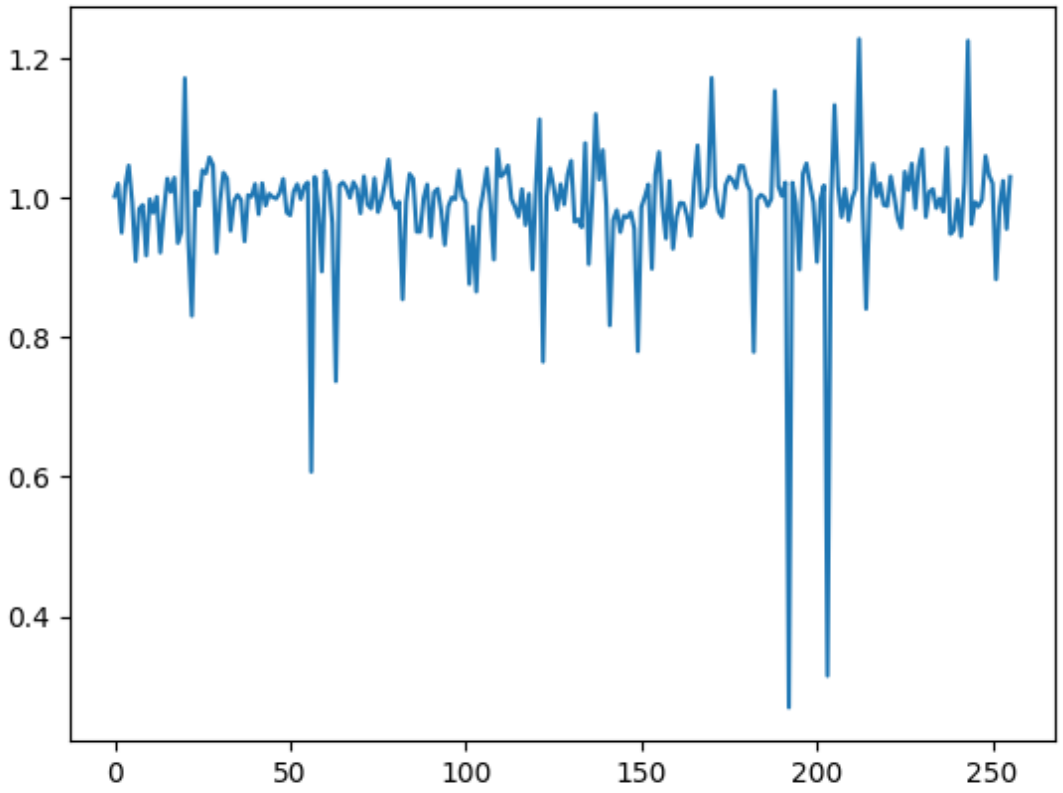
Weights of filters by layer

Third conv layer



Weights of filters by layer

Fourth conv layer



Outline

1. Intro
2. MLPs
3. Architectures
4. Understanding Convolution
5. Conclusion

Conclusion

- Deep Learning is everywhere, many opportunities for designing new systems, solving new tasks
- Huge spread of Machine Learning and Deep Learning ideas: Neurosciences, Security, Health, HEP...
- On the contrary ow what many people say and think
  - It is not so easy (except if you reuse the code (= the design and the learning) from someone else)
  - Many thinks to understand
  - Many things to invent

Réferences

- Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, Neural Machine Translation by Jointly Learning to Align and Translate. CoRR abs/1409.0473 (2014)
- Mickaël Chen, Ludovic Denoyer, Thierry Artières, Multi-View Data Generation Without View Supervision. ICLR, 2018.
- Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, Yann LeCun, The Loss Surfaces of Multilayer Networks. AISTATS 2015
- Ronen Eldan, Ohad Shamir, The Power of Depth for Feedforward Neural Networks. COLT 2016: 907-940
- Clément Farabet, Camille Couprie, Laurent Najman, Yann LeCun: Learning Hierarchical Features for Scene Labeling. IEEE Trans. Pattern Anal. Mach. Intell. 35(8): 1915-1929 (2013)
- Fei Fei Li slides : http://cs231n.stanford.edu/
- Yaroslav Ganin, Victor S. Lempitsky, Unsupervised Domain Adaptation by Backpropagation. ICML 2015: 1180-1189
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition. CVPR 2016: 770-778
- Geoffrey E. Hinton, Oriol Vinyals, Jeffrey Dean, Distilling the Knowledge in a Neural Network. CoRR abs/1503.02531 (2015)
- Reducing the Dimensionality of Data with Neural Networks G. E. Hinton and R. R. Salakhutdinov, VOL 313 SCIENCE, 2006
- Sepp Hochreiter, Jürgen Schmidhuber, Long Short-Term Memory. Neural Computation 9(8): 1735-1780 (1997)
- Krizhevsky, Sutskever, Hinton, ImageNet Classification with deep convolutional neural networks NIPS 2012

Réferences

- Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, Yoshua, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, Arxiv, 2014
- LeCun, Bottou, Bengio and Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998
- Lecun tutorial ICML 2013: https://cs.nyu.edu/ yann/talks/lecun-ranzato-icml2013.pdf
- Gilles Louppe, Michael Kagan, Kyle Cranmer, Learning to Pivot with Adversarial Networks. NIPS 2017: 982-991
- Mehdi Mirza, Simon Osindero, Conditional Generative Adversarial Nets. CoRR abs/1411.1784 (2014)
- Guido F. Montúfar, Razvan Pascanu, KyungHyun Cho, Yoshua Bengio, On the Number of Linear Regions of Deep Neural Networks. NIPS 2014: 2924-2932
- Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, Nati Srebro, Exploring Generalization in Deep Learning. NIPS 2017: 5949-5958
- Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, Dmitry P. Vetrov, Tensorizing Neural Networks. NIPS 2015: 442-450
- Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, Jose M. Álvarez, Invertible Conditional GANs for image editing. CoRR abs/1611.06355 (2016)
- Alec Radford, Luke Metz, Soumith Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. CoRR abs/1511.06434 (2015)
- Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, Richard S. Zemel: Meta-Learning for Semi-Supervised Few-Shot Classification. CoRR abs/1803.00676 (2018)

Réferences

- Marco Túlio Ribeiro, Sameer Singh, Carlos Guestrin, Why Should I Trust You? Explaining the Predictions of Any Classifier. KDD 2016: 1135-1144
- Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, Yoshua Bengio, FitNets: Hints for Thin Deep Nets. CoRR abs/1412.6550 (2014)
- Shreyas Saxena, Jakob Verbeek, Convolutional Neural Fabrics. NIPS 2016: 4053-4061
- Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR abs/1409.1556 (2014)
- Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus, End-To-End Memory Networks. NIPS 2015: 2440-2448
- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Going deeper with convolutions. CVPR 2015: 1-9
- Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, Lior Wolf, DeepFace: Closing the Gap to Human-Level Performance in Face Verification. CVPR 2014: 1701-1708
- Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson, How transferable are features in deep neural networks? NIPS 2014: 3320-3328
- Matthew D. Zeiler, Rob Fergus, Visualizing and Understanding Convolutional Networks. ECCV (1) 2014: 818-833
- Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, Oriol Vinyals, Understanding deep learning requires rethinking generalization. CoRR abs/1611.03530 (2016)