Deep Learning

Caio Corro, Michèle Sebag
CNRS & Université Paris-Saclay

March 31st, 2023

Credit for slides: Yoshua Bengio, Yann Le Cun, Nando de Freitas, Christian Perone, Honglak Lee, Stéphane Canu, Paul-Louis Pröve, Fei-Fei Li

Types of Machine Learning problems

WORLD – DATA – USER

Observations	+ **Target**	+ Rewards
Understand	**Predict**	Decide
Code	**Classification/Regression**	Action Policy/Strategy
Unsupervised	**Supervised**	Reinforcement
LEARNING	**LEARNING**	LEARNING

Types of Machine Learning problems, 2

News

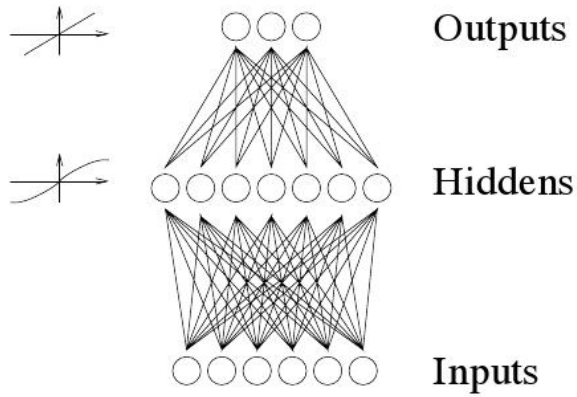
Good News: Neural Nets can be used for all three goals:

- Unsupervised learning — change of representation
- Supervised learning — achieves prediction
- Reinforcement learning — yields the state-action value

Bad News

- not so easy to learn — optimization
- not easy to understand — black-box model
- its extensions (to complex/higher order logic domains) require *finesse*

Neural Nets



(C) David McKay - Cambridge Univ. Press

History

1943	A neuron as a computable function $y = f(\mathbf{x})$ Intelligence $\rightarrow$ Reasoning $\rightarrow$ Boolean functions	Pitts, McCullough
1960	Connexionism + learning algorithms	Rosenblatt
1969	AI Winter	Minsky-Papert
1989	Back-propagation	Amari, Rumelhart & McClelland, LeCun
1995	Winter again	Vapnik
2005	Deep Learning	Bengio, Hinton

Properties of NN

Good news

- NN are universal approximators (activation function: sigmoids; Radius

For every decent function f ($= f^2$ has a finite integral on every compact of $\mathbb{R}^d$)
for every $\epsilon > 0$,
there exists some MLP/RBF g such that $||f - g|| < \epsilon$.

Bad news

- Not a constructive proof (the solution exists, so what ?)
- Everything is possible $\rightarrow$ no guarantee (overfitting).

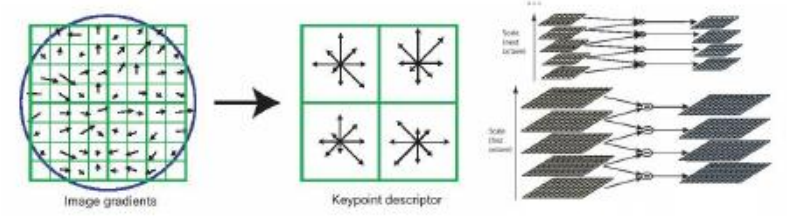
Very bad news

- A non convex (and hard) optimization problem
- Lots of local minima
- Low reproducibility of the results

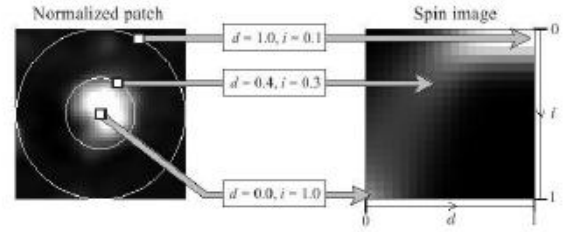
The Deep ML revolution: what is new ?

Former state of the art

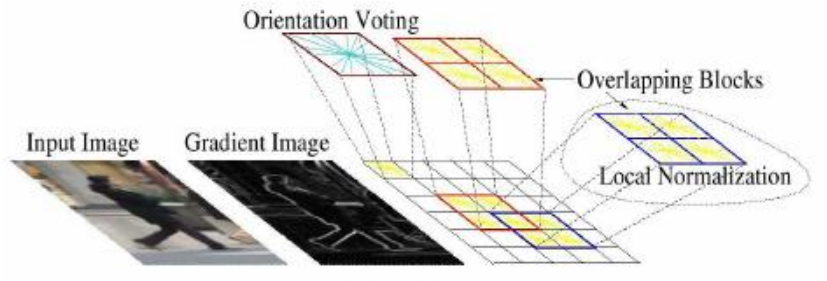
e.g. in computer vision



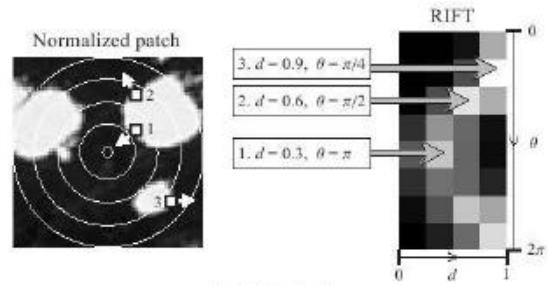
SIFT



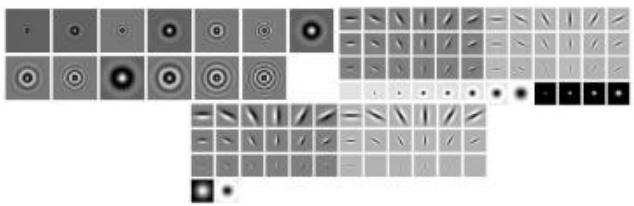
Spin image



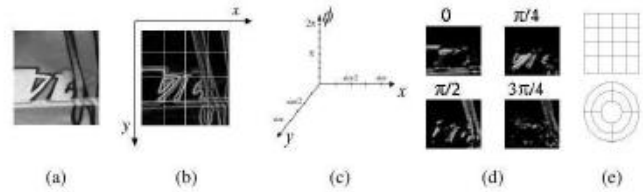
HoG



RIFT



Textons



GLOH

SIFT: scale invariant feature transform
HOG: histogram of oriented gradients

Textons: "vector quantized responses of a linear filter bank"

What is new, 2

Traditional approach

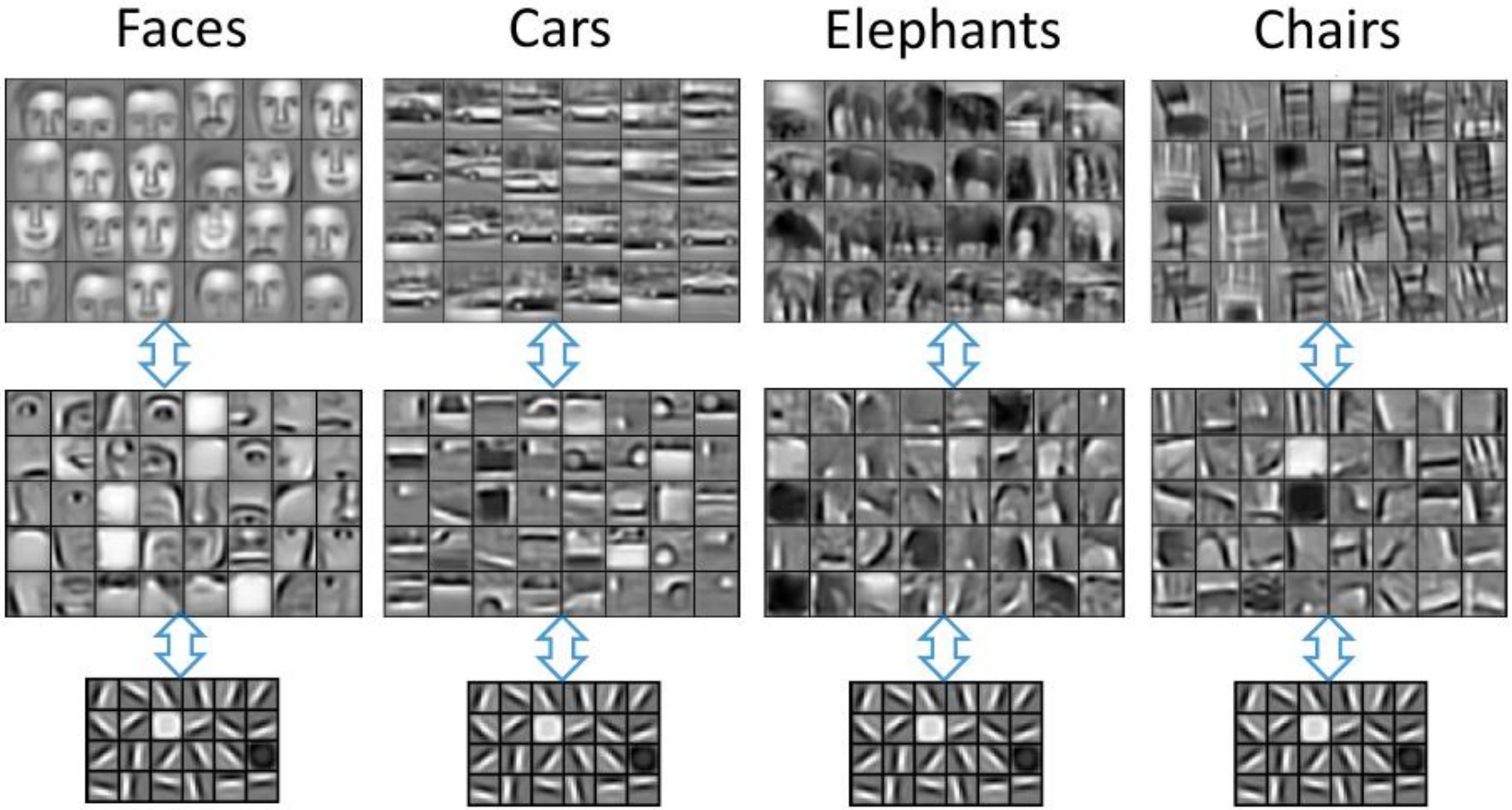
→ Manually crafted features → Trainable classifier

Deep learning

→ Trainable feature extractor → Trainable classifier

A new representation is learned

Bengio et al. 2006

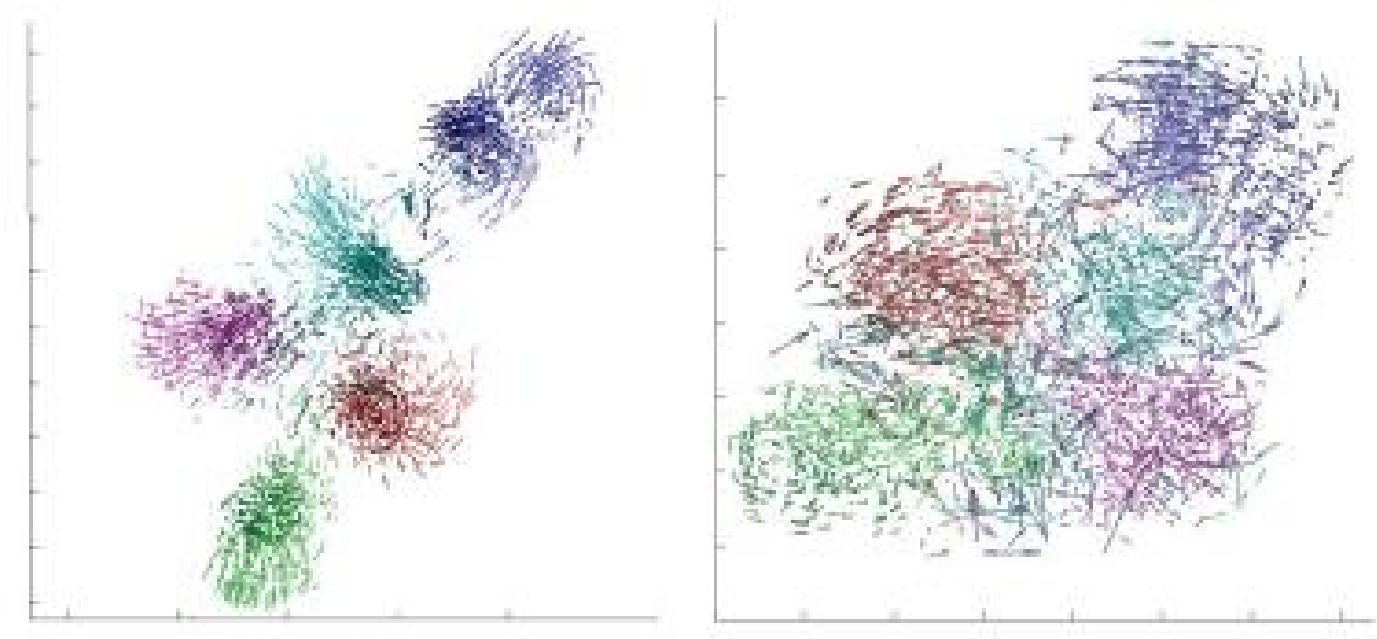


Good features ?

Le Cun, 2015: https://www.youtube.com/watch?v=Y-XTcTusUxQ

data (MNIST; each class, a color) $\rightarrow$ change of representation $\rightarrow$ visualisation

Comparing representations



Which representation is the best one ?

The CONs

Model selection / Auto Deep Learning

- Selecting architecture
- Adjusting learning criterion

Is More Better ?

avoid overfitting

Algorithmic choices

- Enforce stability
- Adjust the learning rate
- Stopping criterion

a difficult optimization problem

regularization)

Adam

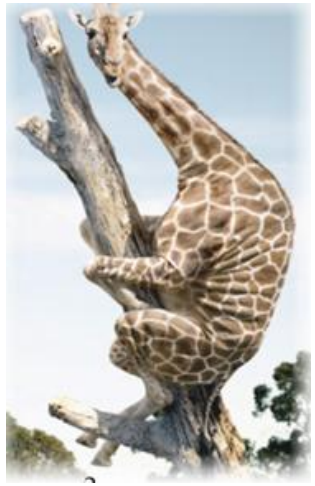
early stopping

Tricks

- Normalize data
- Initialize **W** small
  initialization options

Glorot Bengio, 2010 pay attention to the weight

Toward deeper representations



Invariances matter

- The label of an image is invariant through small translation, homothety, rotation...
- Invariance of labels $\rightarrow$ Invariance of model

$$y(x) = y(\sigma(x)) \rightarrow h(x) = h(\sigma(x))$$

Enforcing invariances

- by augmenting the training set:

$$\mathcal{E} = \{(x_i, y_i)\} \bigcup \{(\sigma(x_i), y_i)\}$$

- by structuring the hypothesis space

Convolutional networks

A bit of history:

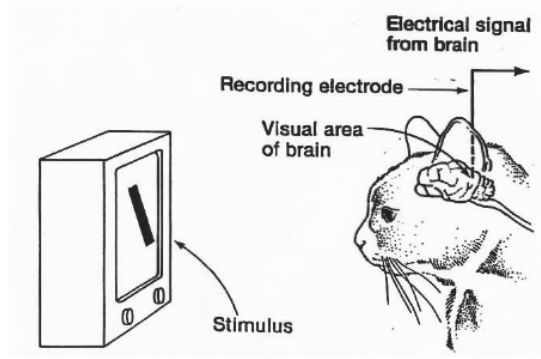
Hubel & Wiesel, 1959

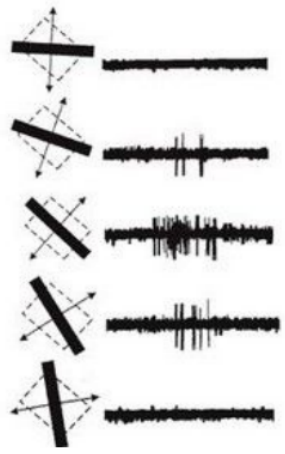
RECEPTIVE FIELDS OF SINGLE NEURONES IN
THE CAT'S STRIATE CORTEX

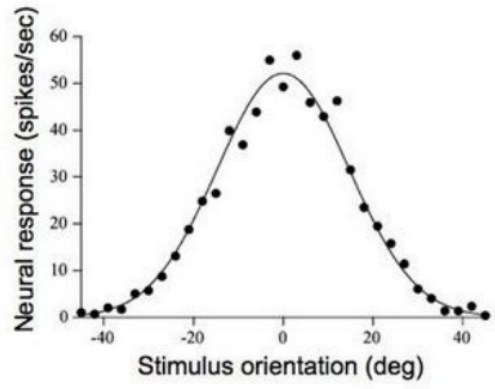
1962

RECEPTIVE FIELDS, BINOCULAR INTERACTION
AND FUNCTIONAL ARCHITECTURE IN
THE CAT'S VISUAL CORTEX

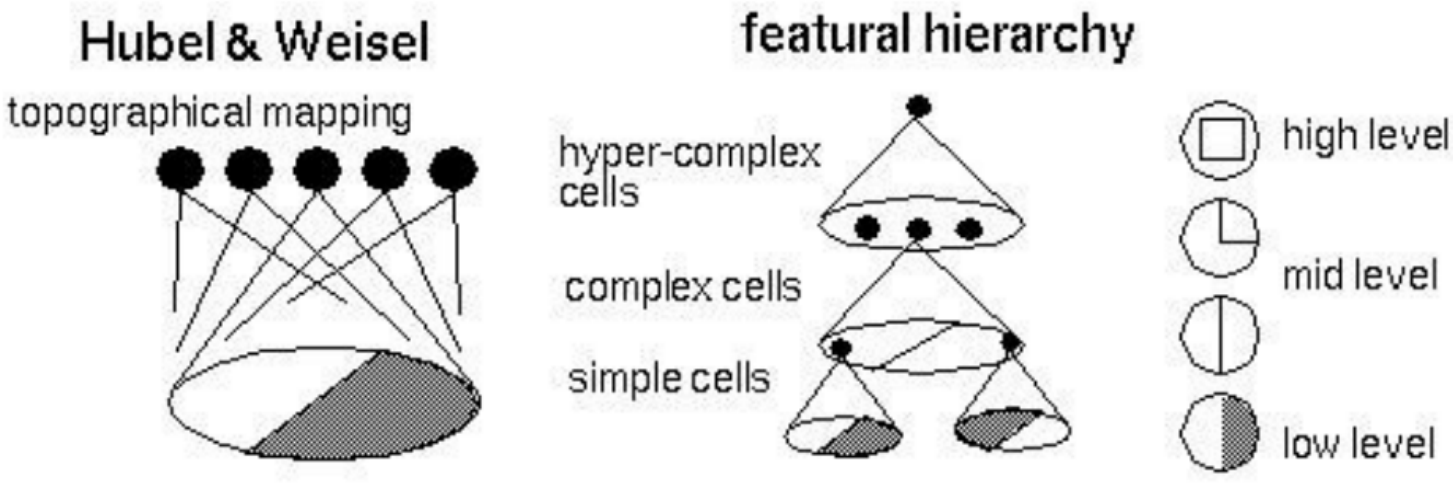
1968...







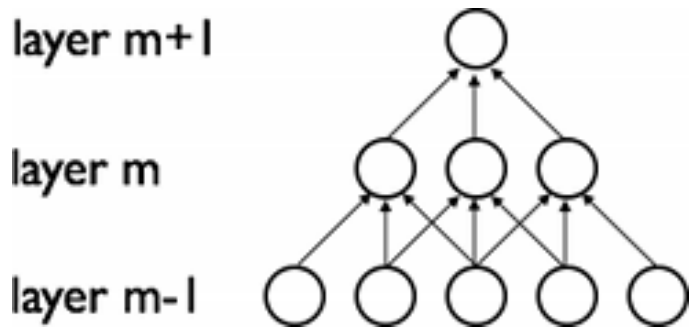
A hierarchical structure



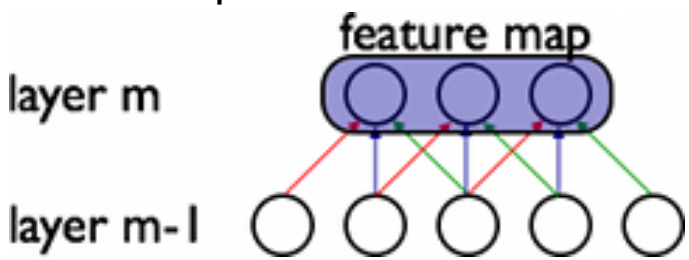
Hubel & Wiesel 1968

Visual cortex of the cat

- cells arranged in such a way that
- ... each cell observes a fraction of the visual field — receptive field
- ... their union covers the whole field



- Layer m: detection of local patterns — (same weights)



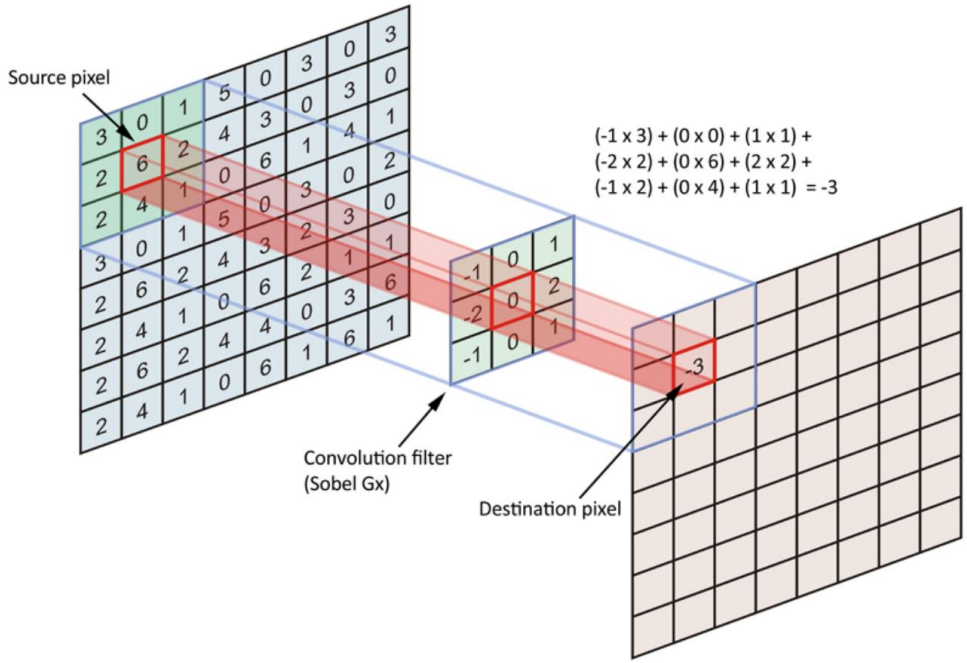
- Layer $m+1$: non linear aggregation of output of layer m

Ingredients of convolutional networks

called...

1. Local receptive fields

Kernel or Filter

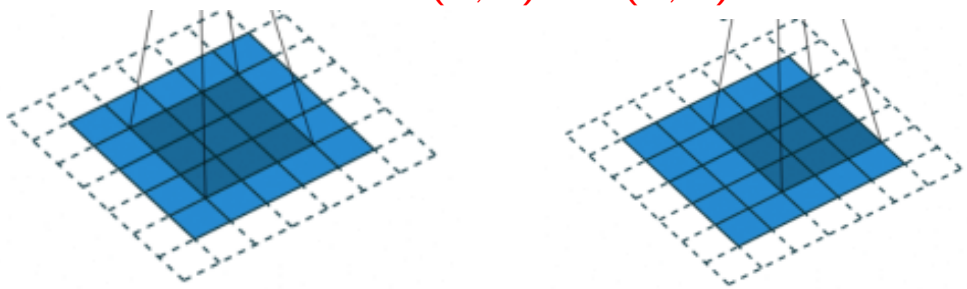


Each patch x in the image $\rightarrow$ multiplied by filter w

$$\langle w, x \rangle + b$$

2. Move: apply the kernel to $x + (0, 1)$ or $(1, 0)$

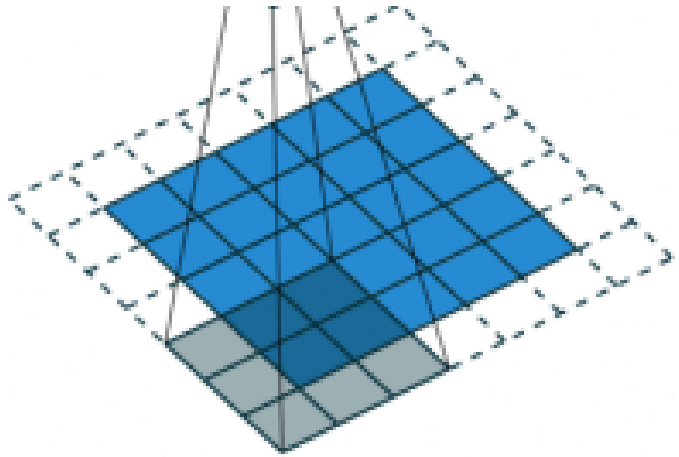
Stride



Ingredients of convolutional networks, followed

3. Beware when you go to the border

Padding



4. Weight sharing

You apply the same filter on all x+strides.
Learning: adaptation of the gradient-based update: the update is averaged over all occurrences of the filter.
Reduces the number of parameters by orders of magnitude

Ingredients of convolutional networks, followed

5. Aggregating the output

Pooling

Next layer: returns the average or max of

$$\rangle w, x_i \langle + b$$

where x_i ranges in $x + 1, 2, 3....$ strides.

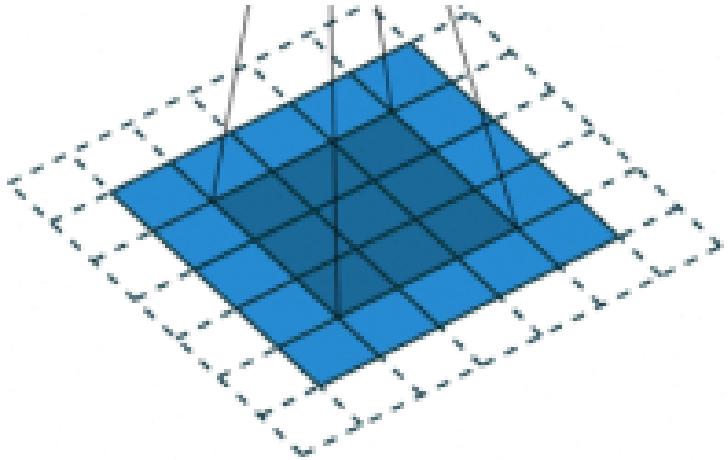
- Overlapping / non-overlapping regions
- Return the max / the sum of the feature map over the region
- Larger receptive fields (see more of input)

Convolutional networks: Glossary 1/2

https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

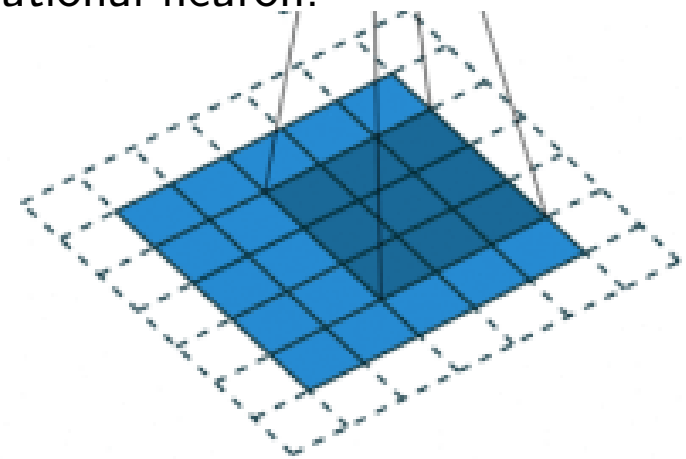
Kernel size (3)

Patch handled by the convolutional neuron, a.k.a filter



Stride (1)

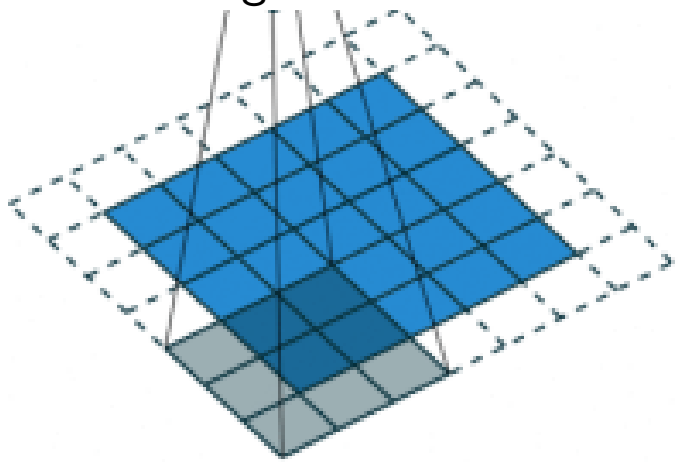
the move of the convolutional neuron.



Convolutional networks: Glossary 2/2

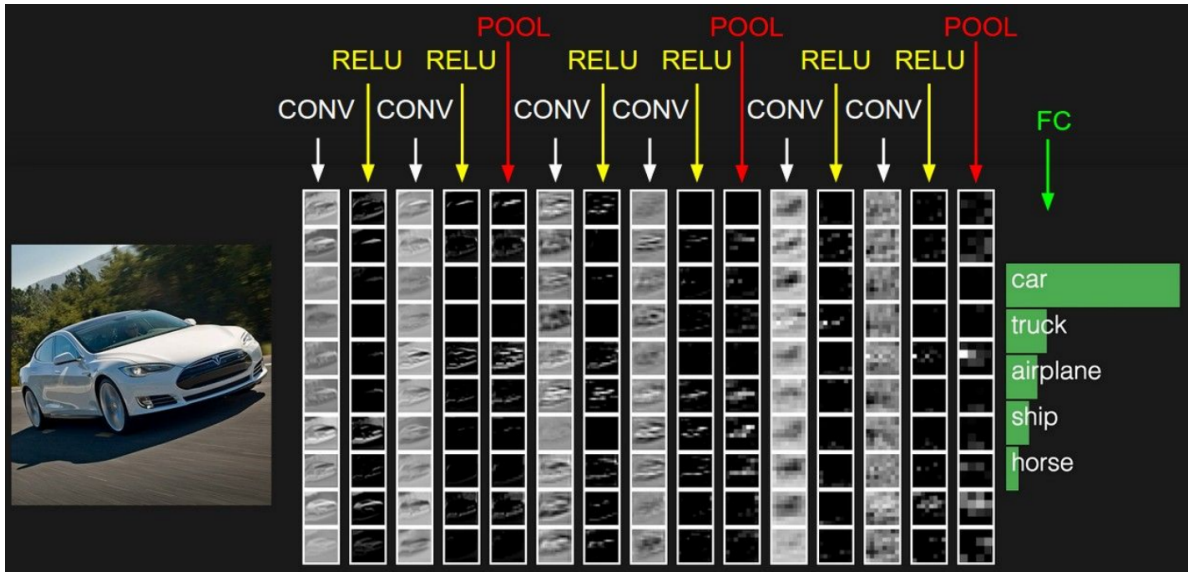
Padding (with 0; or by symmetry)

Handling the boundaries of the image.

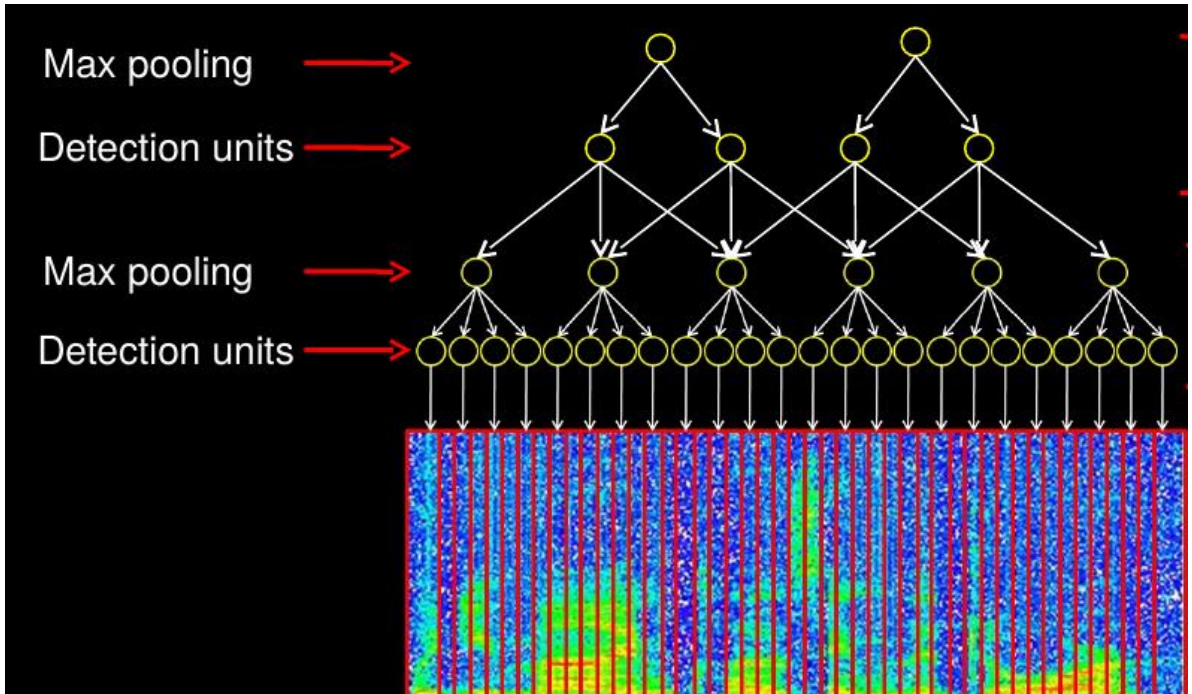


Convolutional networks in action

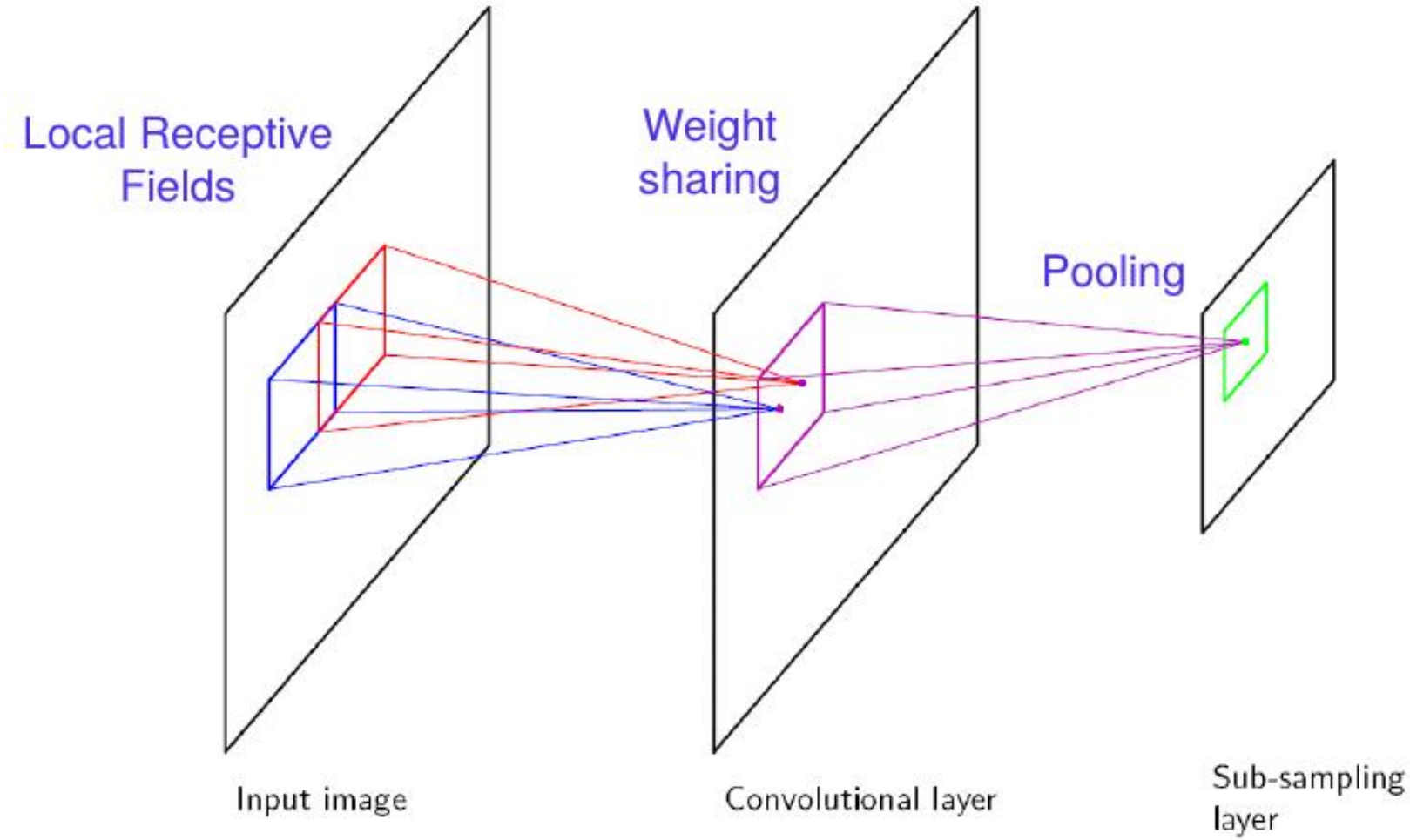
Computer vision



Music



Convolutional networks, summary

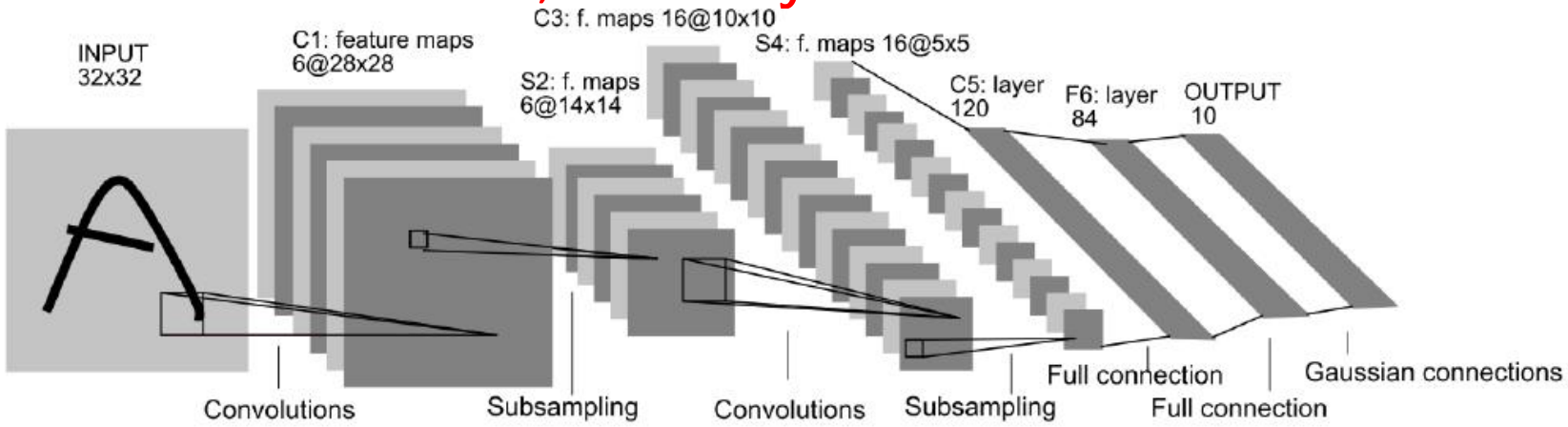


LeCun 1998

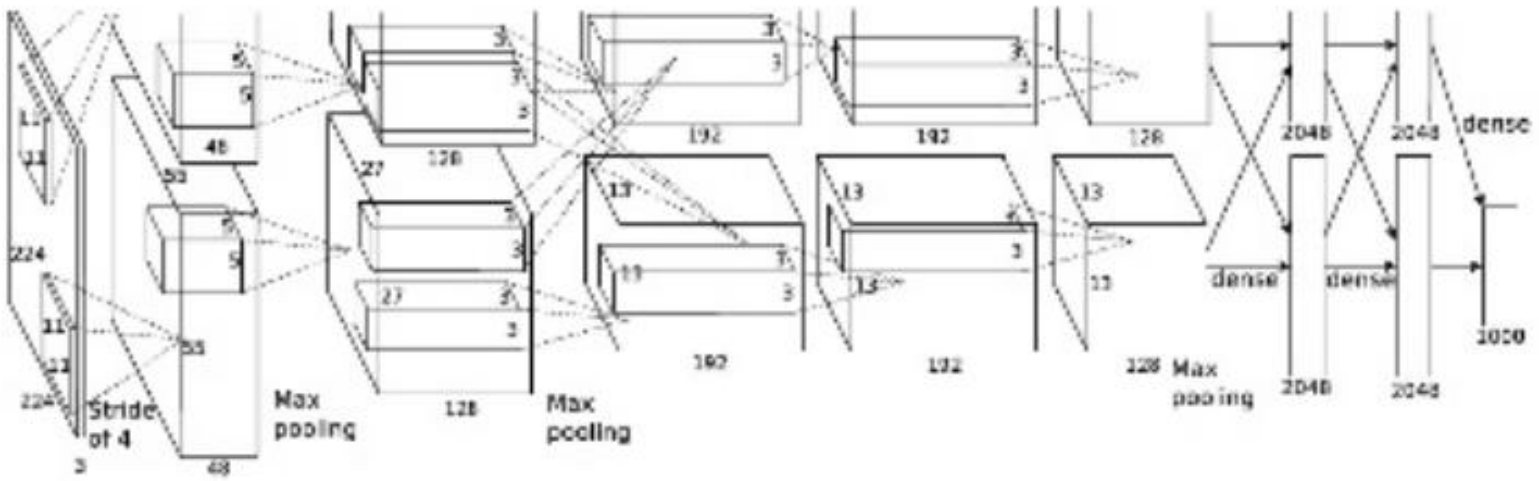
Properties

- Invariance to small transformations (over the region)
- Reducing the number of weights

Convolutional networks, summary



LeCun 1998



Kryzhevsky et al. 2012

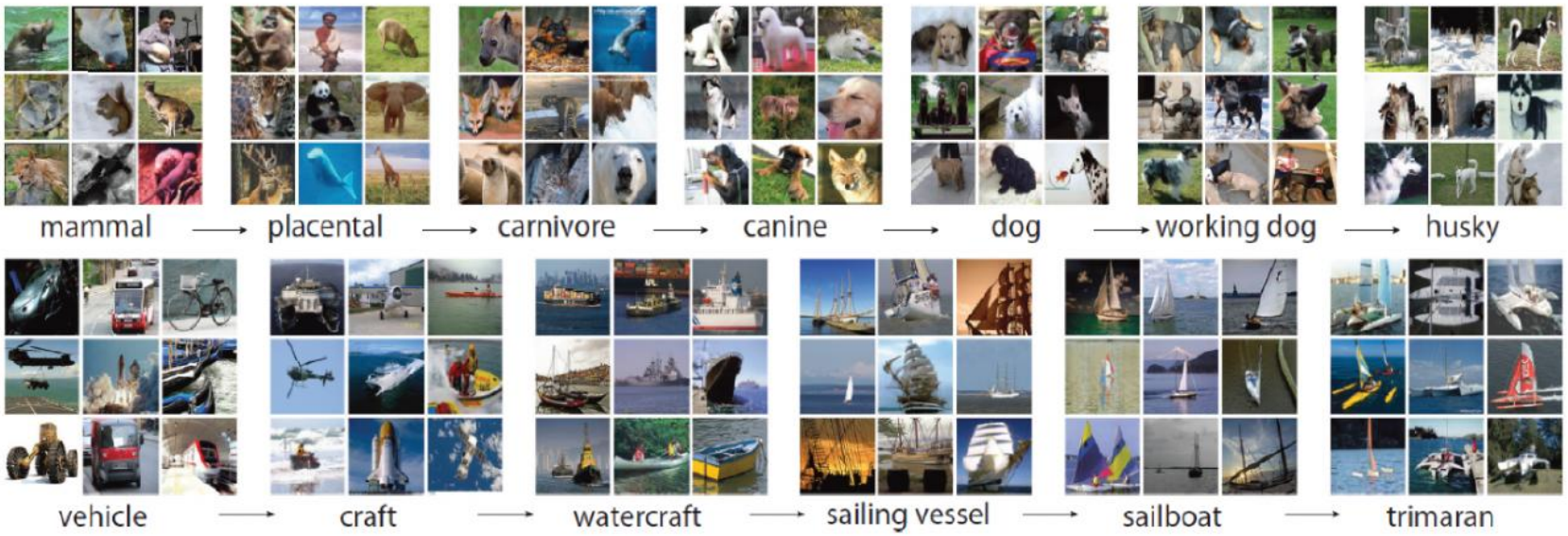
Properties

- Invariance to small transformations (over the region)
- Reducing the number of weights
- Usually many convolutional layers

ImageNet

Deng et al. 12

15 million labeled high-resolution images; 22,000 classes.



Large-Scale Visual Recognition Challenge

- 1000 categories.
- 1.2 million training images,
- 50,000 validation images,
- 150,000 testing images.

A leap in the state of the art

2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

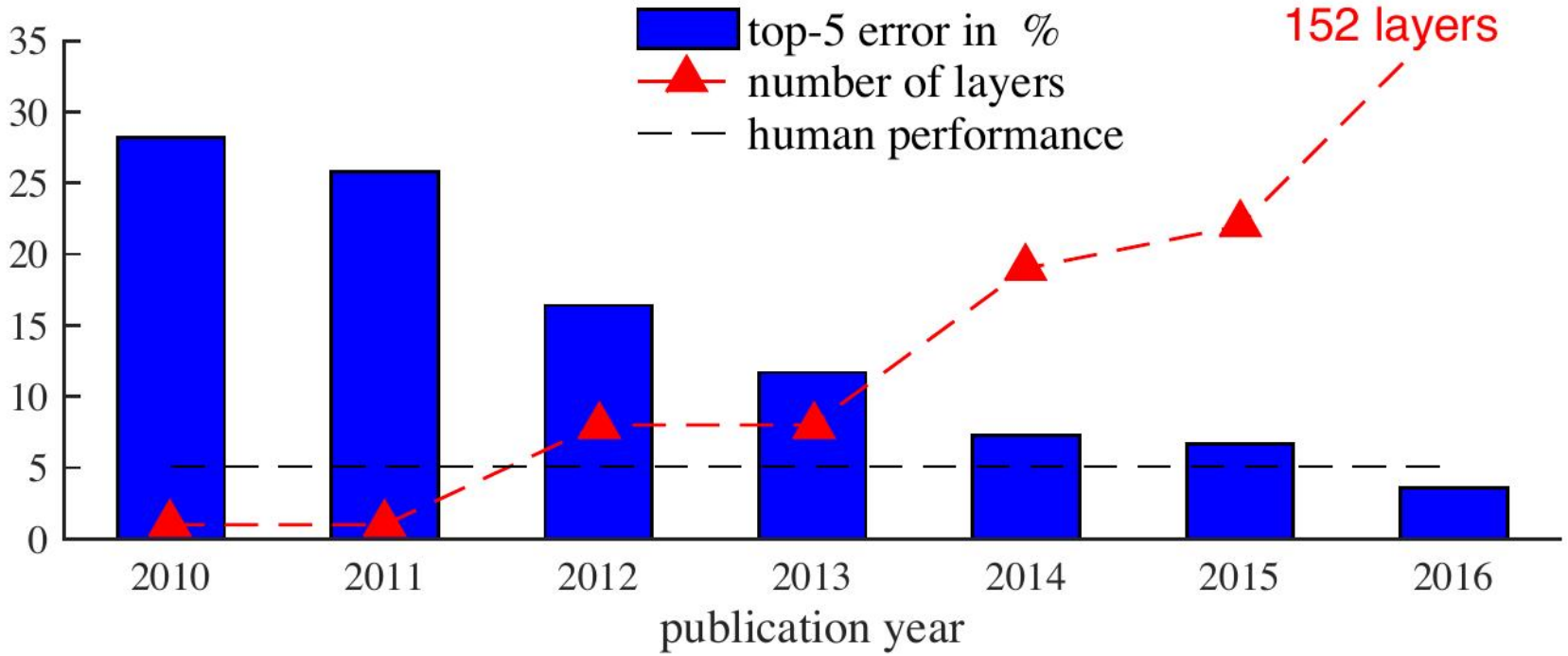
2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

shallow approaches

deep learning

Y. LeCun StatLearn tutorial

Super-human performances



2012 Alex Net

2013 ZFNet

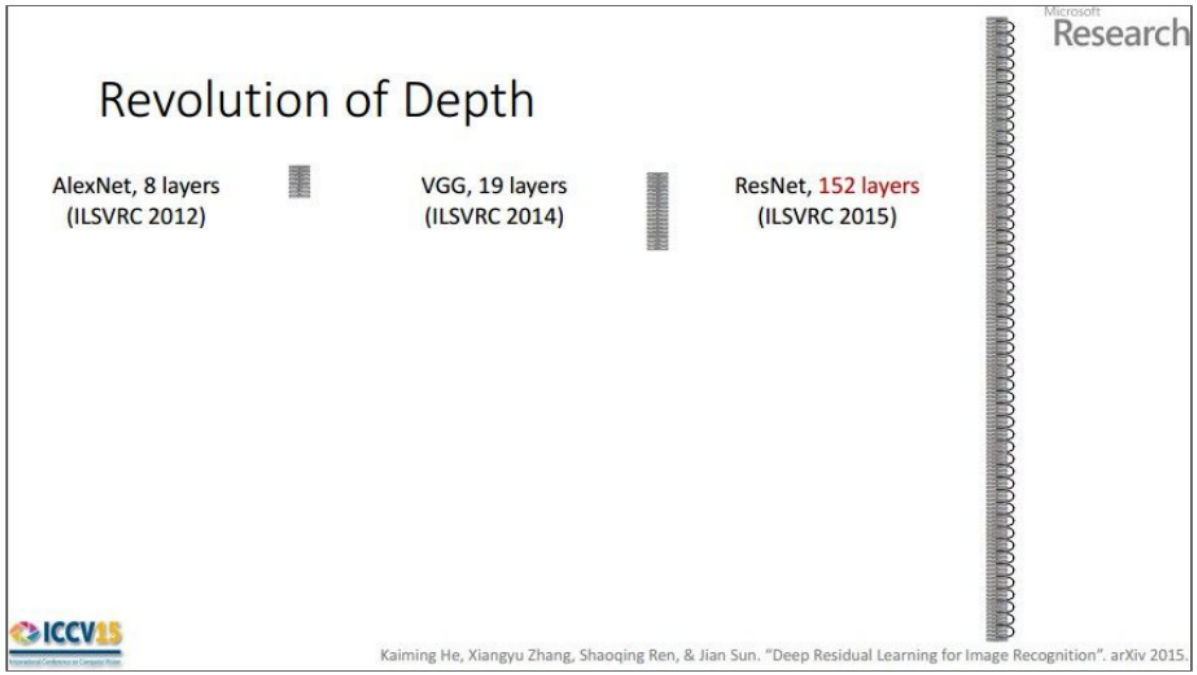
2014 VGG

2015 GoogLeNet / Inception

2016 Residual Network

karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

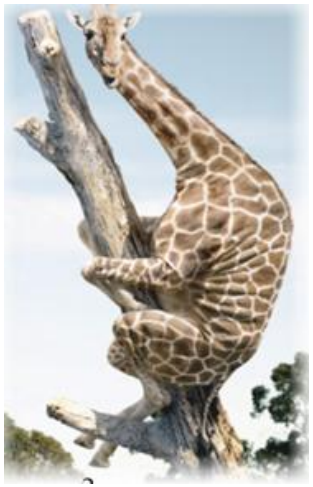
The revolution of depth



2-3 weeks of training on 8 GPU machine

at runtime: faster than a VGGNet! (even though it has 8x more layers)

Data augmentation



Enforcing invariances

- by **augmenting the training set**:

$$\mathcal{E} = \{(x_i, y_i)\} \bigcup \{(\sigma(x_i), y_i)\}$$

- by structuring the hypothesis space

Data augmentation, 2

Other invariance operators

- symmetry wrt vertical axis

- invariance wrt noise in the boundaries

Data augmentation

- Add samples generated from true samples, with (likely) same label

Exercize

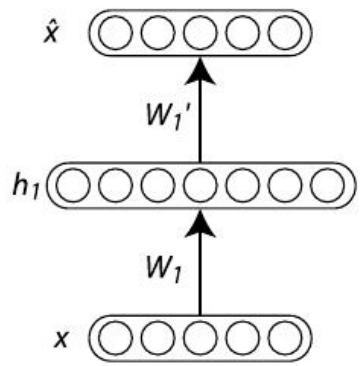
- Which data augmentation would you recommend for musical signals ?

Auto-encoders

$$\mathcal{E} = \{\mathbf{x}_i \in \mathbb{R}^D, i = 1 \dots n\}$$
$$\mathbf{x} \longrightarrow h_1 \in \mathbb{R}^d \longrightarrow \hat{\mathbf{x}}$$

- An auto-encoder:

$$\text{Find } W^* = \arg\min_W \left(\sum_i ||W' o W(\mathbf{x}_i) - x_i||^2 \right)$$



(*) Instead of min squared error, use cross-entropy loss:

$$\sum_j \mathbf{x}_{i,j} \log \hat{\mathbf{x}}_{i,j} + (1 - \mathbf{x}_{i,j}) \log (1 - \hat{x}_{i,j})$$

(**) Why W for encoding and W' for decoding ?

Auto-encoders and Principal Component Analysis

Assume

- A single layer
- Linear activation

Then

- An auto-encoder with k hidden neurons $\approx$ first k eigenvectors of PCA

Why ?

Stacked auto-encoders were used to initialize deep networks

In the early Deep Learning era...

Bengio, Lamblin, Popovici, Larochelle 06

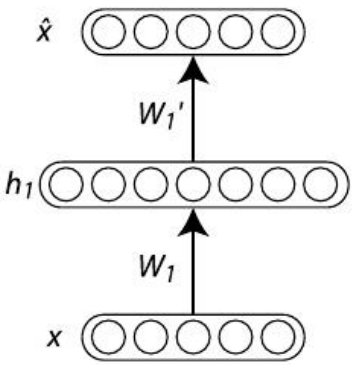
First layer

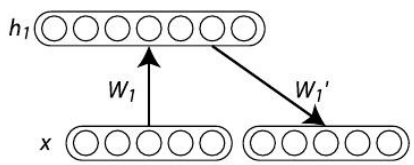
$$\mathbf{x} \longrightarrow \mathbf{h}_1 \longrightarrow \hat{\mathbf{x}}$$

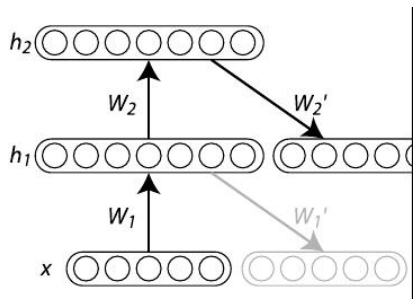
Second layer

same, replacing $\mathbf{x}$ with $\mathbf{h}_1$

$$\mathbf{h}_1 \longrightarrow \mathbf{h}_2 \longrightarrow \hat{\mathbf{h}_1}$$







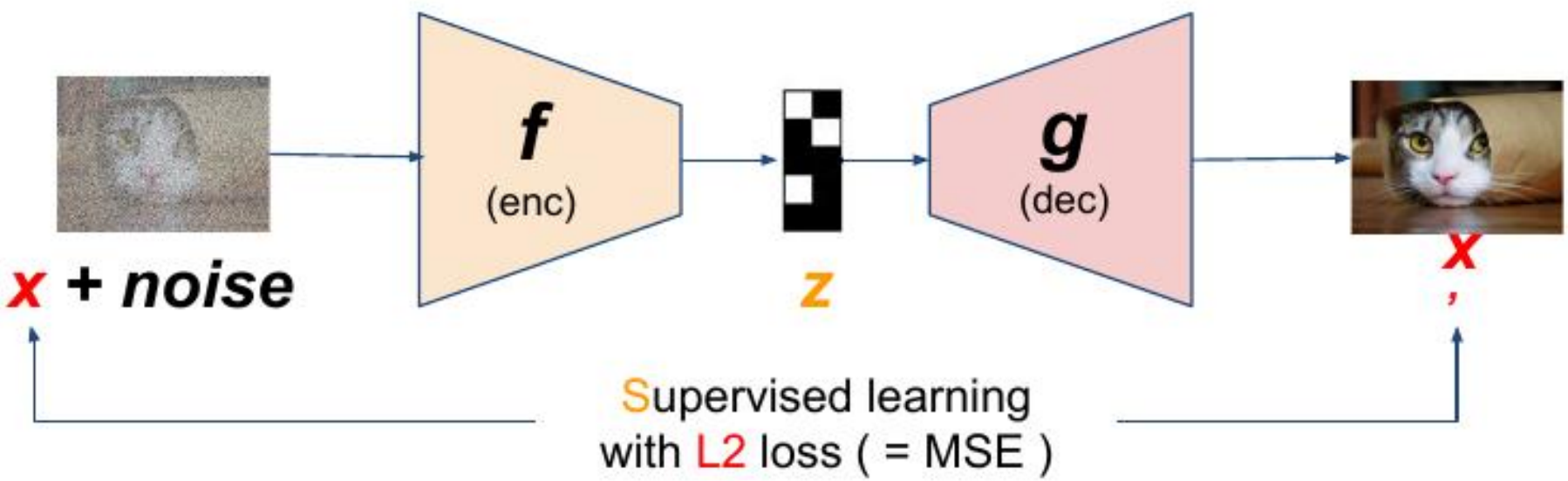
Denoising Auto-Encoders

Vincent, Larochelle, Bengio, Manzagol, 08

Principle

- Add noise to **x**
  - Gaussian noise
  - Or binary masking noise (akin drop-out)
- Recover **x**.

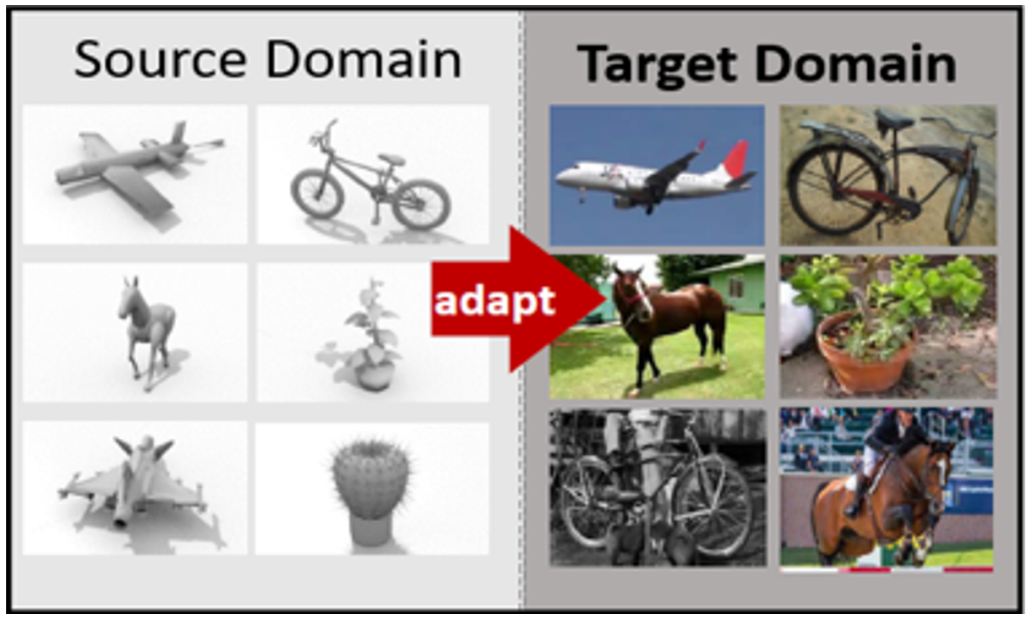
$$\text{Find } W^* = \arg\min_W \left(\sum_i ||W' o W(\mathbf{x}_i + \epsilon) - x_i||^2 \right)$$



Domain Adaptation

Context

- Source domain (sufficient labelled data)
- Target domain (insufficient data and/or insufficient labels)



Goal

- Build model for target domain by leveraging (data or model) from source domain
- Wanted: better performances; or faster learning

Auto-encoders for domain adaptation

Stacked Denoising Auto-Encoders

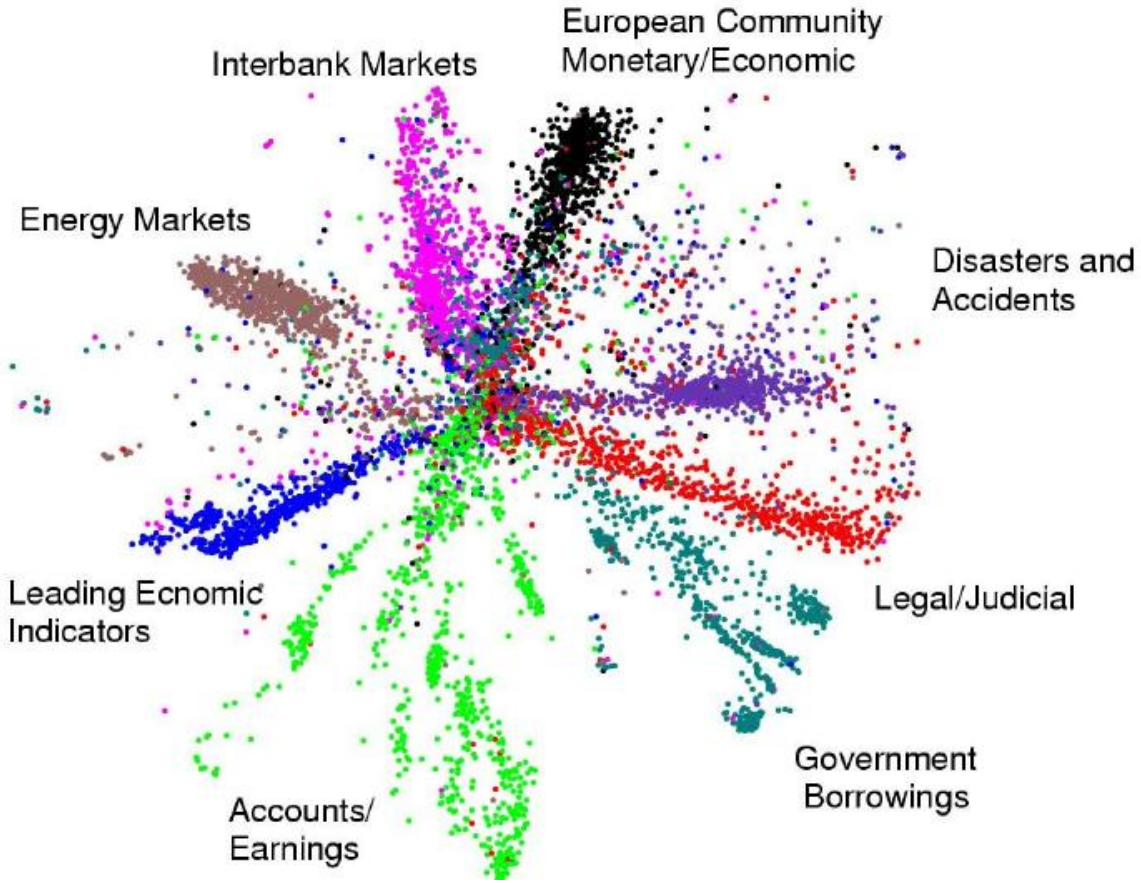
- on source and target instances
- use latent representation to learn on source domain

Why should it work ?

SDAs are able to disentangle hidden factors which explain the variations in the input data, and automatically group features in accordance with their relatedness to these factors. This helps transfer across domains as these generic concepts are invariant to domain-specific vocabularies.

Visualization with (non linear) Autoencoders

For $d = 2$,



dimensionality reduction on the latent representation

- Multidimensional scaling
- t-Distributed Stochastic Neighbor Embedding (t-SNE)

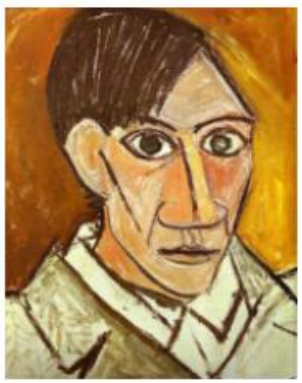
vdMaaten, Hinton 08

t-SNE Do and Don't https://distill.pub/2016/misread-tsne/

Morphing of representations

Gatys et al. 15, 16

Used for *Content*



Used for *Style*

Decrease α/β

- Style and contents in a convolutional NN are separable
- Use a trained VGG-19 Net:
  - applied on image 1 (content)
  - applied on image 2 (style)
  - find input matching hidden representation of image 1 (weight α) and hidden representation of image 2 (weight β)

The style

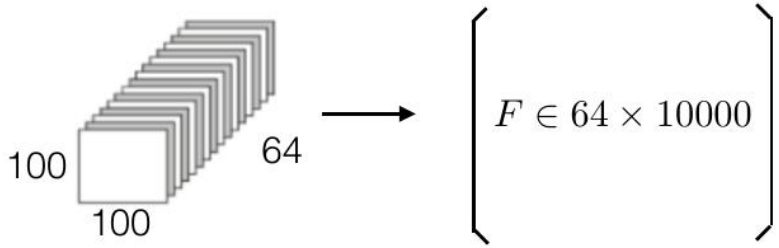
Gatys et al. 15, 16

- Style representation: **correlations** between the different filter responses over the spatial extent of feature maps.

  — Provide colours and local structures.

- Synthesize texture by matching correlation matrices calculated from different layers.

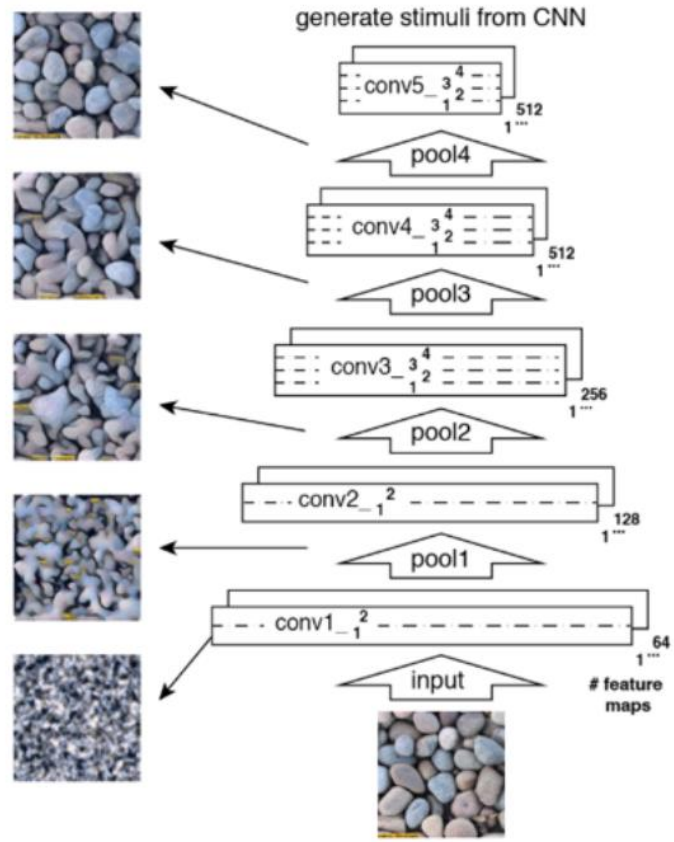
- Key equations: (Check paper for notation)



$$G^l = F^l (F^l)^T$$ Correlation matrix

$$E_l = \frac{1}{Norm} \sum_{i,j} (G^l_{ij} - A^l_{i,j})^2$$ Cost for style reconstruction

$$Loss_{style} = \sum_{l=0}^{L} w_l E_l$$ Accumulate cost for lower layers

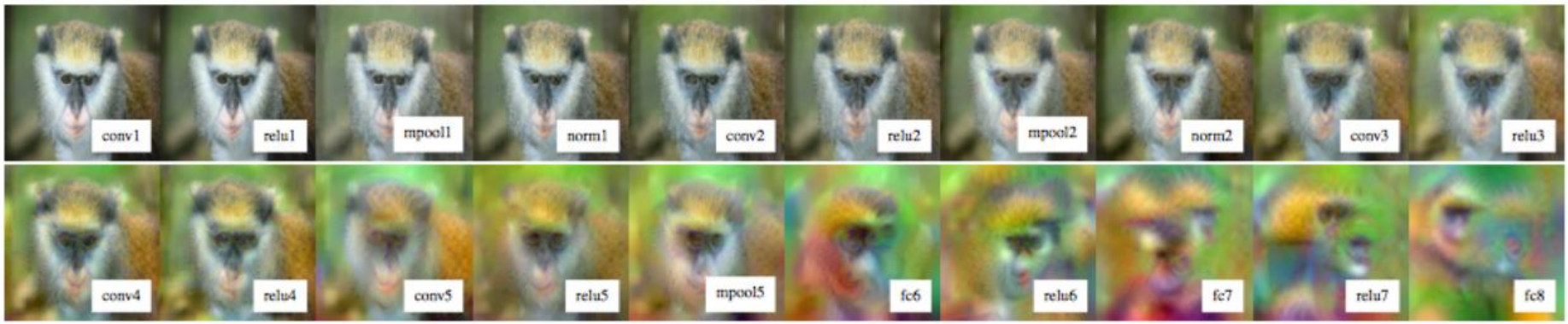


Portilla & Simoncelli, 2000

Gatys, et al. 2015

The content

Gatys et al. 15, 16



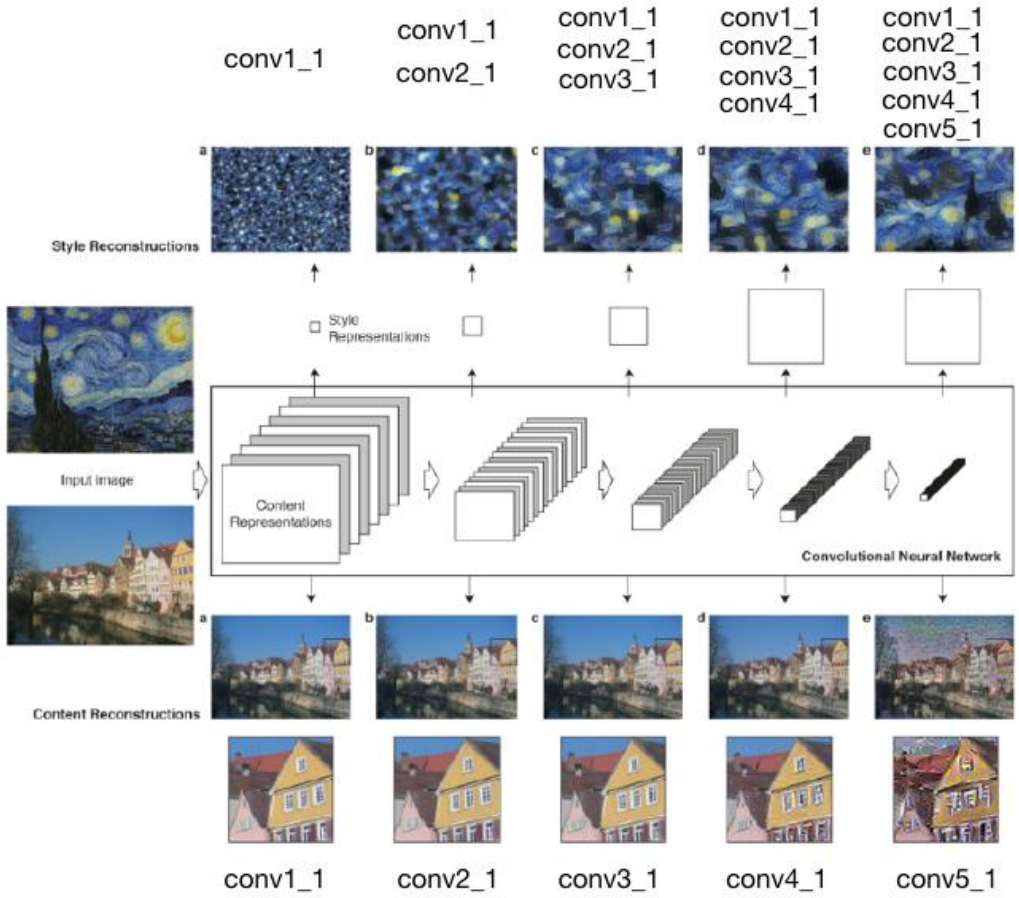
Finally

Use image $\mathbf{x}_0$ for content (AE ϕ), $\mathbf{x}_1$ for style (AEϕ')
Morphing: Find input image $\mathbf{x}$ minimizing

$$\alpha\|\phi(\mathbf{x}) - \phi(\mathbf{x}_0)\| + \beta\langle\phi'(\mathbf{x}), \phi'(\mathbf{x}_1)\rangle$$

Morphing of representations, 2

Gatys et al. 15, 16

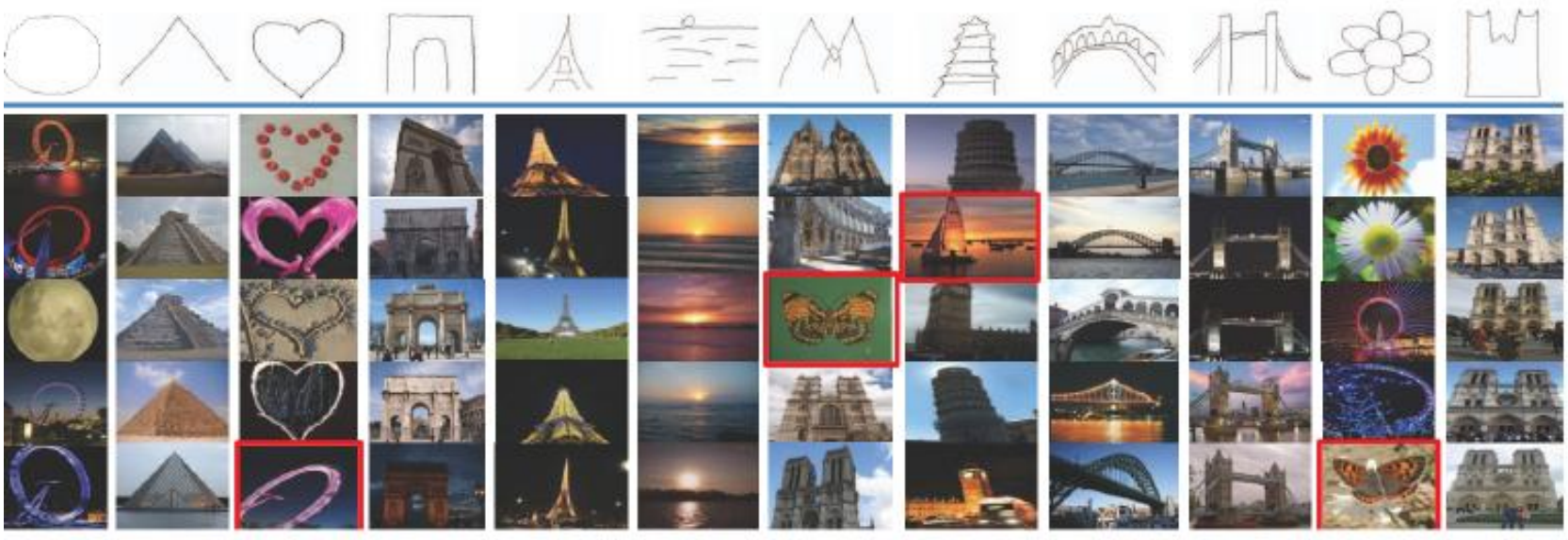


- Contents (bottom): convolutions with decreasing precision
- Style (top): correlations between the convol. features

Morphing of representations, 2

Gatys et al. 15, 16

Siamese Networks



Classes or similarities ?

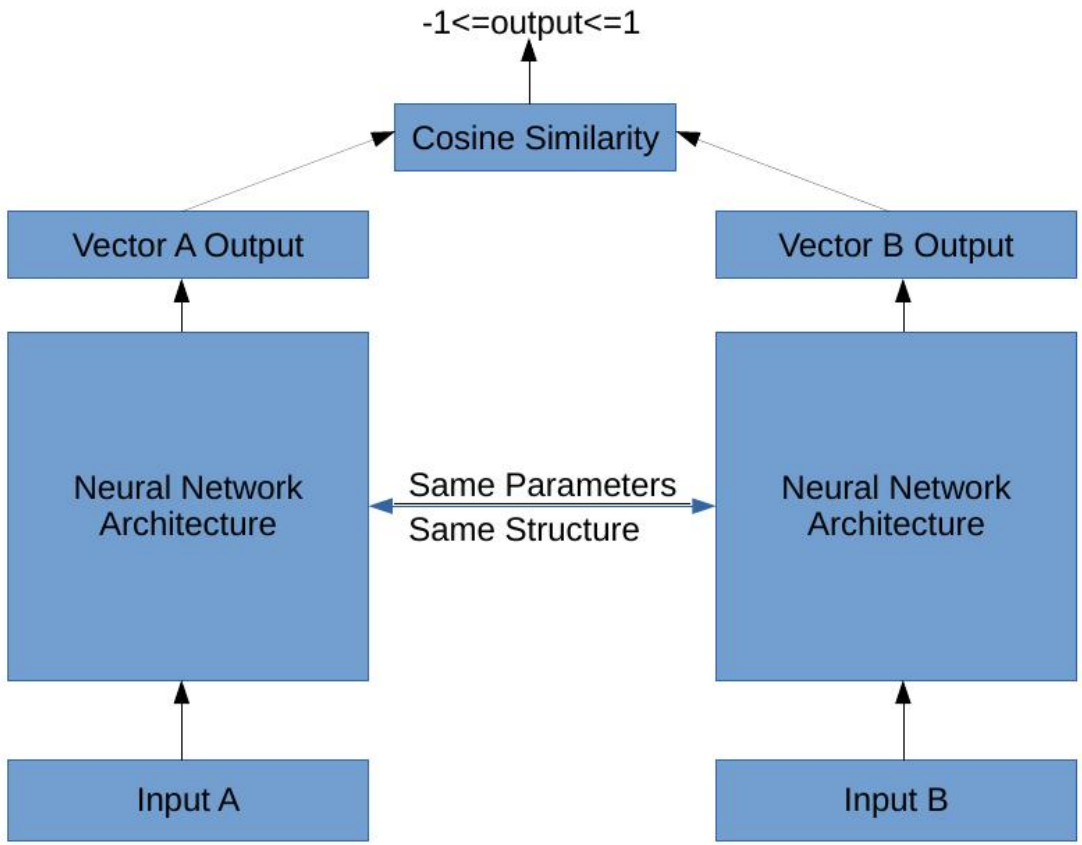
Siamese Networks

Bromley et al. 93

Principle

- Neural Networks can be used to define a latent representation
- Siamese: optimize the related metrics

Schema



Siamese Networks, 2

Data

$$\mathcal{E} = \{x_i \in \mathbb{R}^d, i \in [[1, n]]\}; \mathcal{S} = \{(x_{i,\ell}, x_{j_\ell}, c_\ell) \ s.t. \ c_\ell \in \{-1, 1\}, \ell \in [[1, L]]\}$$

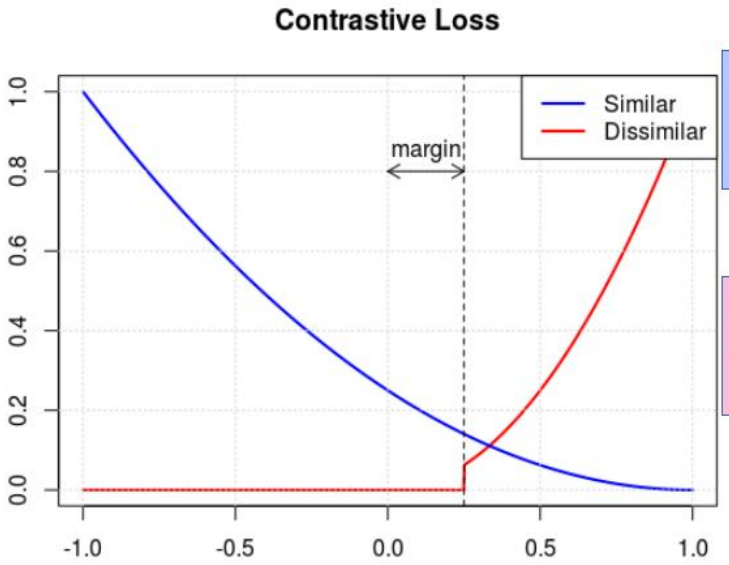
Experimental setting

- Often: few similar pairs; by default, pairs are dissimilar
- Subsample dissimilar pairs (optimal ratio between 2/1 ou 10/1)
- Possible to use domain knowledge in selection of dissimilar pairs

Loss

Given similar and dissimilar pairs (E_+ and E_-)

$$\mathcal{L} = \sum_{(i,j) in E_+} L_+(i,j) + \sum_{(k,\ell) in E_-} L_-(k,\ell)$$



$$L_+ = \frac{1}{4} \cdot (1 - cosine(x_1, x_2))^2$$

$$L_- = \begin{cases} \text{cosine}(x_1, x_2)^2, & \text{if } x \geq \text{margin} \\ 0, & \text{otherwise} \end{cases}$$

Applications

- Signature recognition
- Image recognition, search
- Article, Title
- Filter out typos
- Recommandation systems, collaborative filtering

Siamese Networks for one-shot image recognition

Koch, Zemel, Salakhutdinov 15

Training similarity

same

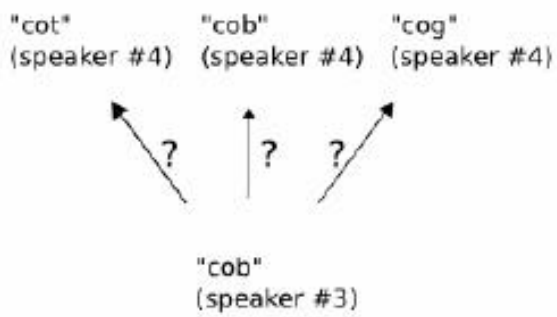
different

same

different

"cow" (speaker #1)	"cow" (speaker #2)	same
"cow" (speaker #1)	"cat" (speaker #2)	different
"can" (speaker #1)	"can" (speaker #2)	same
"can" (speaker #1)	"cab" (speaker #2)	different

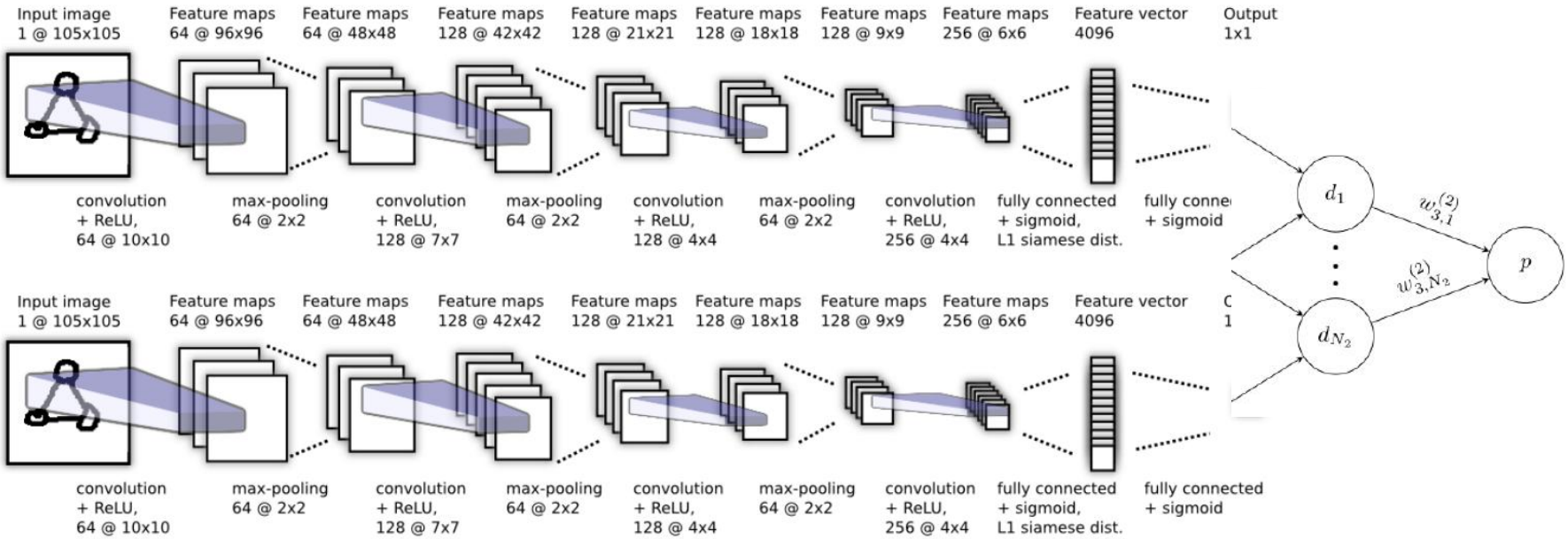
One-shot setting



Ingredients

Koch et al. 15

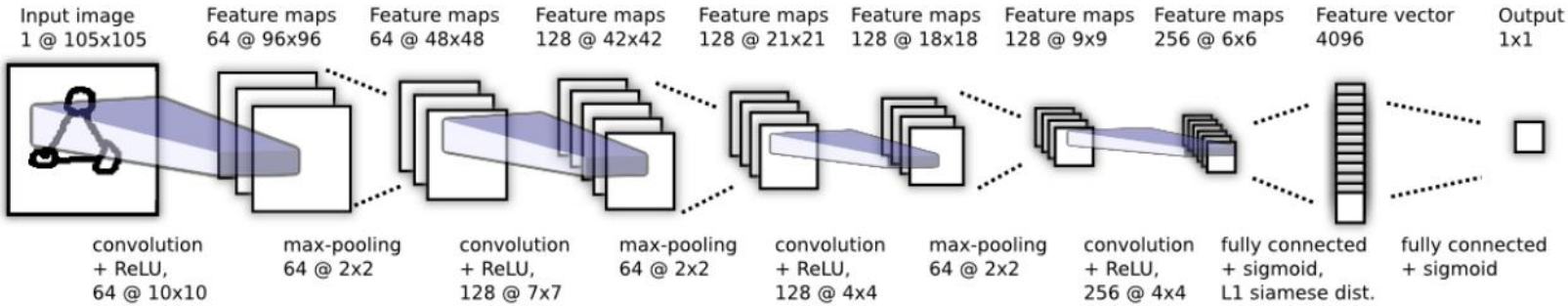
Architecture



Ingredients, 2

Koch et al. 15

Architecture



Distance

$$d(x, x') = \sigma\left(\sum_k \alpha_k |z_k(x) - z_k(x')|\right)$$

Loss

Given a batch $((x_i, x_i'), y_i)$ with $y_i = 1$ iff x_i and x_i' are similar

$$\mathcal{L}(w) = \sum_i y_i \log d(x_i, x_i') + (1 - y_i) \log\left(1 - d(x_i, x_i')\right) + \lambda \|w\|^2$$

Results

Koch et al. 15

Omniglot

Results

Method	Test
Humans	95.5
Hierarchical Bayesian Program Learning	95.2
Affine model	81.8
Hierarchical Deep	65.2
Deep Boltzmann Machine	62.0
Simple Stroke	35.2
1-Nearest Neighbor	21.7
Siamese Neural Net	58.3
Convolutional Siamese Net	92.0

Siamese Networks

PROS

- Learn metrics, invariance operators
- Generalization beyond train data

CONS

- More computationally intensive
- More hyperparameters and fine-tuning, more training