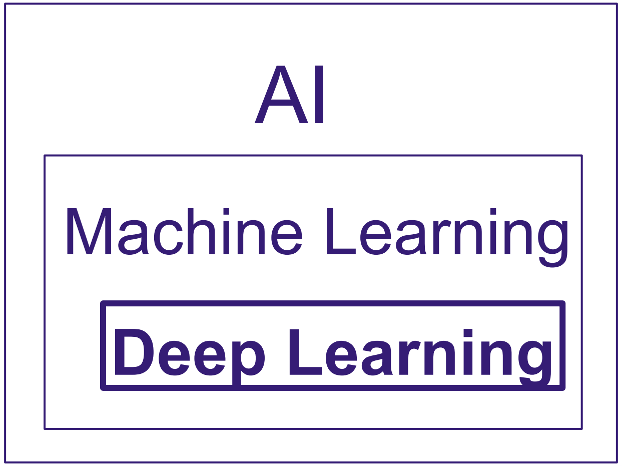
Deep Learning : Theory and Applications

Valentin LEVEAU - Post-doctoral fellow at INRIA/Zenith team

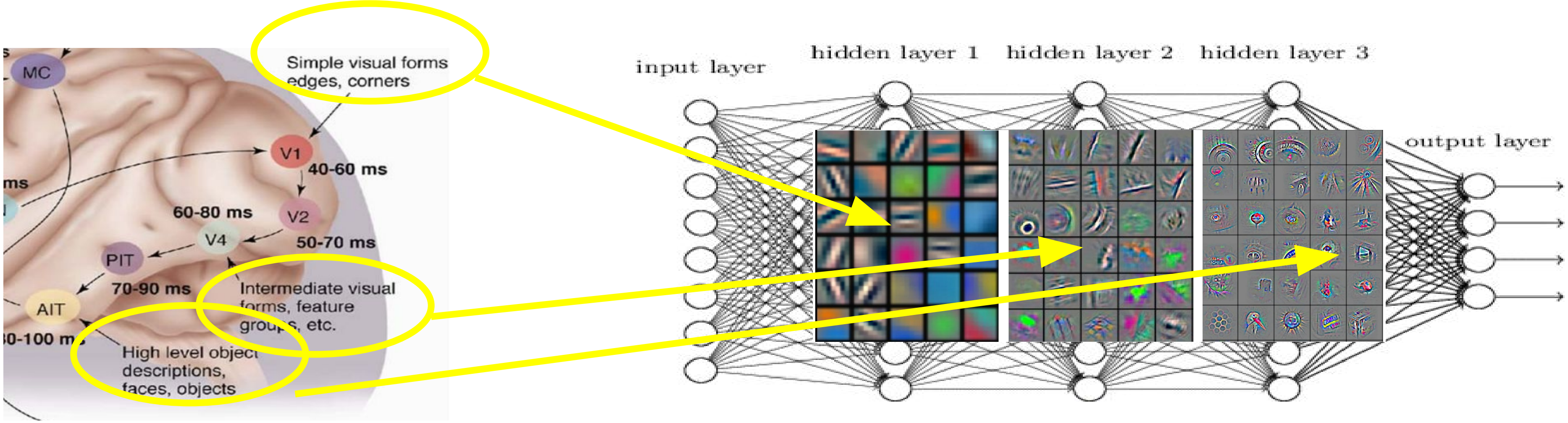
Deep Learning is a particular form of Machine Learning

- We now are good at mimicking some part of intelligence : **Learning**
- **(Machine) Learning** = Learning from examples to do a given task and generalize to new examples.
- Goal = **predict some variables given others.**
- Capture statistical relationships / structure between observed variables.



Deep Learning gets inspiration from biology

- Learning several levels of abstraction of the input signal (compositionality)
- For Image : find the progressive transition from pixels to labels



Some Words about bio-inspiration (Yann LeCun)

- Do we need to copy biology to get truly intelligent systems ?
- Brain is just a possible instance of intelligent device.
- Evolution took a long time to design our cognitive functions.
- We should rather understand the underlying principles of intelligence to build another instance of cognitive system. (e.g. aerodynamics for flying systems).

L'Avion III de Clément Ader, 1897
(Musée du CNAM, Paris)

Outline

1. **What are we fighting against ?**

   Invariances + Curse of dimensionality

   Priors to learn good data representation (toward deep representation learning)

2. **Learning procedures for deep architectures**

   From Artificial Neural Networks → Deep Convolutional Neural Networks (ConvNet)

   Recent advances : why ConvNet got famous so late ?

   Applications

3. **Unsupervised Learning**

Outline

1. **What are we fighting against ?**
   - Invariances + Curse of dimensionality
   - Priors to learn good data representation (toward deep representation learning)
2. **Learning procedures for deep architectures**
   - From Artificial Neural Networks → Deep Convolutional Neural Networks (ConvNet)
   - Recent advances : why ConvNet got famous so late ?
   - Applications
3. **Unsupervised Learning**

Classification in high dimensional spaces

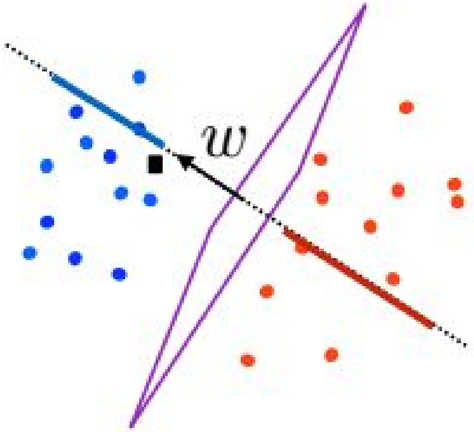
- Find a function that maps high dim input variables to output variables

$$\{x_i\ ,\ y_i = f(x_i)\}_{i \leq n}$$

- Simple solution: **Linear Model**

$$f(\mathbf{x_i}) = sign(\mathbf{W}^T\mathbf{x_i})$$

  - Equivalent to finding an hyperplane that separates the data

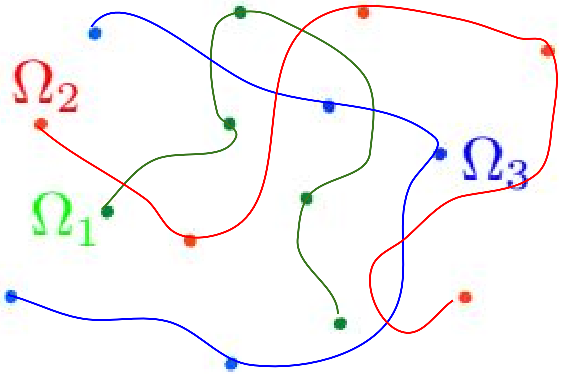


Problem n°1 : Highly Nonlinear Structure (S. Mallat)

Classes

Level sets of $f(x)$

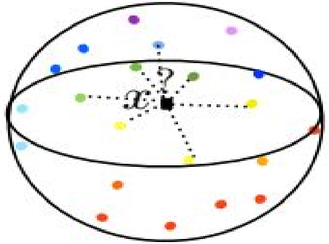
$$\Omega_t = \{x \ : \ f(x) = t\}$$

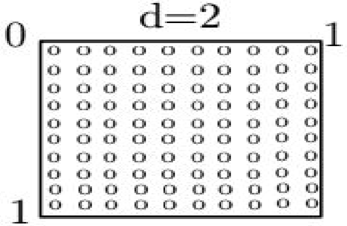


Reduce dimensionality of the problem (S.Mallat)

- $f(x)$ can be approximated from examples $\{x_i\ ,\ f(x_i)\}_i$ by

local interpolation if f is regular and there are close examples:





- To cover $[0,1]^d$ at a distance 10^{-1} we need 10^d points

Problem: $\|x - x_i\|$ is always large

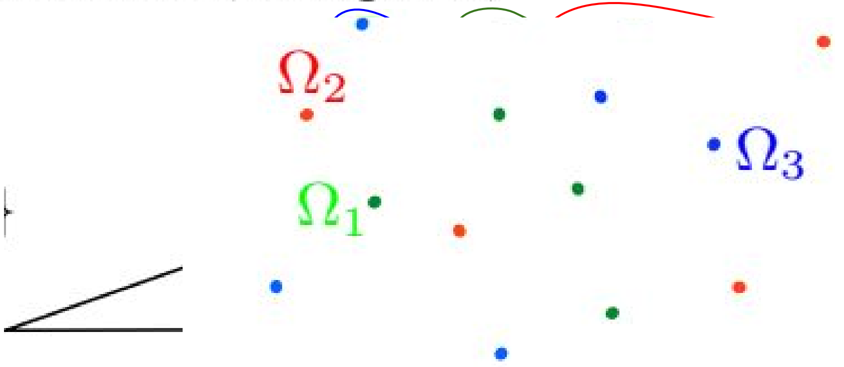
Huge variability inside classes

Find invariants

Reduce dimensionality of the problem (S. Mallat)

- If level sets Ω_t are not parallel to a linear space
  - Linearise them with a change of variable $\Phi(x)$
  - Then reduce dimension with linear projections

- Difficult because Ω_t are high-dimensional, irregular, known on few samples.

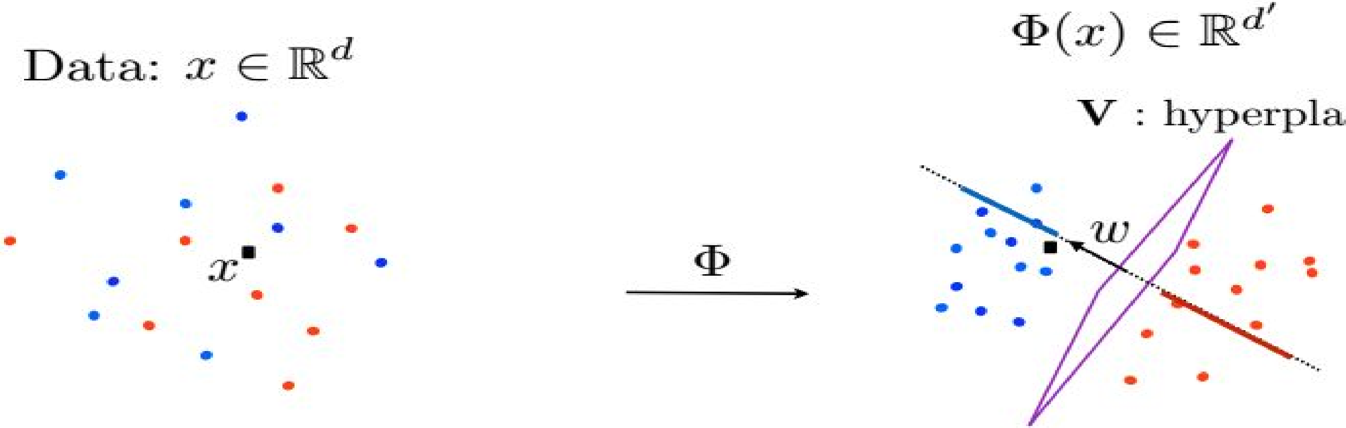


Find Non Linear Invariant in the data (S. Mallat)

1. Find a change of variable $\Phi(x)$

linearization
separation

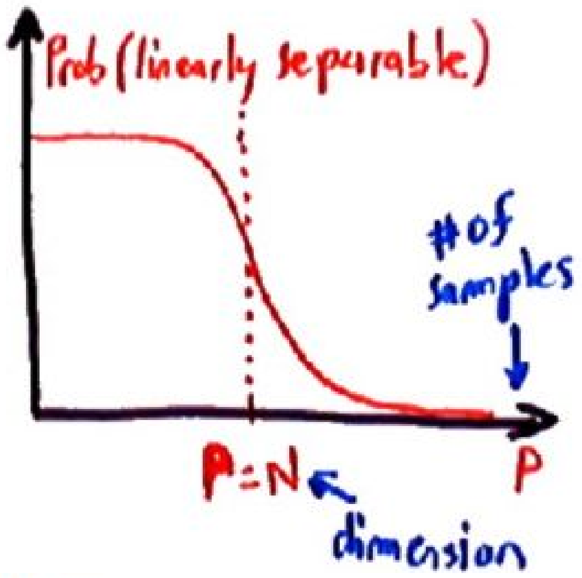
2. Find a linear projection: $\langle \Phi(x), w \rangle = \sum_k w_k \, \phi_k(x)$



- How and when is possible to find such a Φ ?

Several strategies to go non linear (Cover's Theorem)

The probability that P samples of dimension N are linearly separable goes to zero very quickly as P grows larger than N (Cover's theorem, 1966).

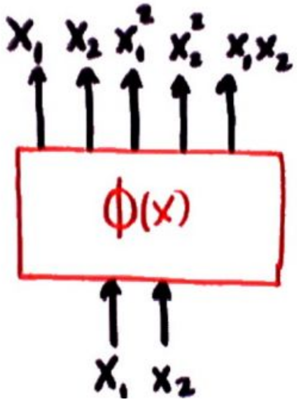


- Problem: there are 2^P possible dichotomies of P points.
- Only about N are linearly separable.
- If P is larger than N, the probability that a random dichotomy is linearly separable is very, very small.

Yann LeCun

New York University

Several strategies to go non linear

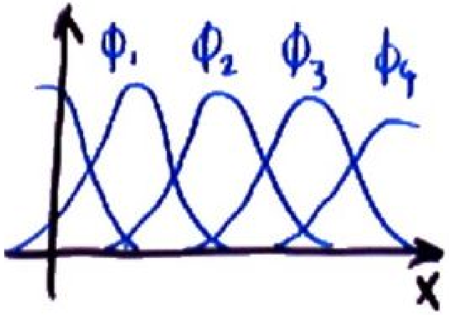


- **Feature Augmentation: Polynomial mapping**

  → Adding all cross products of the original variables

- Problem: The order of the polynom might be high

  → Gives rise to impractical feature's dimension size



- **Tiling the space + Kernel Methods**:
  - Decision function is a linear combination of different position in the feature space
  - Kernel: Just put bumps where data live

$$y = \sum_{i=1}^{P} \alpha_i K(X, X^i)$$

cognition – p. 24/36

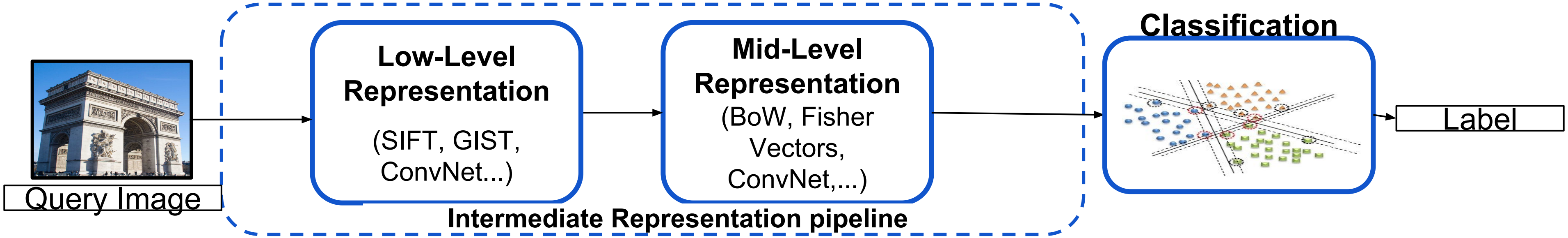
Yann LeCun

New York University

Several strategies to go non linear

- **Produce handcrafted intermediate representation of images**

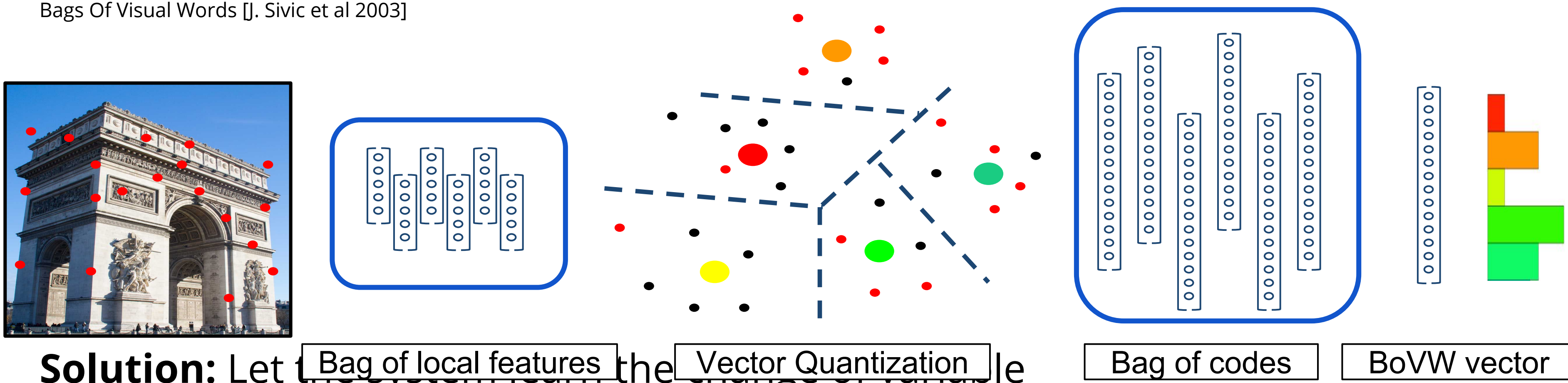
Problem: Decide manually which kind of features are good for the different tasks



Several strategies to go non linear

Produce handcrafted intermediate representation of images

Bags Of Visual Words [J. Sivic et al 2003]



Solution: Let the system learn the change of variable

→ Toward **Representation Learning**

Outline

1. **What are we fighting against ?**

   Invariances + Curse of dimensionality

   Priors to learn good data representation (toward deep representation learning)

2. **Learning procedures for deep architectures**

   From Artificial Neural Networks → Deep Convolutional Neural Networks (ConvNet)

   Recent advances : why ConvNet got famous so late ?

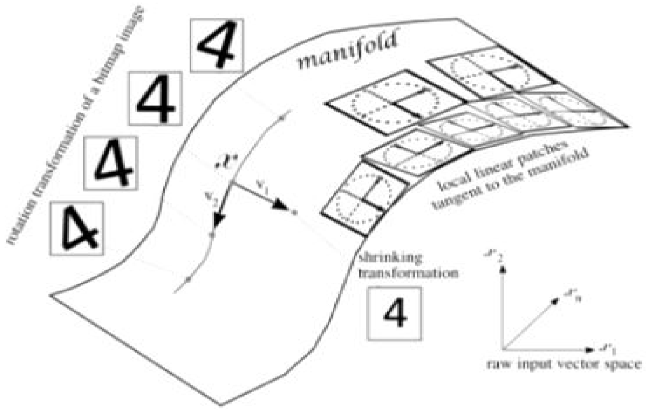
   Applications

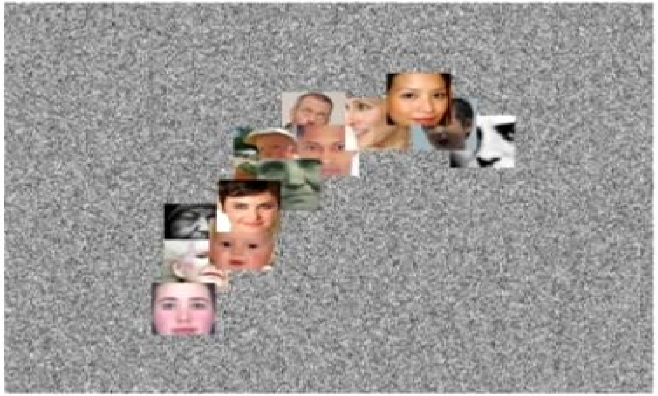
3. **Unsupervised Learning**

The manifold hypothesis of the data (Y. Bengio)

- examples **concentrate** near a lower dimensional “manifold
- **Evidence: most input configurations are unlikely**

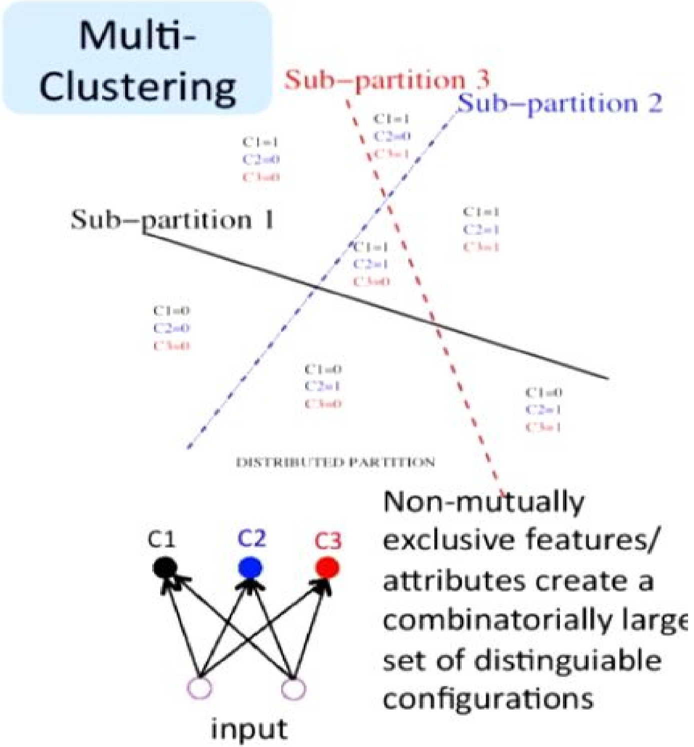
→ We need to find such authorized directions of variations in the input space
→ Put probability mass where data live





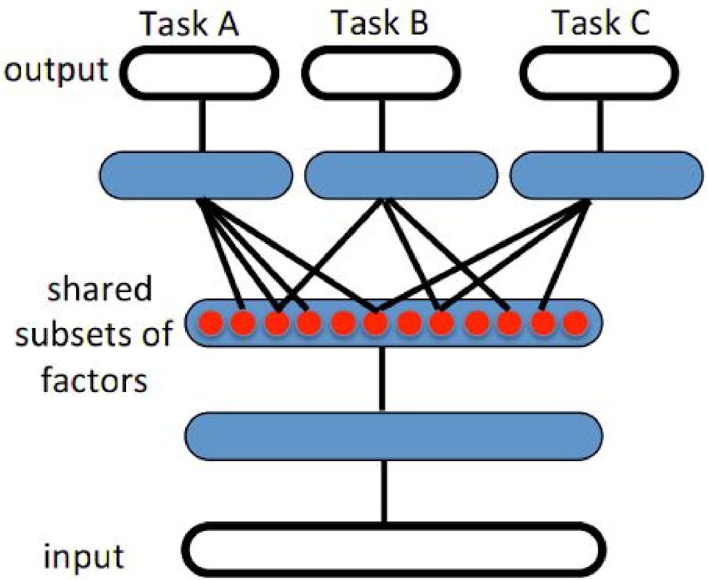
Distributed representation (Y. Bengio)

- Factor models, PCA, RBMs, Neural Nets, Sparse Coding, Deep Learning, etc.
- Each parameter influences many regions, not just local neighbors
- **# of distinguishable regions grows almost exponentially with # of parameters**
- **GENERALIZE NON-LOCALLY TO NEVER-SEEN REGIONS**



Multi Task Learning (Y. Bengio)

- **Better to share factors across tasks, modalities, etc ….**
- **Better generalization because Explanatory factor are likely to be meaningfull**



Do we rather need deep or large architecture ?

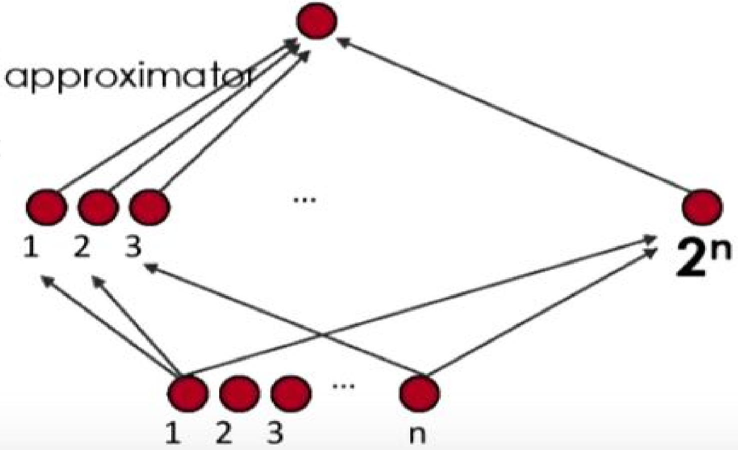
$$y = \sum_{i=1}^{P} \alpha_i K(X, X^i) \qquad y = F(W^1.F(W^0.X))$$

2 layers of $\left\{ \begin{array}{l} \text{Logic gates} \\ \text{Formal neurons} \\ \text{RBF units} \end{array} \right.$ = universal approximator

RBMs & auto-encoders = universal approximator

Theorems on advantage of depth:
(Hastad et al 86 & 91, Bengio et al 2007, Bengio & Delalleau 2011, Braverman 2011)

Some functions compactly represented with k layers may require exponential size with 2 layers

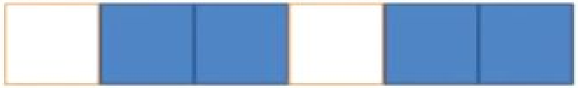


Sparse representation (Y. Bengio)

- Just add a sparsifying penalty on learned representation (prefer 0s in the representation)
- Information disentangling (compare to dense compression)
- More likely to be linearly separable (high-dimensional space)
- Locally low-dimensional representation = local chart
- Hi-dim. sparse = efficient variable size representation
  = data structure

Few bits of information

Many bits of information



Prior: only few concepts and attributes relevant per example

Outline

1. **What are we fighting against ?**

   Invariances + Curse of dimensionality

   Priors to learn good data representation (toward deep representation learning)

2. **Learning procedures for deep architectures**

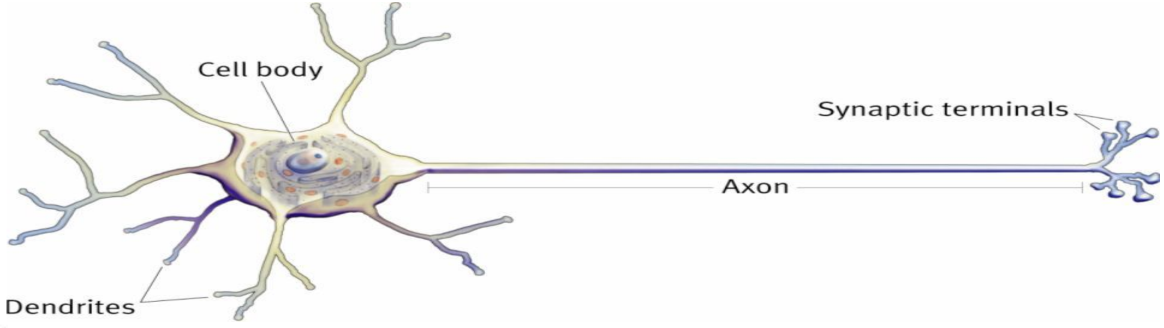
   From Artificial Neural Networks → Deep Convolutional Neural Networks (ConvNet)

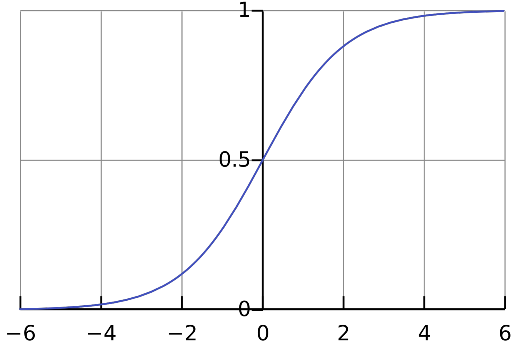
   Recent advances : why ConvNet got famous so late ?

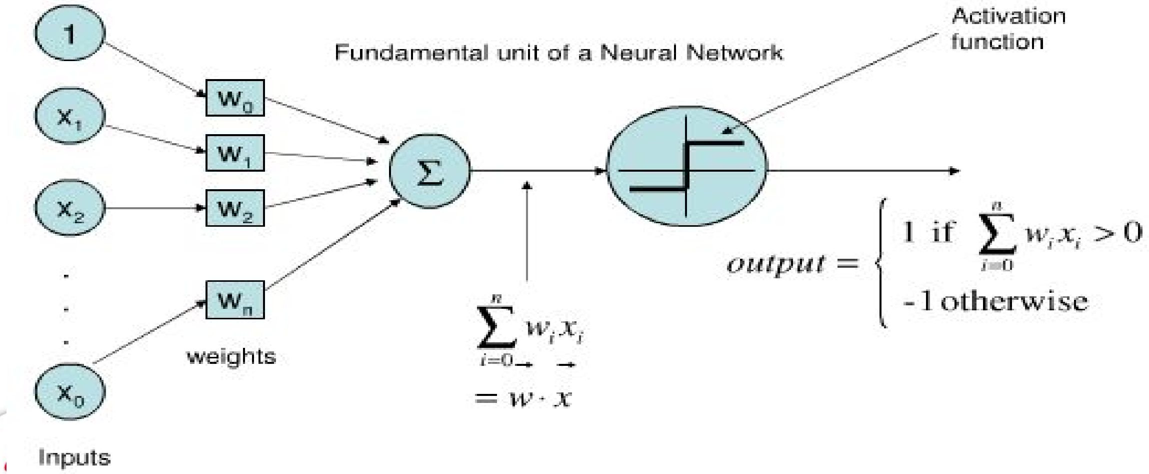
   Applications

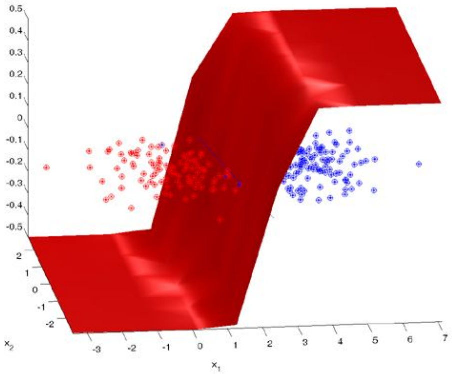
3. **Unsupervised Learning**

Perceptron : Simple Elementary Neural Unit

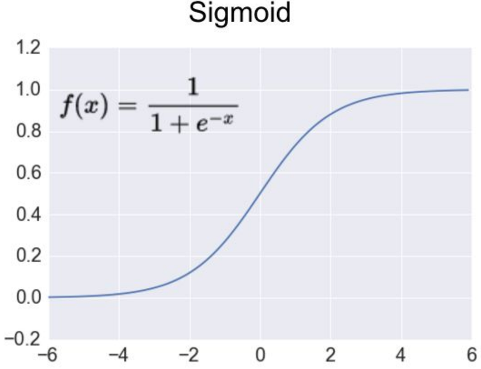


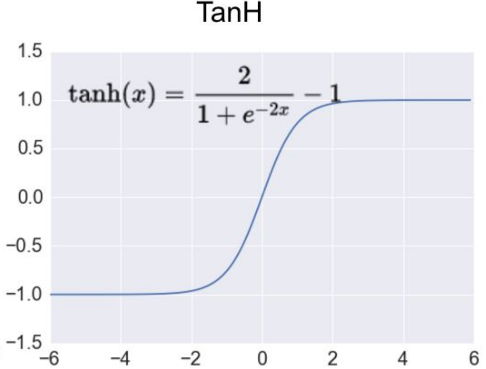




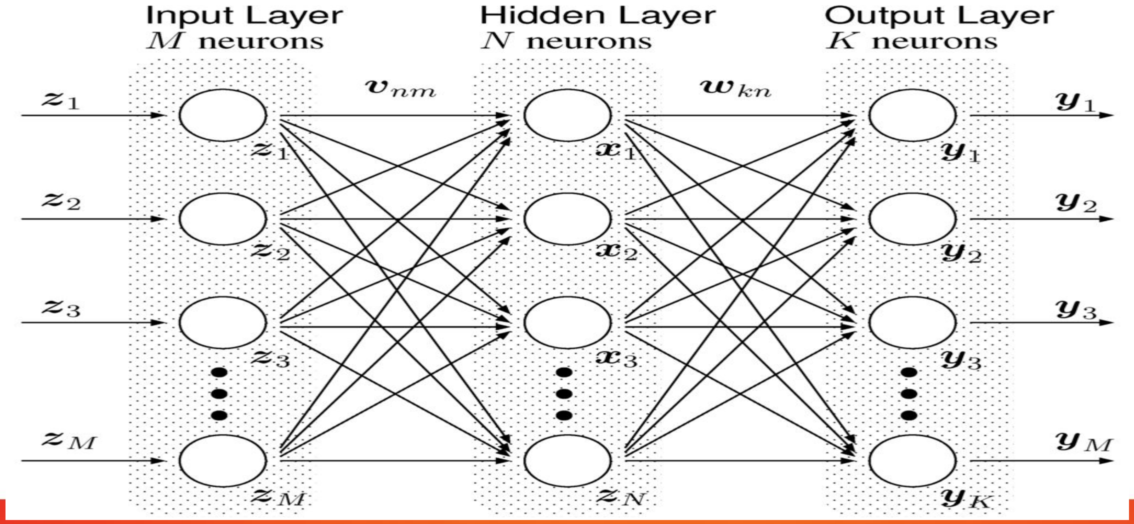


Multi Layer Perceptron





- 2 layers architecture :
  - Layer 1: several units in parallel + **non-linearities**
  - Layer 2 : final linear classifier unit

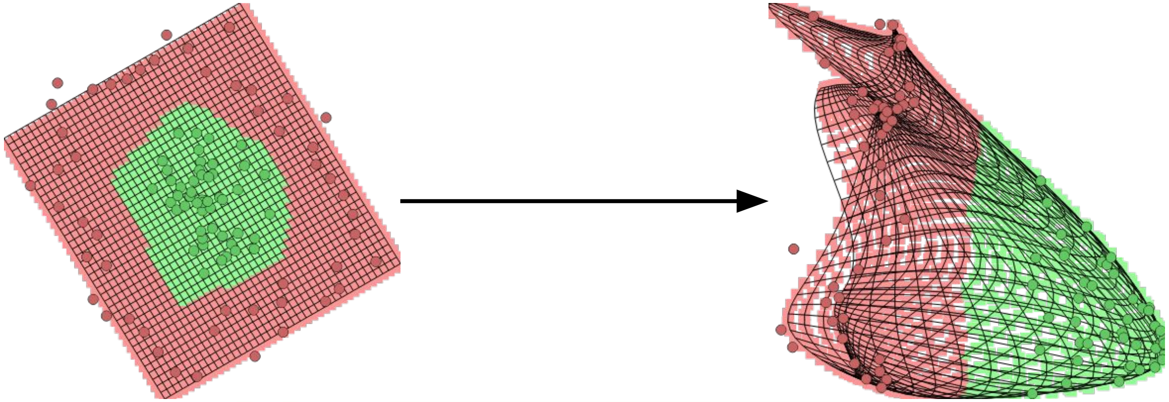


Multi Layer Perceptron

- The non linearities are crucial !!!
  - Linear combinations of linear combinations = Linear combinations (useless)

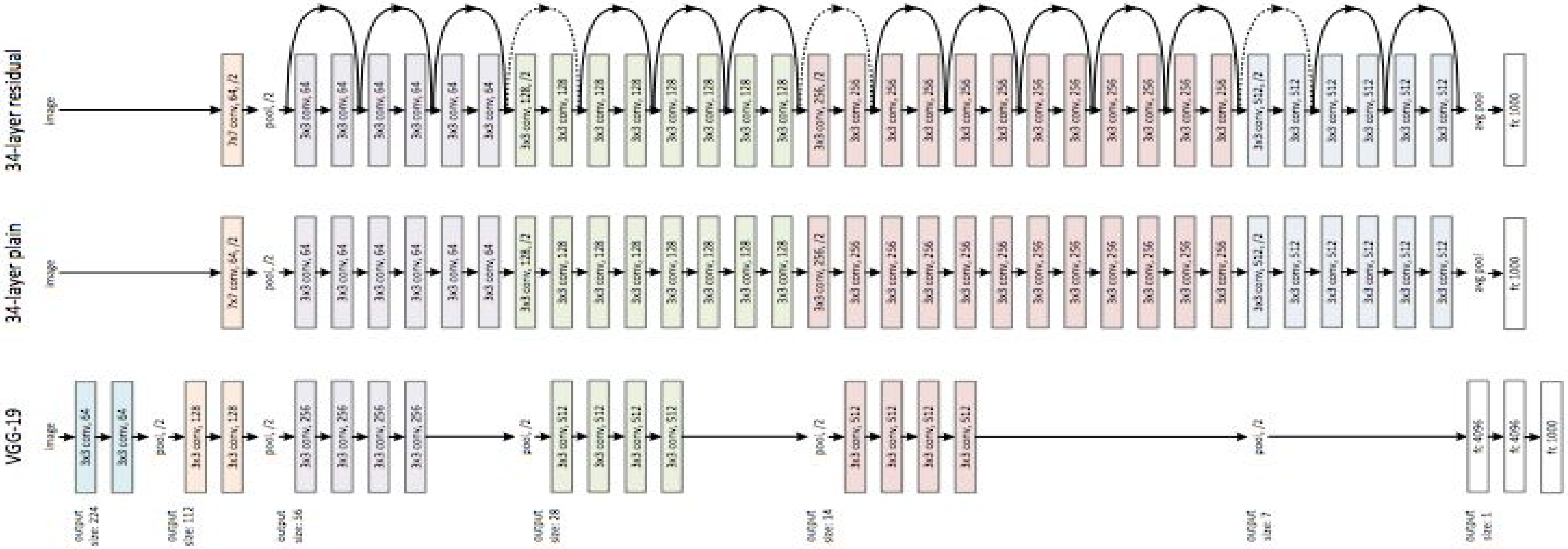
$$\mathbf{W_N}(\mathbf{W_{N-1}}(....(\mathbf{W_1}(\mathbf{W_0 x_i}))...)) = \tilde{\mathbf{W}}\mathbf{x_i}$$

  - Allow to bend the space to get samples linearly separable (click the curved space)



Deep Learning at scale

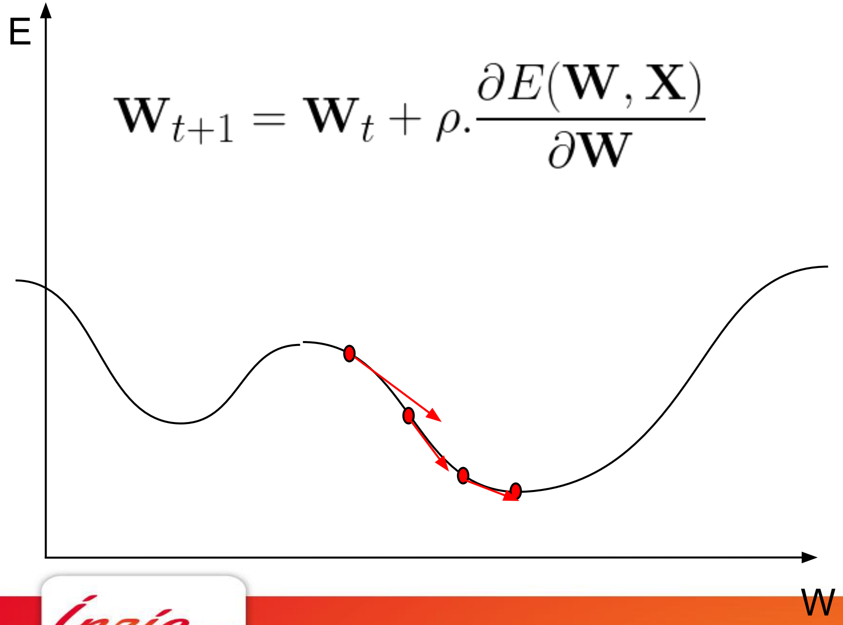
- A lot of parameters

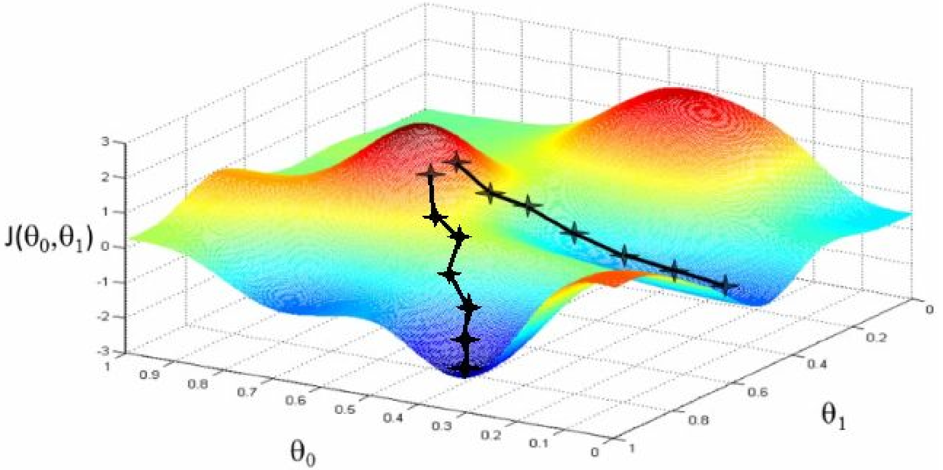


Energy Landscape and Gradient Descent

- Useful for non-convex function !!!
- Problem: lots of local minima ….

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \rho . \frac{\partial E(\mathbf{W}, \mathbf{X})}{\partial \mathbf{W}}$$





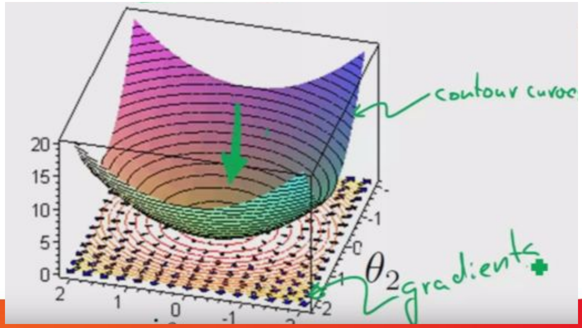
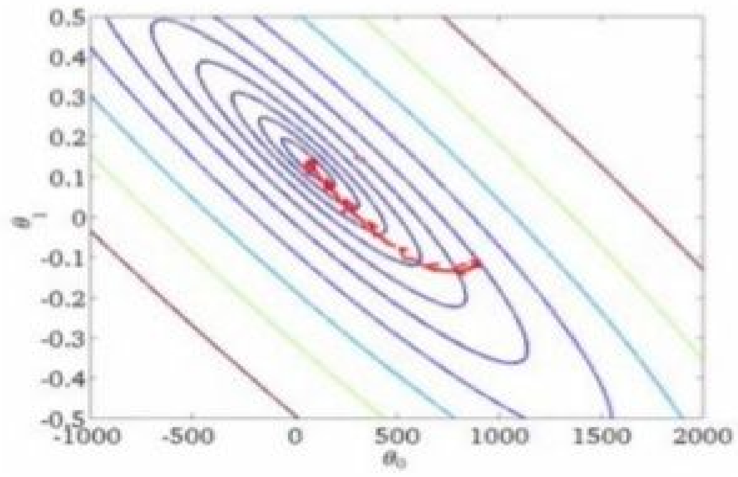
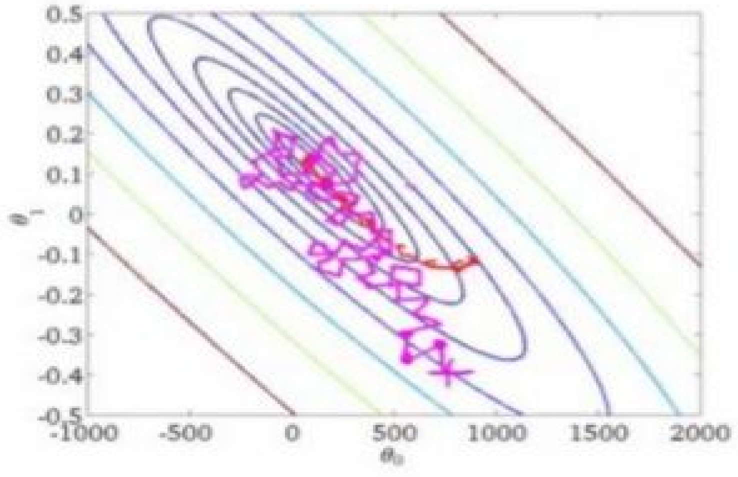


Illustration: batch gradient descent vs stochastic gradient descent



Batch: gradient

$$x \leftarrow x - \eta \nabla F(x)$$

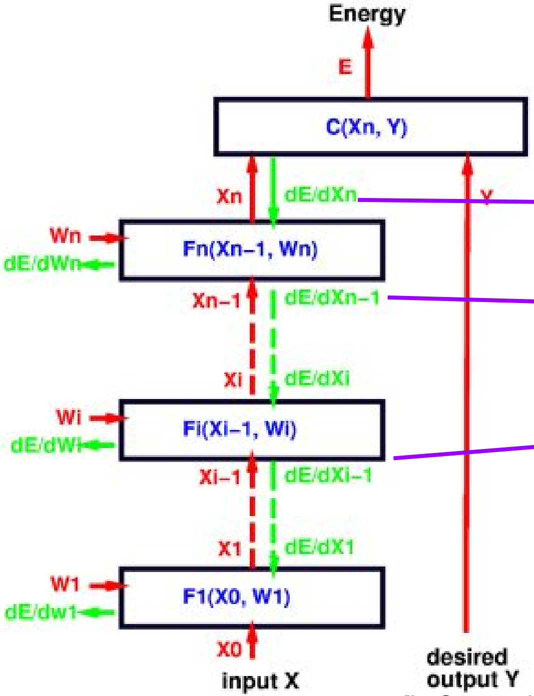


Stochastic: single-example gradient

$$x \leftarrow x - \eta \nabla F_i(x)$$

Backpropagation of the gradient

To compute all the derivatives, we use a backward sweep called the **back-propagation algorithm** that uses the recurrence equation for $\frac{\partial E}{\partial X_i}$



[LeCun et al. 89]

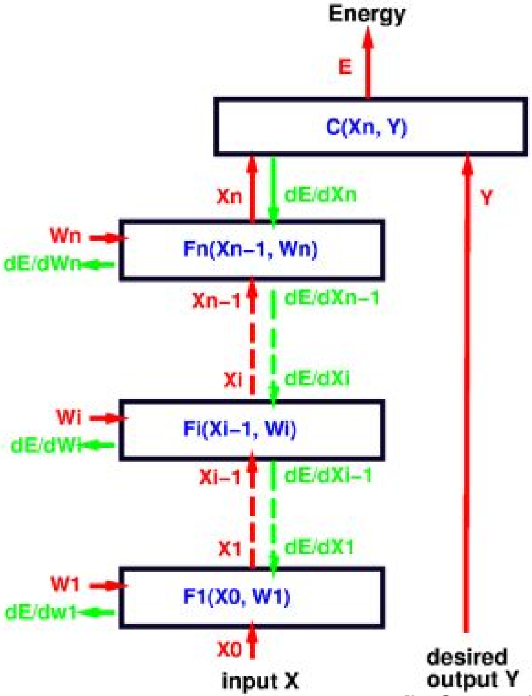
$$\frac{\partial E}{\partial X_n} = \frac{\partial C(X_n, Y)}{\partial X_n}$$

$$\frac{\partial E}{\partial X_{n-1}} = \frac{\partial E}{\partial X_n} \frac{\partial F_n(X_{n-1}, W_n)}{\partial X_{n-1}}$$

$$\frac{\partial E}{\partial W_n} = \frac{\partial E}{\partial X_n} \frac{\partial F_n(X_{n-1}, W_n)}{\partial W_n}$$

Backpropagation of the gradient

To compute all the derivatives, we use a backward sweep called the **back-propagation algorithm** that uses the recurrence equation for $\frac{\partial E}{\partial X_L}$



[LeCun et al. 89]

Forward Pass:

$$x_i = f_i(W_i x_{i-1})$$
$$E = \|x_L - t\|_2^2$$

Backward Pass:

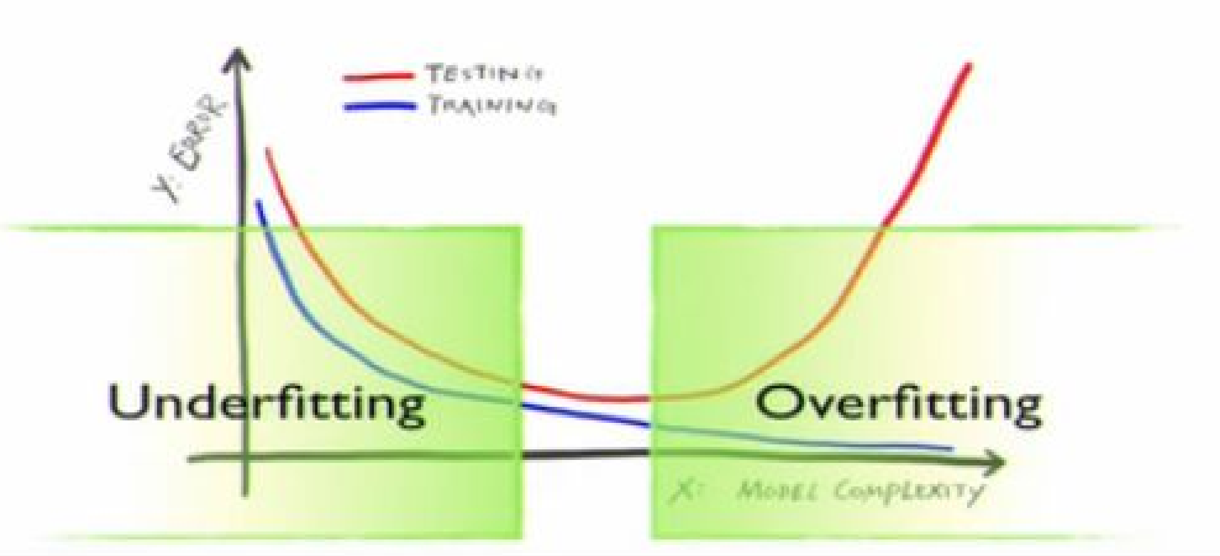
$$\delta_L = (x_L - t) \circ f_L'(W_L x_{L-1})$$
$$\delta_i = W_{i+1}^T \delta_{i+1} \circ f_i'(W_i x_{i-1})$$

Weight Update:

Weights move co-linearly in the direction of input data

$$\boxed{\frac{\partial E}{\partial W_i} = \delta_i x_{i-1}^T}$$

Generalization : find the “good” model capacity



[Understanding deep learning requires rethinking generalization. C. Zhang et al. ICLR17]

Regularization

- Objective: constraint the hypothesis space to be smaller

- Another formulation: Put some mess in the learning algorithm so that it does not converge to bad local minima

- Joint optimization of two function:

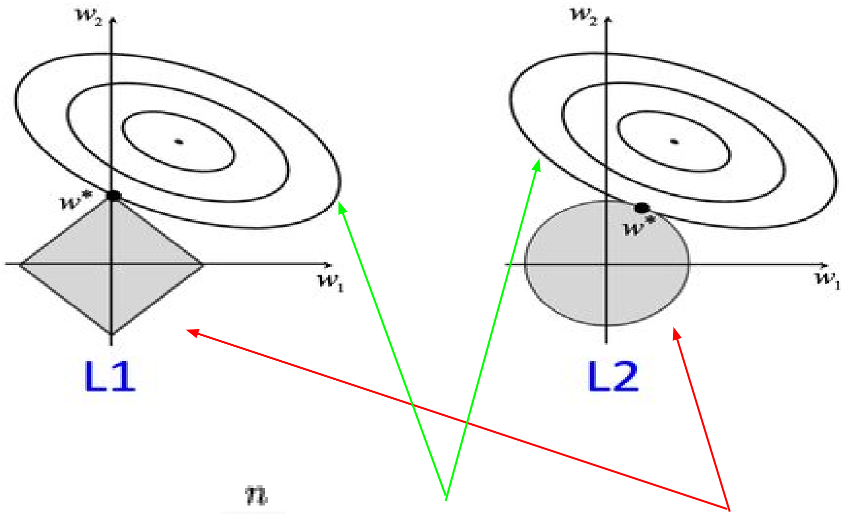
Regularization factor

$$\min_{f} \sum_{i=1}^{n} \underbrace{\boxed{V(f(\hat{x}_i), \hat{y}_i)}}_{\text{Learning objective term}} + \lambda \boxed{R(f)}$$

Regularization term

→ We can see this as two player playing against each other

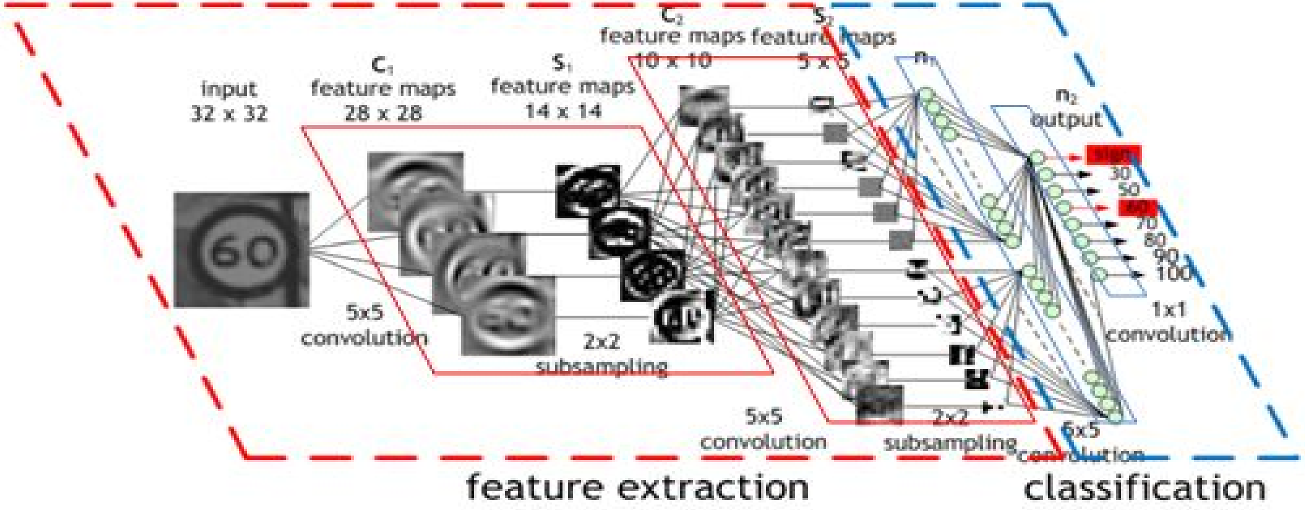
Regularization: common examples (L1/L2)



$$\min_f \sum_{i=1}^{n} V(f(\hat{x}_i), \hat{y}_i) + \lambda R(f)$$

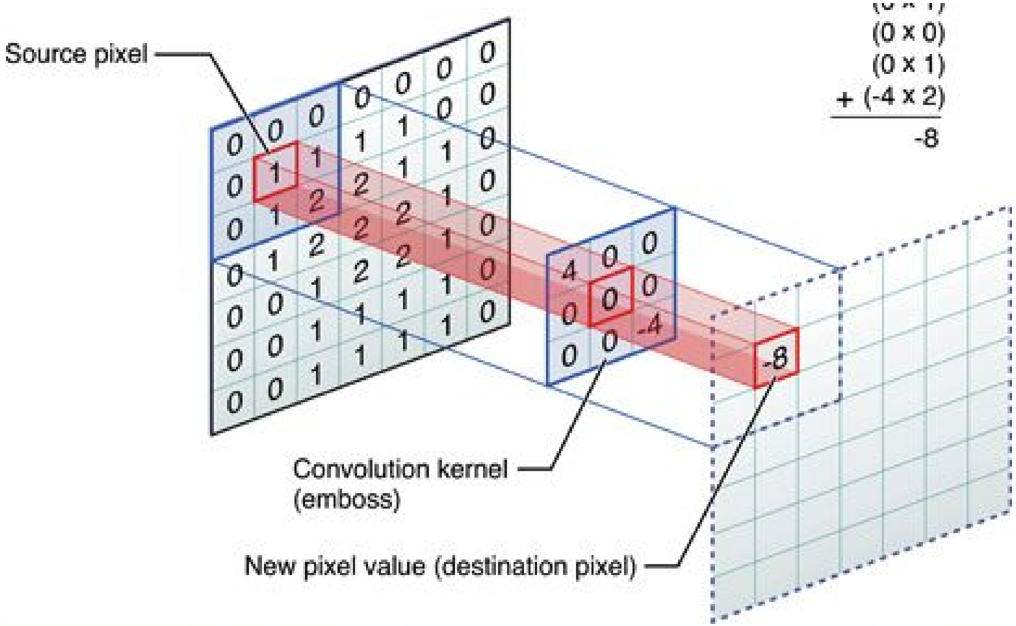
Convolutional Neural Network

Global architecture (LeNet5)



Translation Invariance with ConvNets

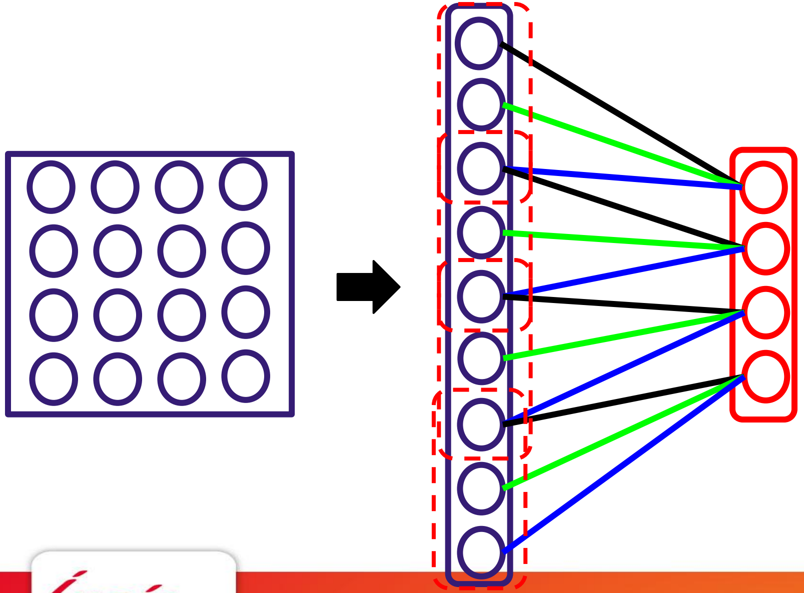
Replace dot product by a convolution

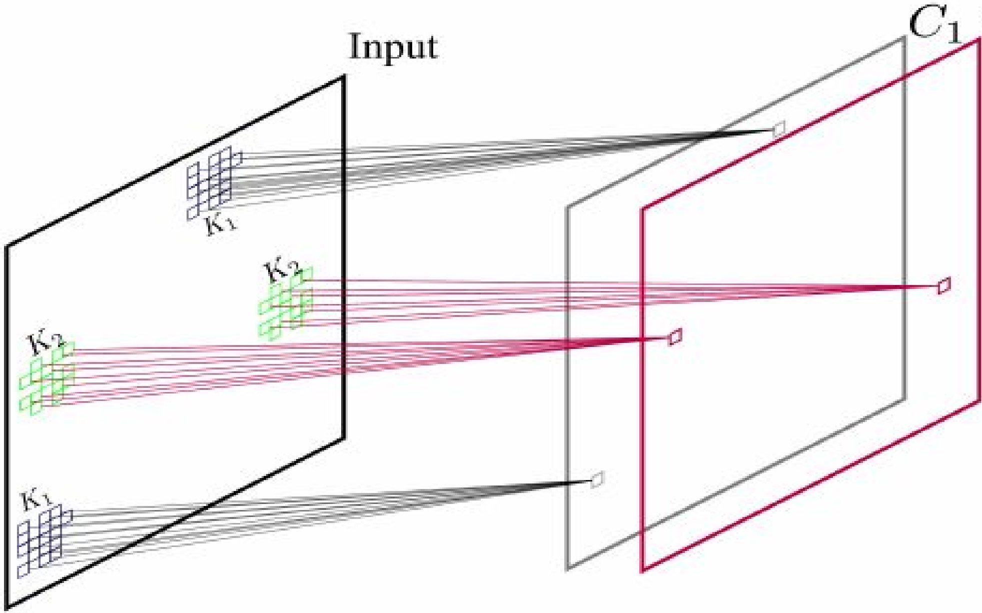


Translation Invariance with ConvNets

Integrate some spatial structure with local receptive fields ...

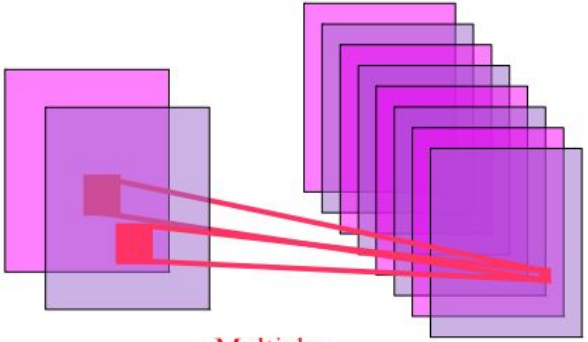
... is equivalent to sharing receptive fields parameters



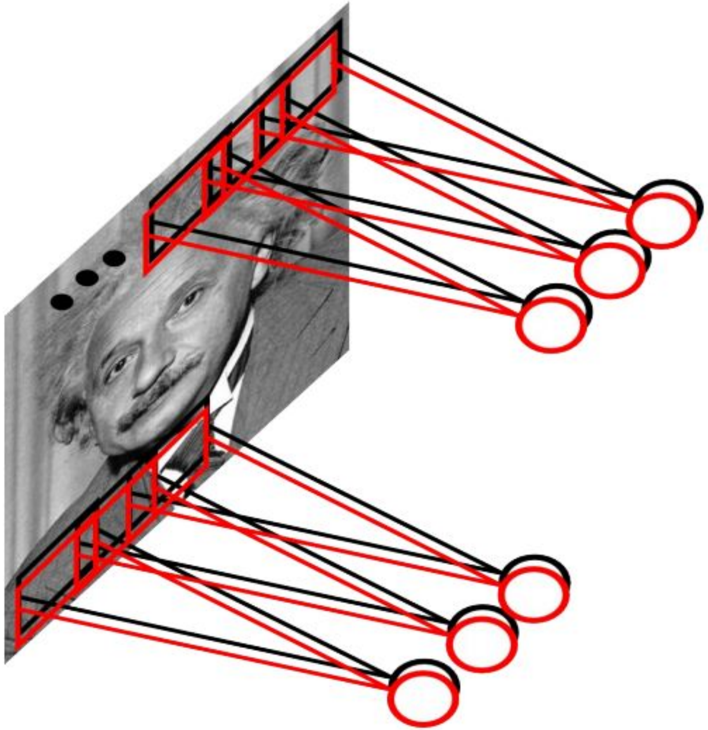


Translation Invariance with ConvNets

- Detects multiple motifs at each location
- The collection of units looking at the same patch is akin to a feature vector for that patch.
- The result is a 3D array, where each slice is a feature map.



Multiple convolutions

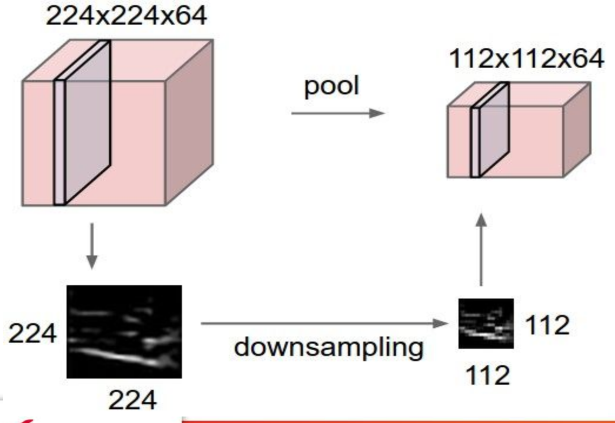


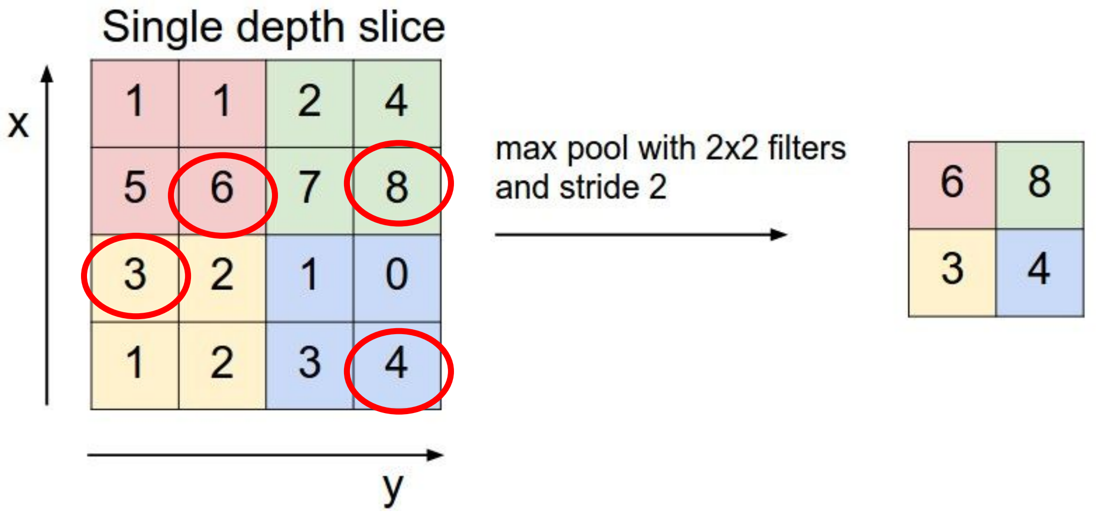
Backprop with ConvNet

- Weigth sharing for Conv Layers = sum the gradients

$$\Delta \mathbf{W_k} = \sum_{(x,y)} \Delta \mathbf{W_k^{(x,y)}}$$

- Max pooling :

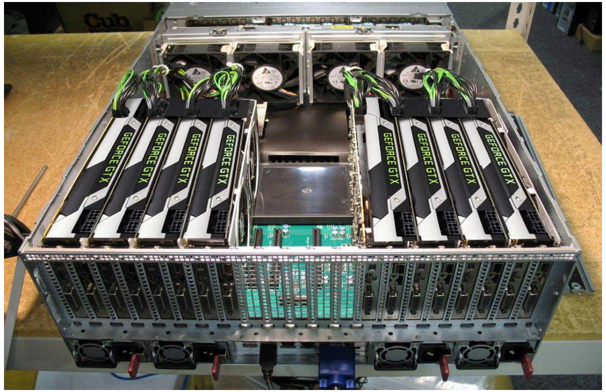




Why ConvNet got famous so late ?

- Not many theoretical justifications (since recently, but a lot remains to come !)
- Too much parameters for little datasets and too slow algorithms and hardwares
- Motivations in Unsupervised Learning (2000-2011)

Why ConvNet got famous so late ?



- Large scale labelled dataset : ImageNet (15 M)
- Hardware acceleration : GPU

What Changed ?

- The **ReLU** non linearity :
- The killer detail : as good (if not better) as unsupervised pretraining !

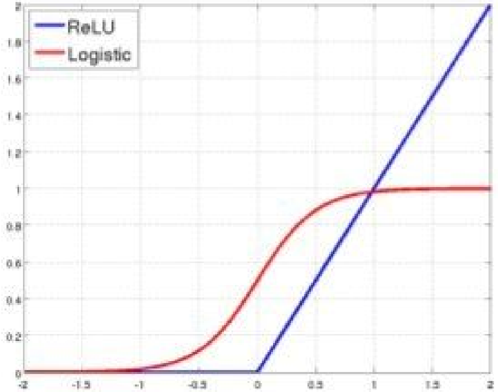
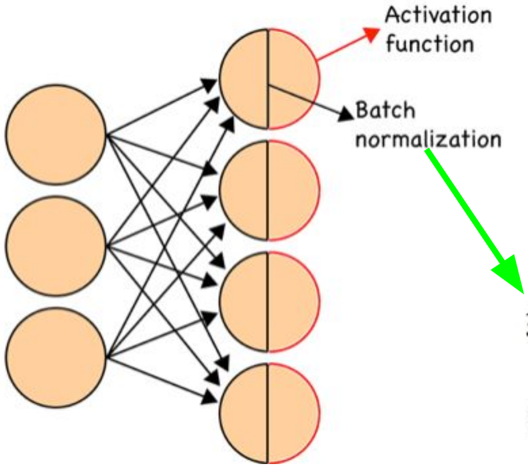


Fig. 1. The proposed non-linearity, ReLU, and the standard neural network non-linearity, logistic.

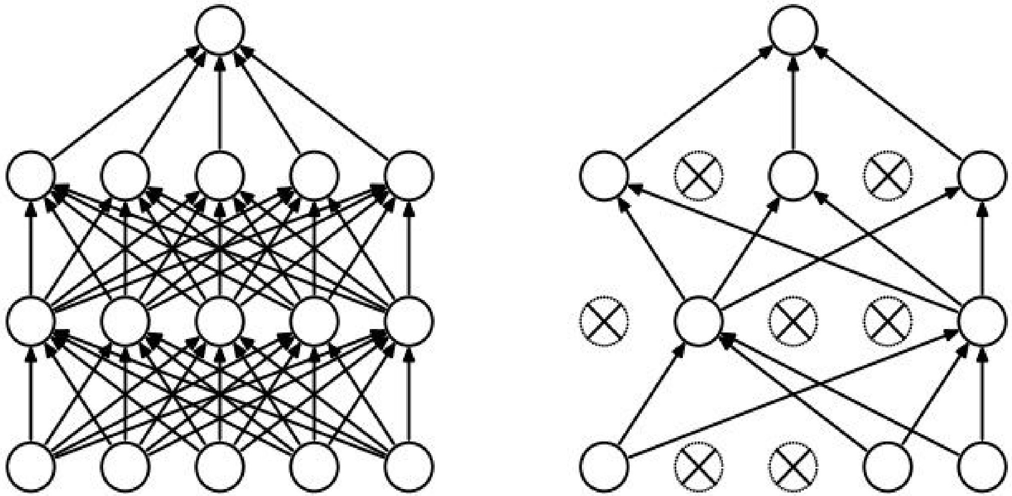
What Changed ?

- Dropout
- Batch Normalization(x14 faster !!!)



$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

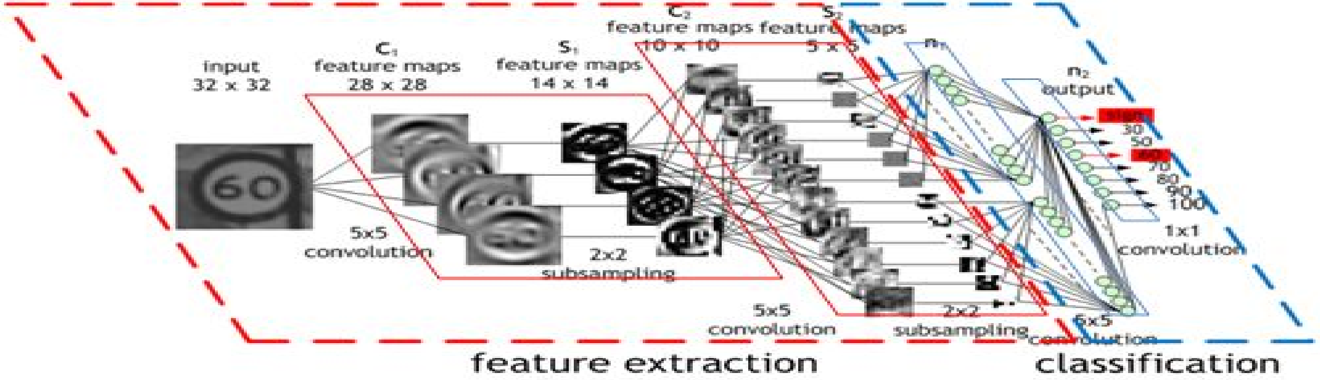
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$$



(a) Standard Neural Net (b) After applying dropout.

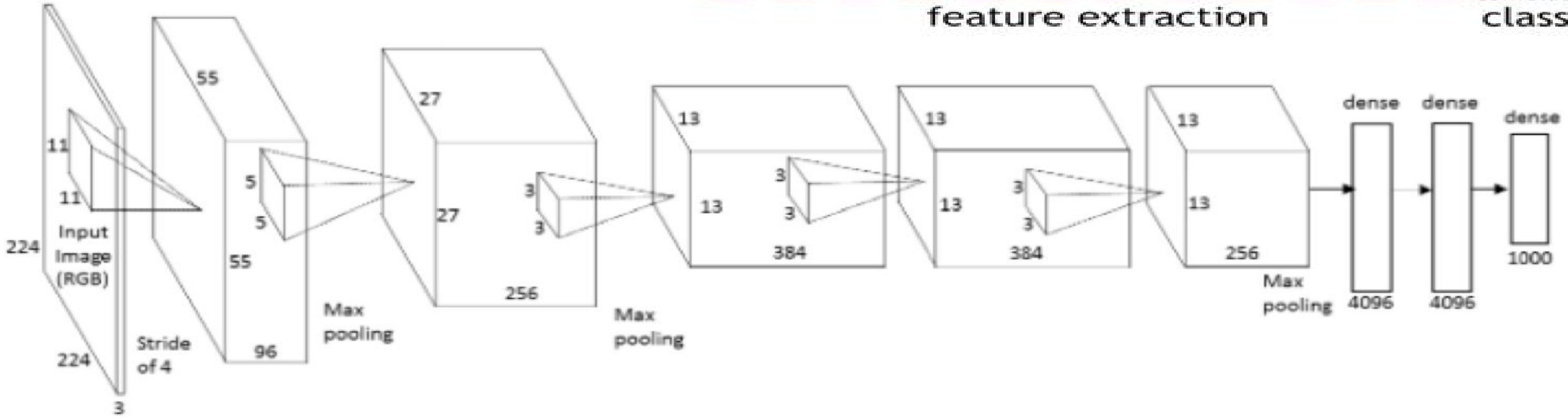
Evolution of the architectures

- LeNet1-5



- AlexNet

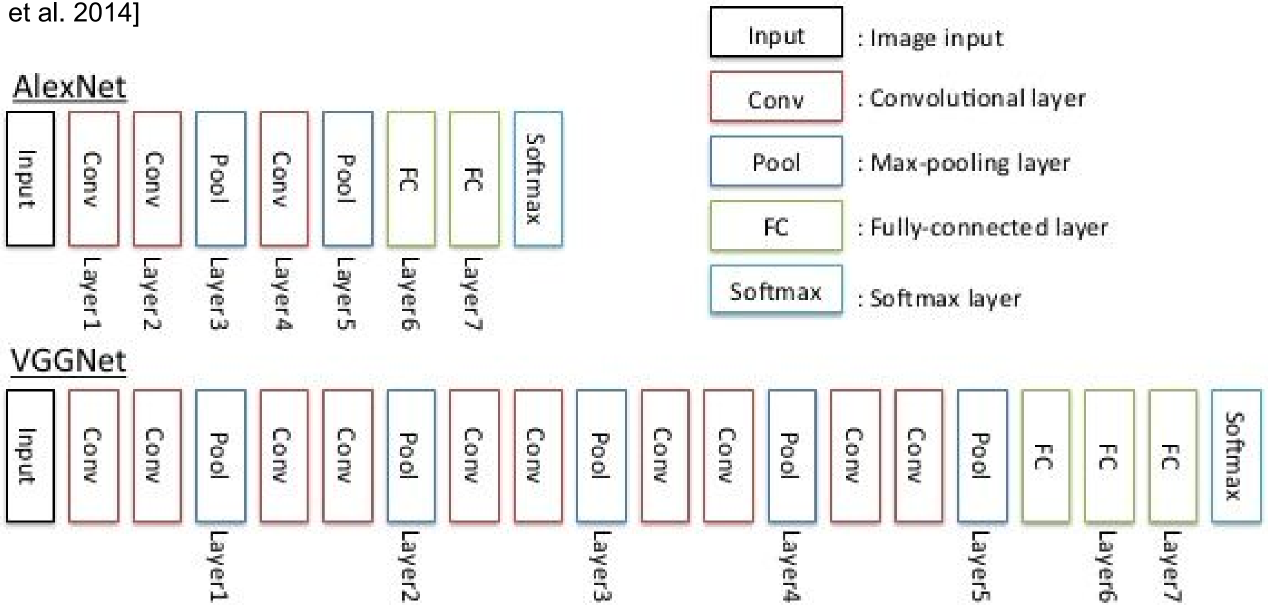
[A. Krizhevsky et al. 2012]



Evolution of the architectures

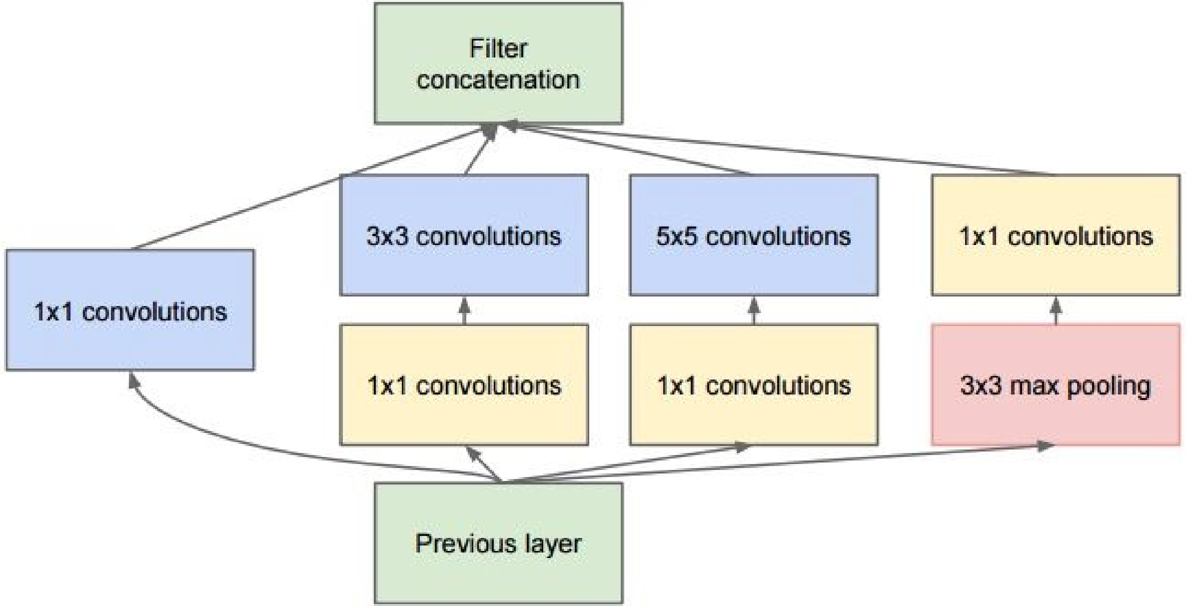
- VGG Net

[K. Simonyan et al. 2014]



Evolution of the architectures

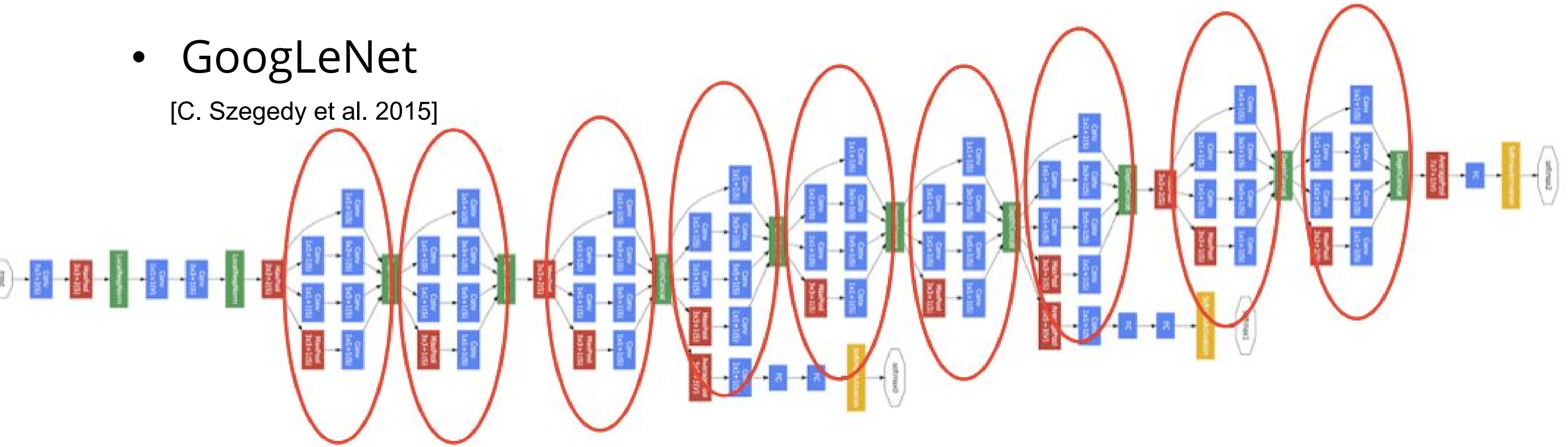
- Inception modules



Evolution of the architectures

- GoogLeNet

[C. Szegedy et al. 2015]

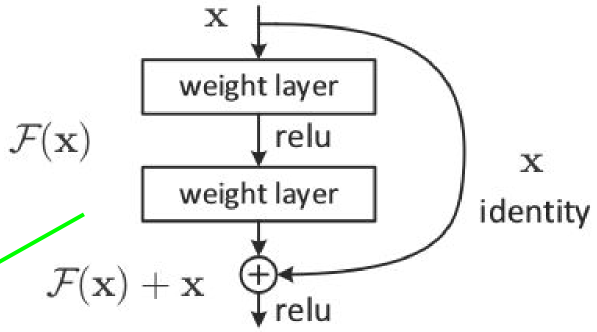


9 **Inception** modules

Network in a network in a network...

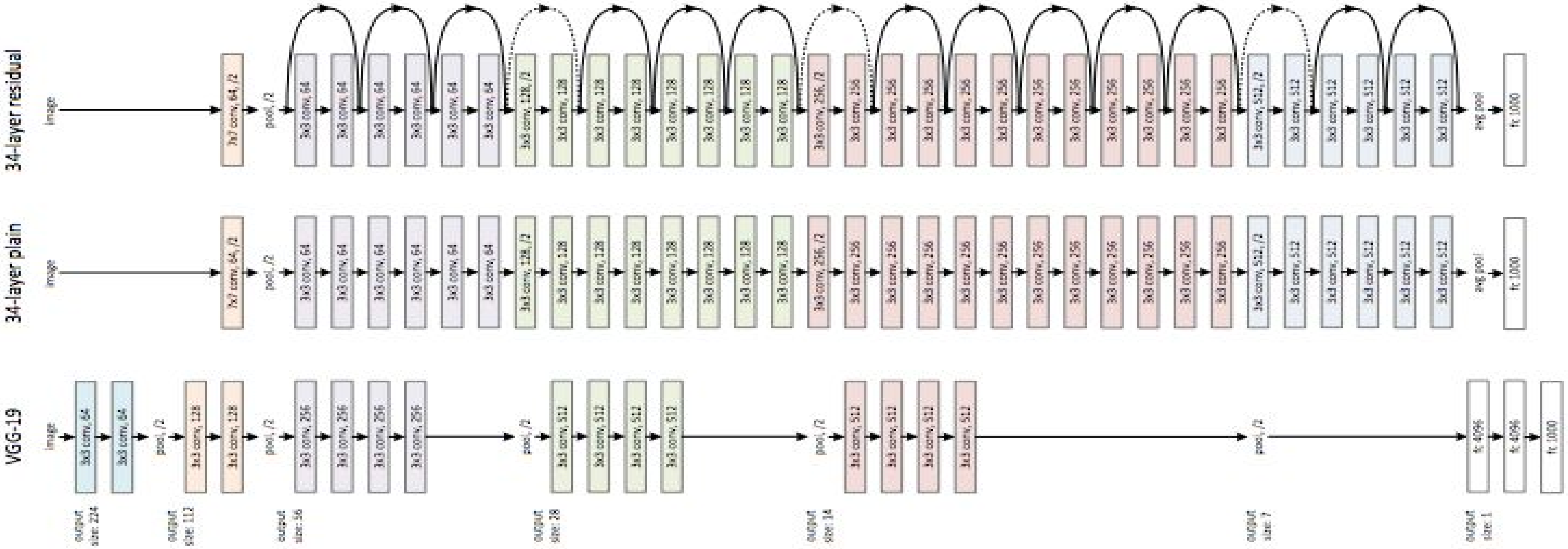
Convolution
Pooling
Softmax
Other

Evolution of the architectures



- ResNet (**152 layers**, even 1,000 !!!)

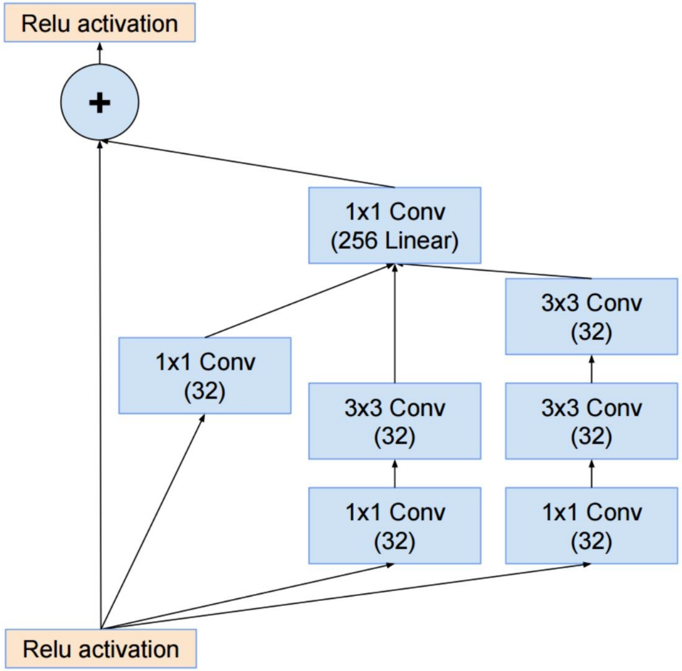
  [K. He et al. 2015]



Evolution of the architectures

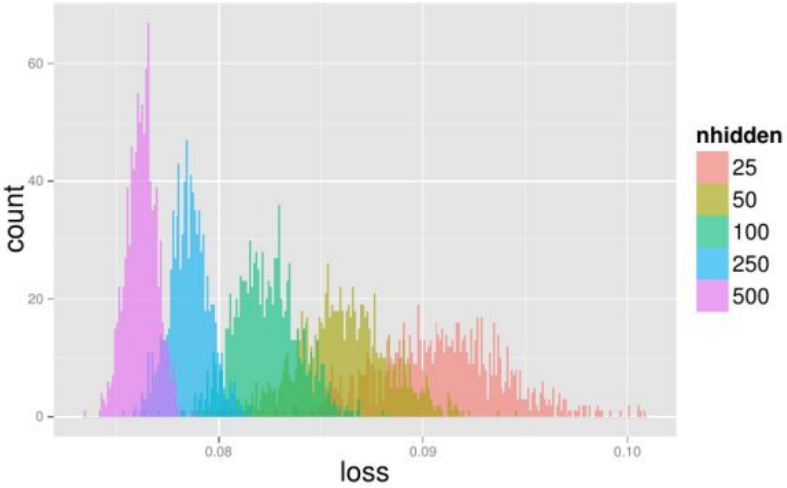
- ResNet modules combined with inception modules (**3.1%** Top-5 error on IN)

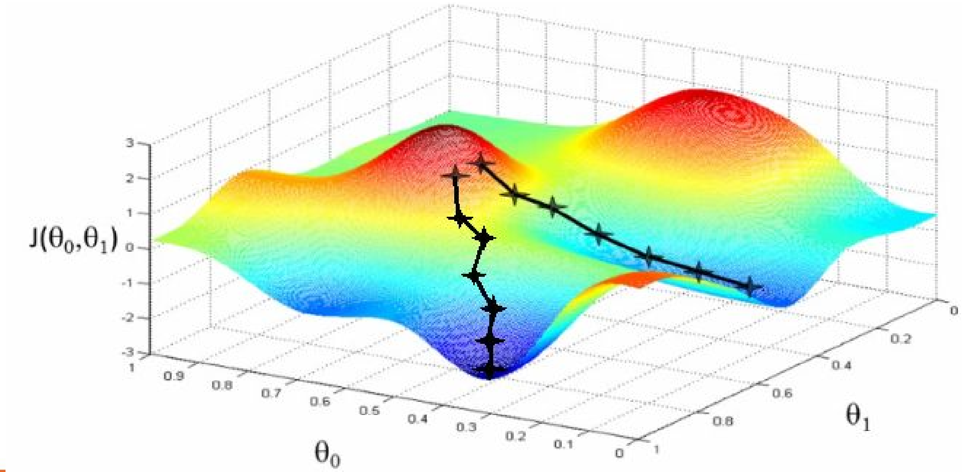
[C. Szegedy et al. 2016]



Energy function in High dimension

- Properties of ReLU Networks:
  - -All the local minima tends to have the same value of the energy function
  - -So we don't care where we start from and where we arrive
  - -Non convexity is a false problem

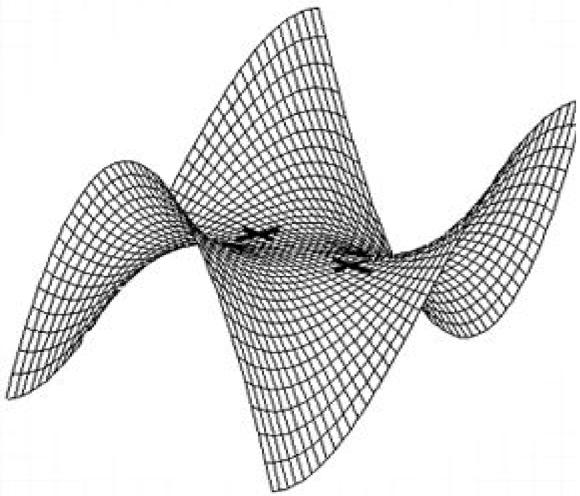


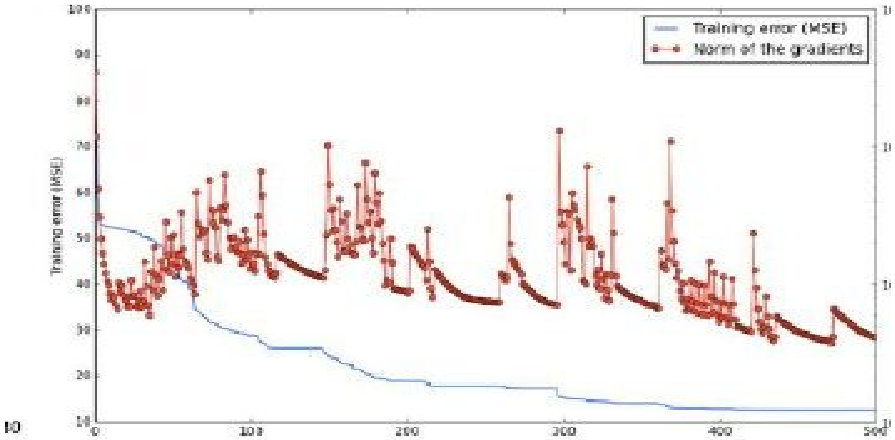


[Choromanska, Henaff, Mathieu, Ben Arous, LeCun 2015]

Energy function in High dimension

- Energy function are actually highly populated by saddle points
- As we get close to the global minimum value of E(), it becomes harder and harder to find directions that goes up rather than down.
- The proportion of going up directions grows exponentially





[R. Pascanu et al. 2014]

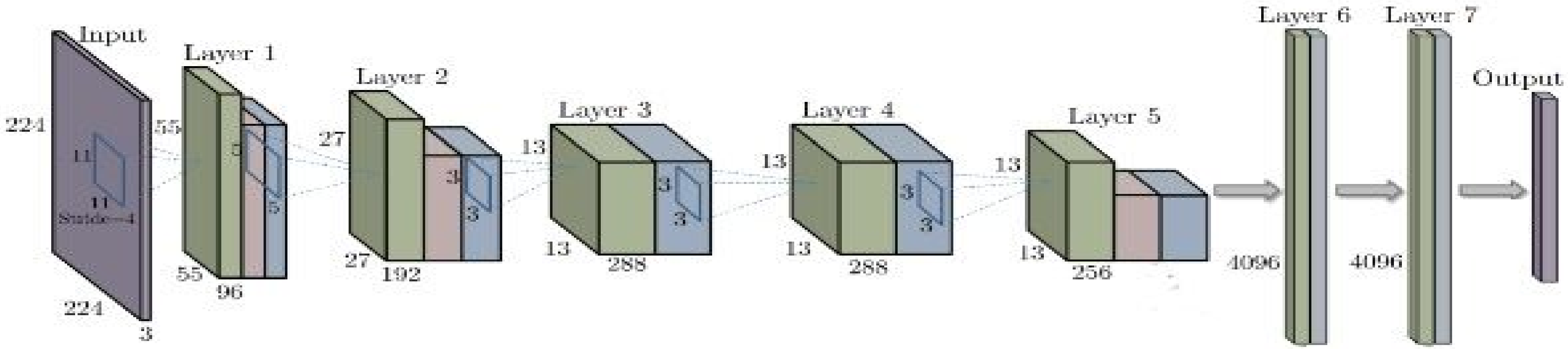
ConvNet in Practice : HyperParameters

- Hyper parameters
  - Weigth Decay : L2 regularisation
  - Momentum
  - **Learning Rate policy**
  - **Shuffle the data**
  - **Normalize (cf BN)**
  - **Dropout**

Transfer learning (fine-tuning)

Problem: CNNs require huge training data to learn the millions of parameters
Solution: Learn domain specific features by transfer learning

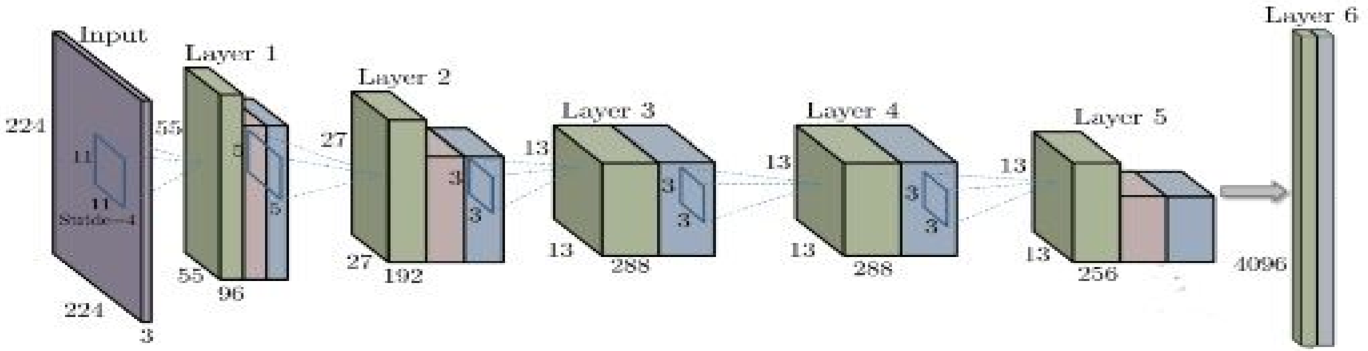
1. Train CNN on a generalist image dataset with millions of images



Transfer learning (fine-tuning)

Problem: CNNs require huge training data to learn the millions of parameters
Solution: Learn domain specific features by transfer learning

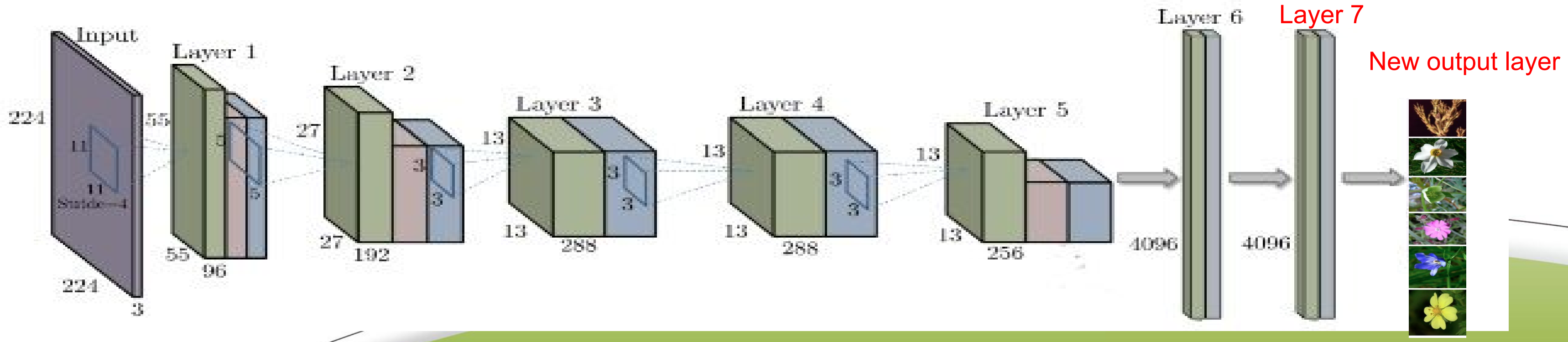
1. Train CNN on a generalist image dataset with millions of images
2. Keep the weights of the lowest layers but remove/reset the top layers



Transfer learning (fine-tuning)

Problem: CNNs require huge training data to learn the millions of parameters
Solution: Learn domain specific features by transfer learning

1. Train CNN on a generalist image dataset with millions of images
2. Keep the weights of the lowest layers but remove/reset the top layers
3. Feed forward and back-propagate new domain specific images (with usually a different number of classes C)



The power of transfer learning

Transfer learning usually works for any domain

	Trademark Logos	Car models	Paris Buildings	Aircraft models	Bird species	Flower species
GoogLeNet trained from scratch	67.7%	59.3%	55.3%	72.7%	24.4%	59.5%
GoogLeNet pre-trained on ImageNet	**87.5%**	**79.9%**	**71.3%**	**88.1%**	**72.4%**	**89.5%**

Table 1 - accuracy measured on several fine-grained image classification datasets

Even very specific ones:

Rice seeds varieties recognition
100 classes, 1 500 texture images

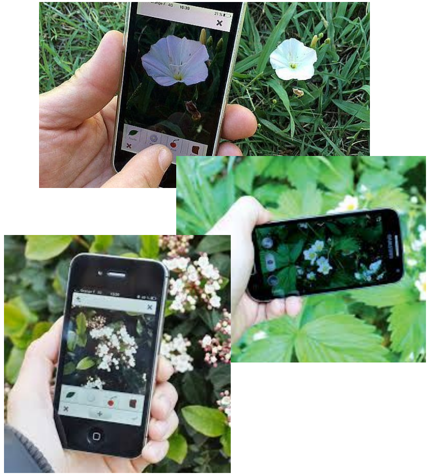
GoogLeNet trained from scratch	8.8%
GoogLeNet pre-trained on ImageNet	**52.4%**

Herbaria species recognition
255 classes, 11K herbaria sheets

GoogLeNet trained from scratch	58.1%
GoogLeNet pre-trained on ImageNet	**70.4%**

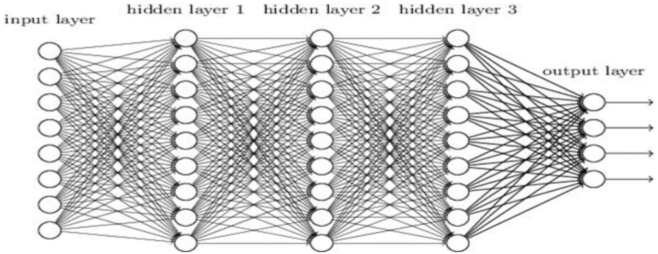
Plant species recognition: Pl@ntNet

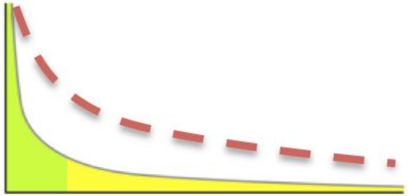
informatics mathematics
INRA
SCIENCE & IMPACT
IRD
Institut de recherche
pour le développement
cirad
agropolis fondation
Tela Botanica



Plant species recognition: Pl@ntNet

Collaborative application





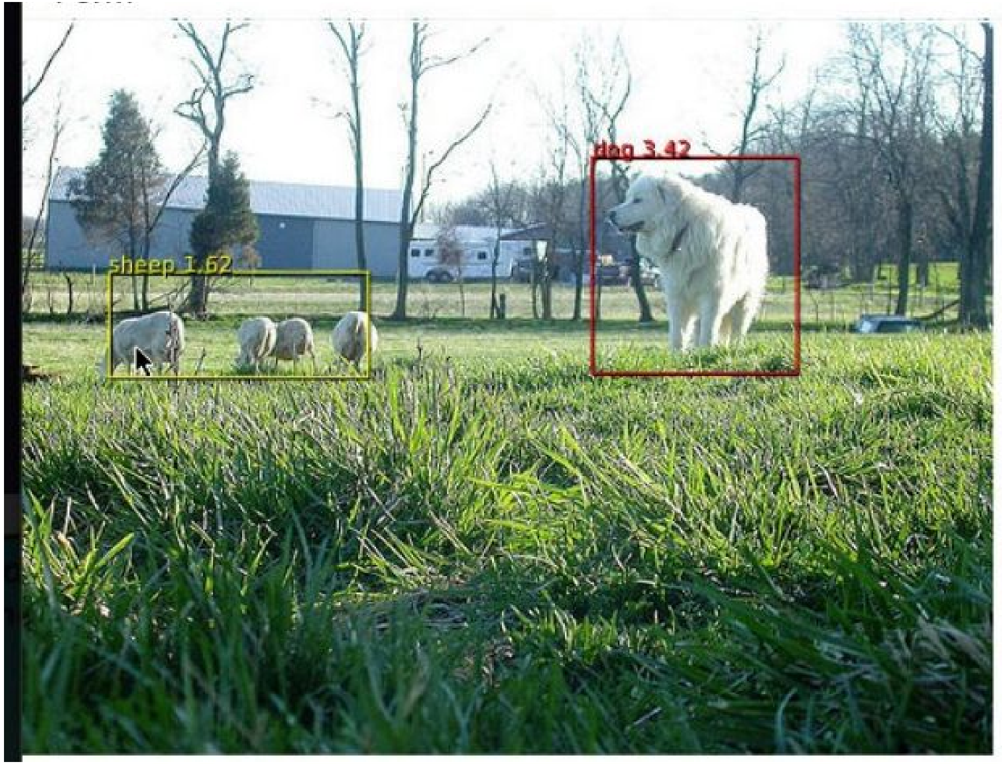
Database

Localization / Segmentation

- **Apply convnet with a sliding window over the image at multiple scales**
- **Important note: it's very cheap to slide a convnet over an image**
  - Just compute the convolutions over the whole image and replicate the fully-connected layers

[Y. LeCun Collège de France]

Localization / Segmentation

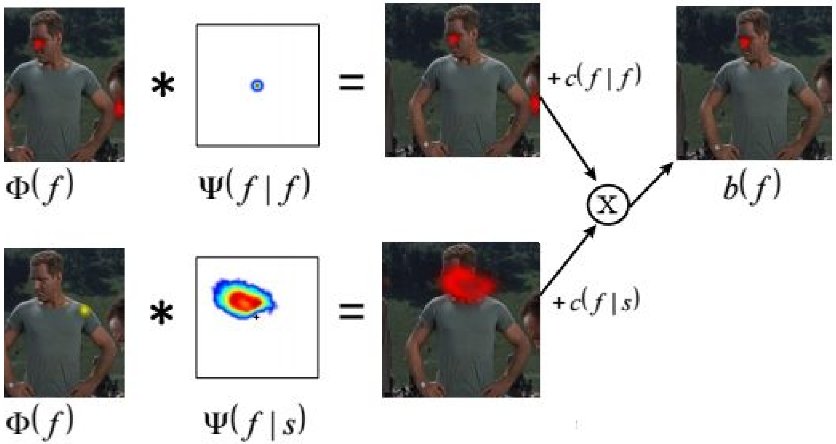


Localization / Segmentation

Start with a tree graphical model

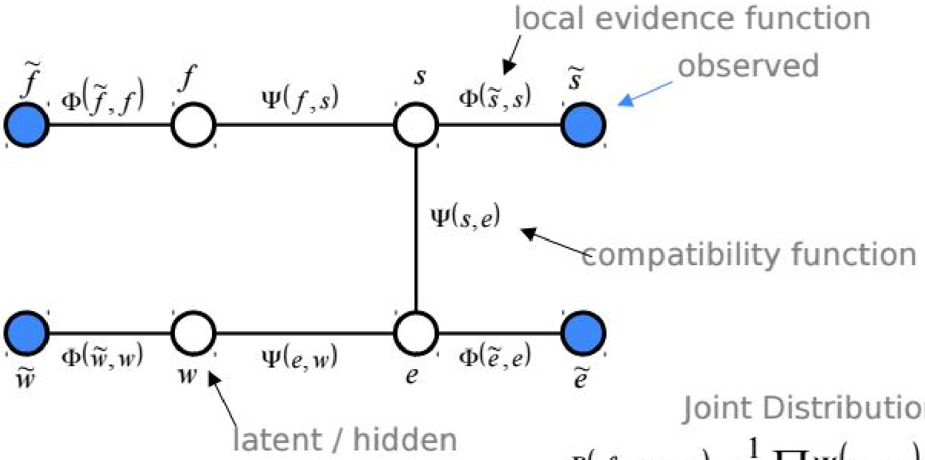
... And approximate it

$$b(f)=\Phi(f)\prod_{i}\left(\Phi(x_i)*\Psi(f\mid x_i)+c(f\mid x_i)\right)$$



Start with a tree graphical model

MRF over spatial locations



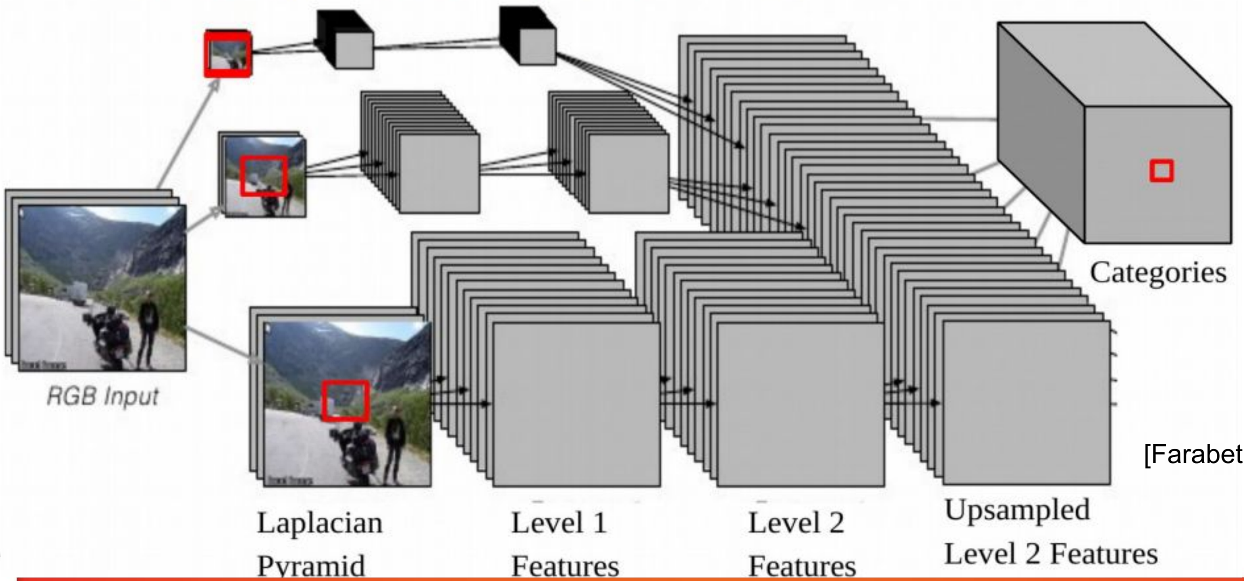
Joint Distribution:

$$P(f,s,e,w)=\frac{1}{Z}\prod_{i,j}\Psi(x_i,x_j)\prod_{i}\Phi(x_i,\tilde{x}_i)$$

Localization / Segmentation

Pixel Labelling

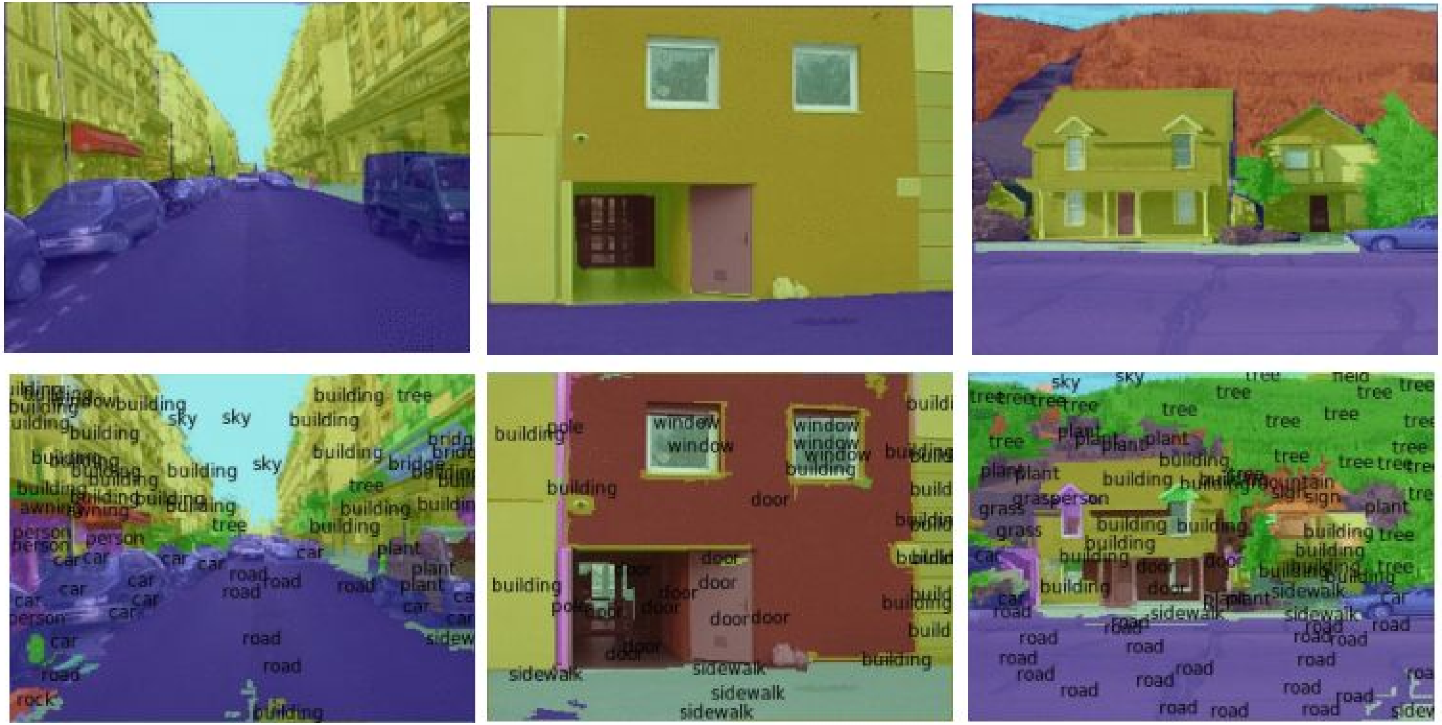
- **Each output sees a large input context:**
  - **46x46** window at full rez; **92x92** at ½ rez; **184x184** at ¼ rez
  - [7x7conv]->[2x2pool]->[7x7conv]->[2x2pool]->[7x7conv]->
  - Trained supervised on fully-labeled images



[Farabet et al. 2013]

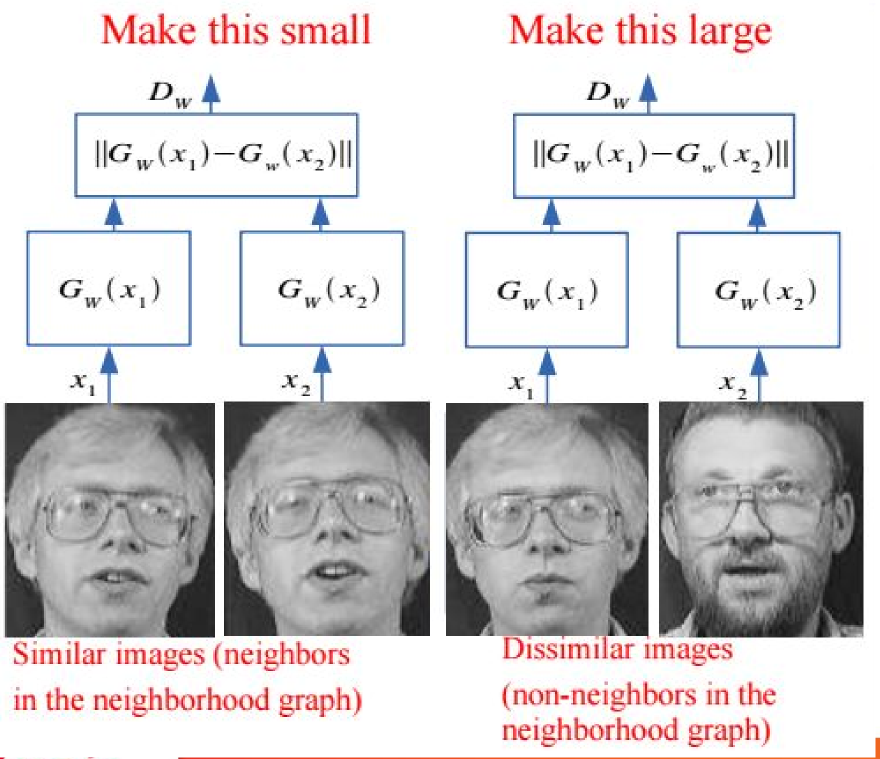
Pixel Labelling

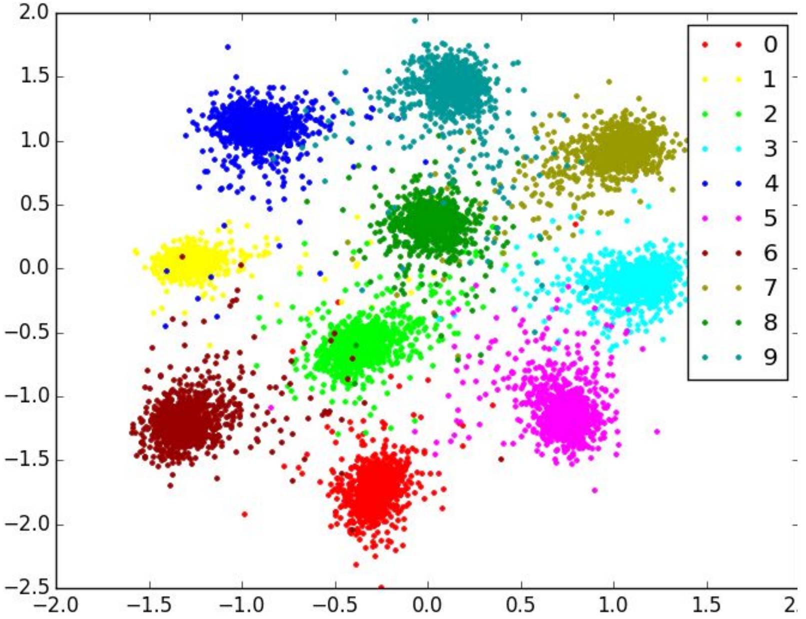
[Farabet et al. 2013]



Metric Learning with Siamese Networks

[Chopra et al. 2005]





Deep Dream

- Force the network to over-interpret what it sees.
- Amplify maximally activated units + backpropagate signal gradient until the input layer.

Image Captioning via attention based models

[Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. K. Xu et al. 2015]

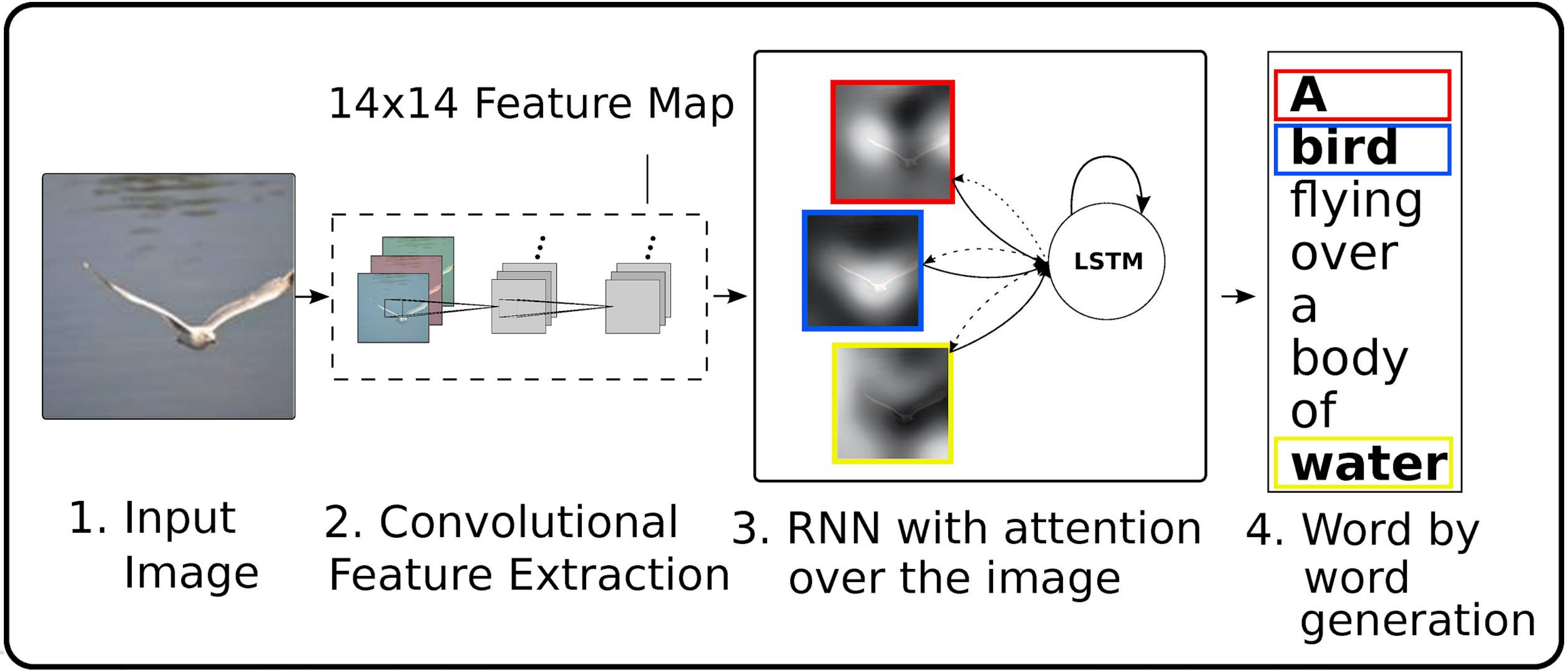
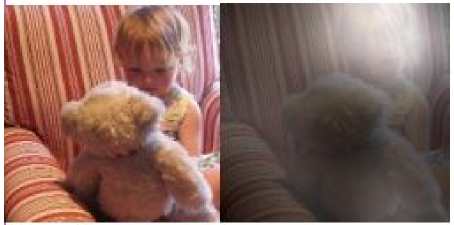


Image Captioning via attention based models

A woman is throwing a frisbee in a park.

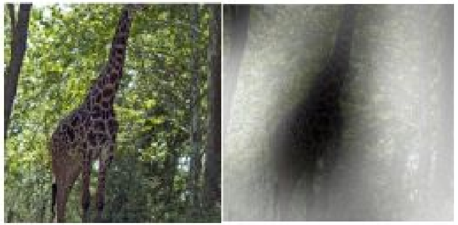
A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

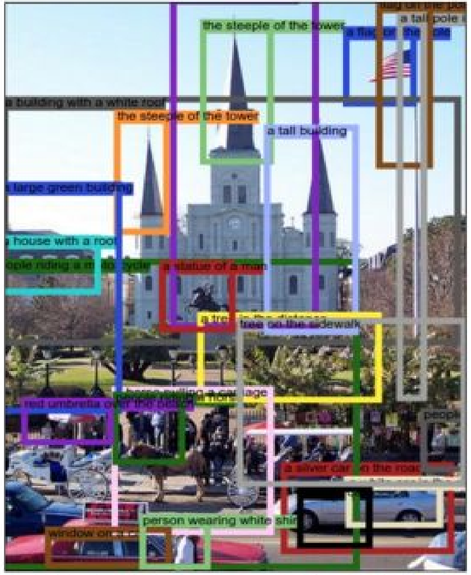


A giraffe standing in a forest with trees in the background.

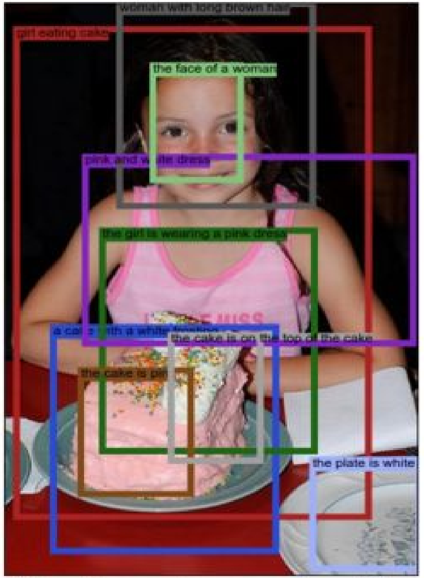
[From Bengio&LeCun tutorial NIPS 2015]

Image Captioning via attention based models

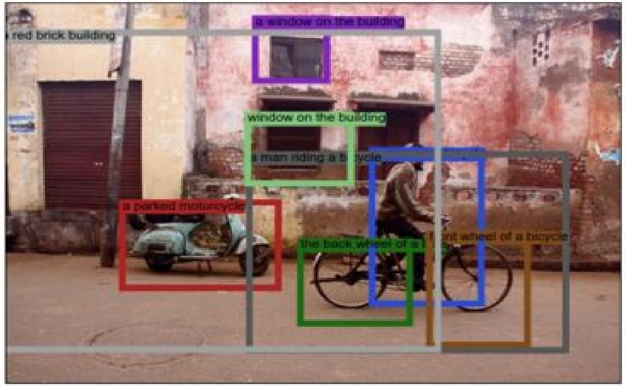
Image Captioning via attention based models



a silver car on the road. a flag on the pole. a building with a white roof. people riding a motorcycle. flag on the pole. a tower on a building. a tall pole in the background. the steeple of the tower. a tall building. a house with a roof. the steeple of the tower. a tree in the distance. a white car in the street. a horse pulling a carriage. a silver car. a statue of a man. a large green building. people standing on the sidewalk. people riding a horse. window on a car. red umbrella over the beach. tree on the sidewalk. person wearing white shirt.



girl eating cake. a cake with a white frosting. woman with long brown hair. the girl is wearing a pink dress. the cake is pink. pink and white dress. the cake is on the top of the cake. the face of a woman. the plate is white.



a parked motorcycle. a man on a bicycle. a man riding a bicycle. the back wheel of a bike. front wheel of a bicycle. a window on the building. a red brick building. window on the building.

“Reverse Image Captioning” :

head of a giraffe | *legs of a zebra* | *red and white sign* | *white tennis shoes* | *hands holding a phone* | *front wheel of a bus*

Outline

1. **What are we fighting against ?**

    Invariances + Curse of dimensionality

    Priors to learn good data representation (toward deep representation learning)

2. **Learning procedures for deep architectures**

    From Artificial Neural Networks → Deep Convolutional Neural Networks (ConvNet)

    Recent advances : why ConvNet got famous so late ?

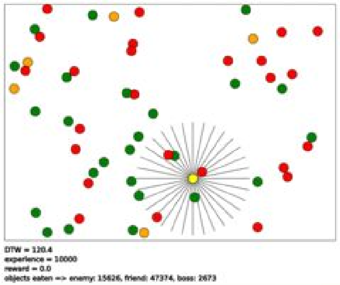
    Applications

3. **Unsupervised Learning**

Machine Learning

- Three types of learning :
  - **<u>Supervised</u>**: Provide the labels Y while learning. Learns to predict Y given input X.

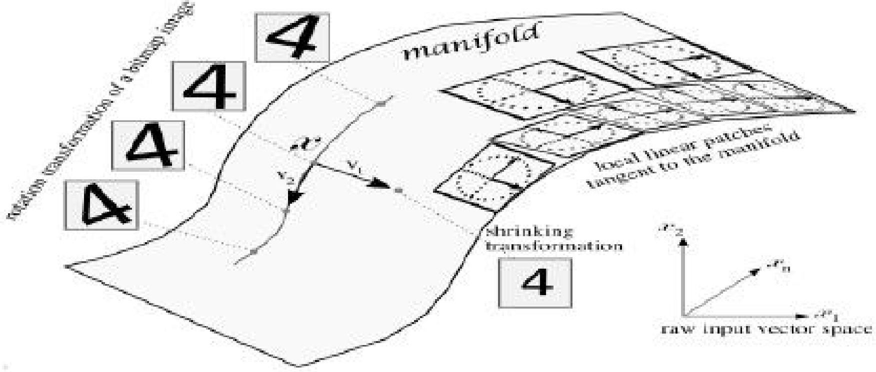
  - **<u>Unsupervised</u>**: Do not privilege some variables rather tha others. Try to catch all the interesting information that is contained in the data.



  - <u>Reinforcement</u>: Wait for the machine to produce the good behavior and give it a reward when it does (very long !).

Unsupervised Learning

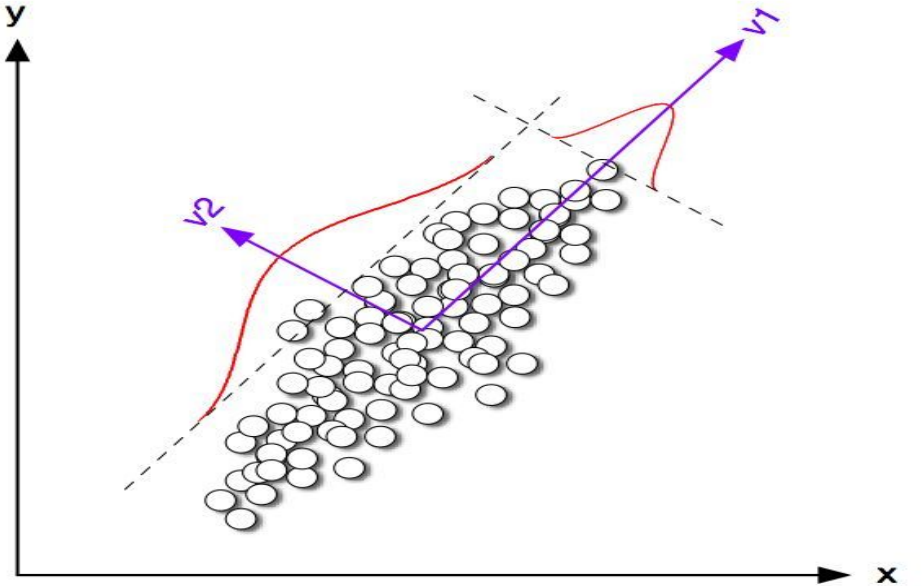
- Learn the generative process of the data: P(X|Theta)
- Manifold Learning:
  - Linear unsupervised model: PCA
  - Non Linear and deep models:
    - Autoencoders
    - Generative Adversarial Networks



Principal Component Analysis (PCA)

- Project data linearly into lower dimensional space

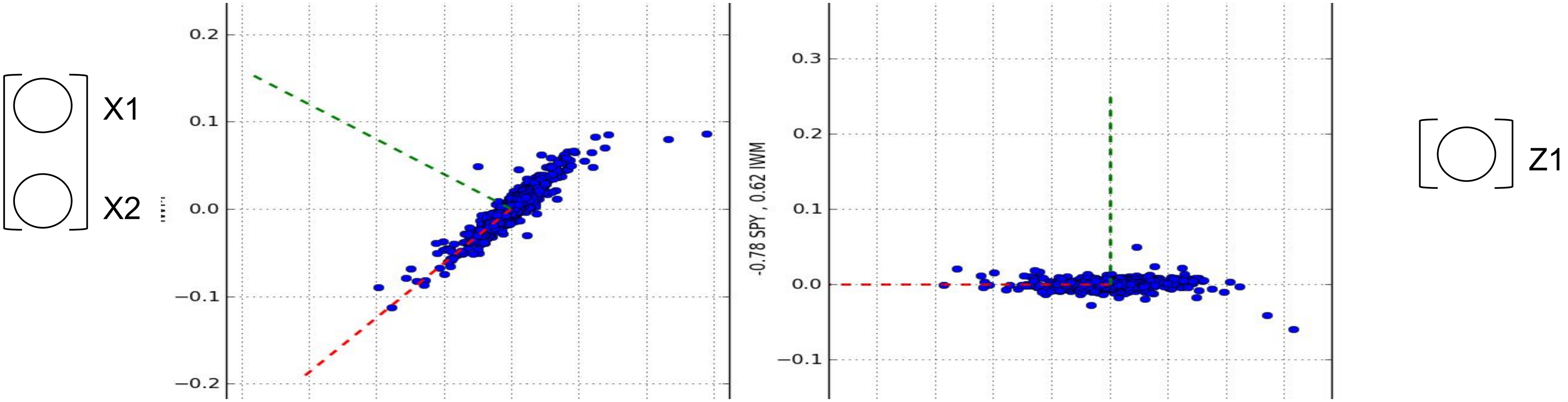
  → Find the rotation matrix to align data with axis of maximal variance



Principal Component Analysis (PCA)

- Project data linearly into lower dimensional space

  → Find the rotation matrix to align data with axis of maximal variance
- Allows us to keep the structure of data of variable are linearly correlated



Autoencoder

- Learn a model to:
  - Project data into a non linear intermediate embedding (Coding)
  - Reconstruct its own input from the codes (Decoding)
- PCA's eigen directions span the same space than linear autoencoder's

Coding function:

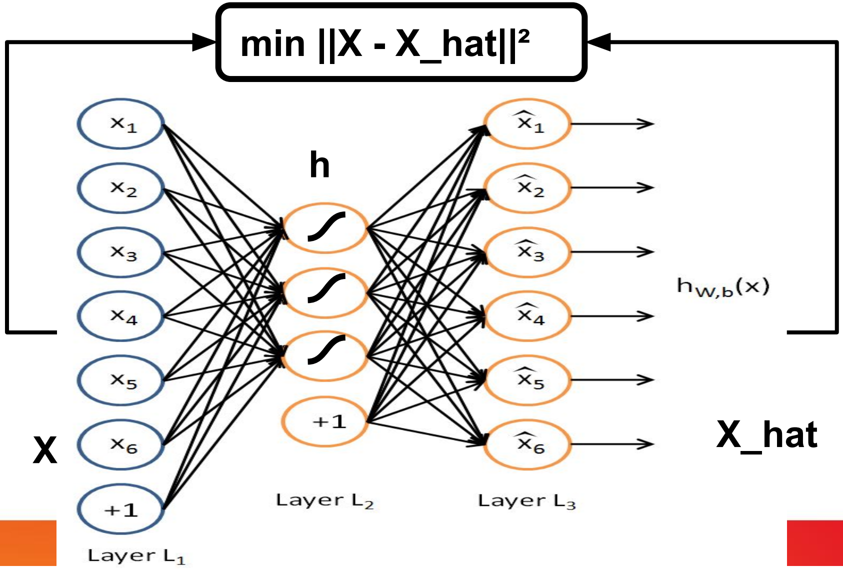
$$\mathbf{h}^i = f_\theta(\mathbf{x}^i) = \sigma(\mathbf{W_f}\mathbf{x}^i + \mathbf{b_f})$$

Decoding function:

$$\hat{\mathbf{x}}^i = g_\theta(\mathbf{h}^i) = \sigma(\mathbf{W_g}\mathbf{h}^i + \mathbf{b_g})$$

Reconstruction objective:

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2$$

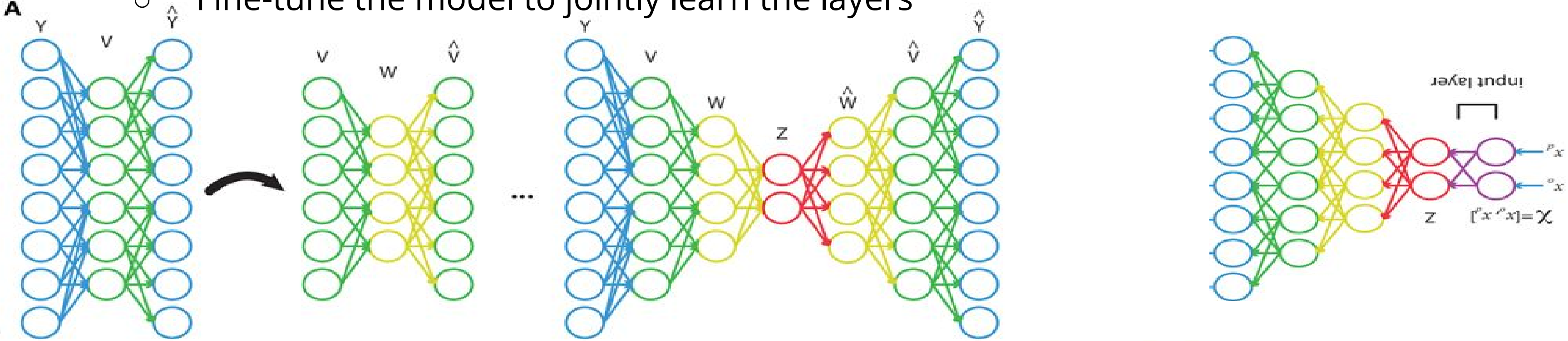


Autoencoder

- It is more efficient in practice to learn the layer separately.

  → Actually with ReLU, that's okay.

- Stacked autoencoders:
  - Train the first autoencoder layer to reconstruct the input
  - Use the intermediate representation as input to the next layer and re-apply the process
  - Fine-tune the model to jointly learn the layers

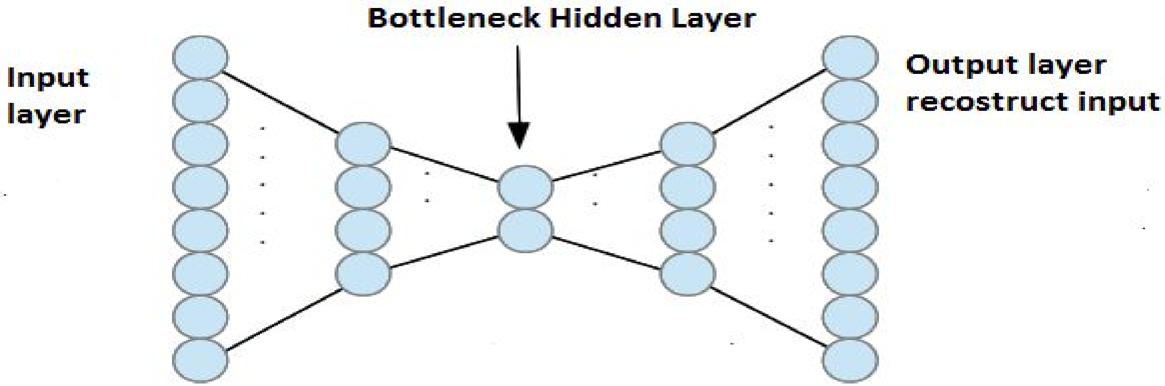


Regularized Autoencoders

- Problem of raw autoencoder: it is likely to learn the identity mapping
- Solution: regularize it to prevent it from doing that:
  - Bottleneck autoencoder
  - Sparse Autoencoder
  - Denoising/Contractive Autoencoder
  - Generative Adversarial Networks (GAN) and Adversarial Autoencoders

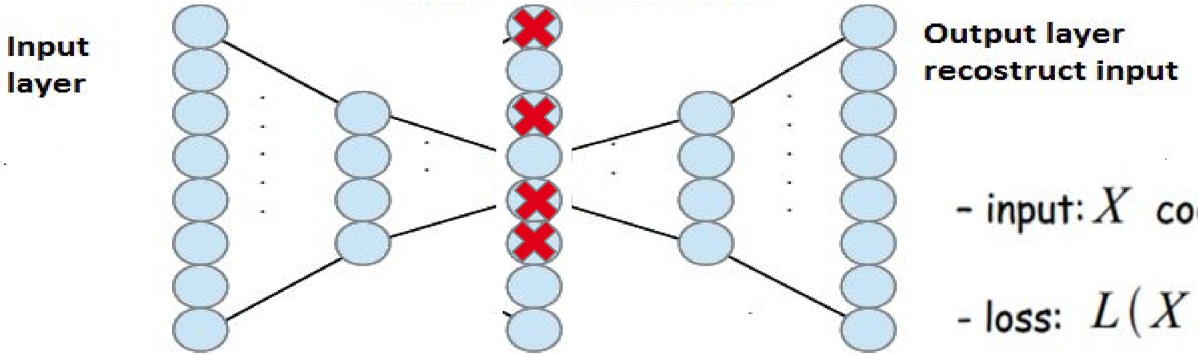
BottleNecked Autoencoders

- Force information to concentrate on a few number of latent variables
- If we can reconstruct well from such low dimensional representations, then we have forced the model to capture useful information
- Problem: we have to assume the dimension of the latent space



Sparse Autoencoders

- Learn an overcomplete representation scheme
- Penalize the model to produce dense codes (L1 penalty)
- Allows the model do choose the intrinsic dimensionality of the data



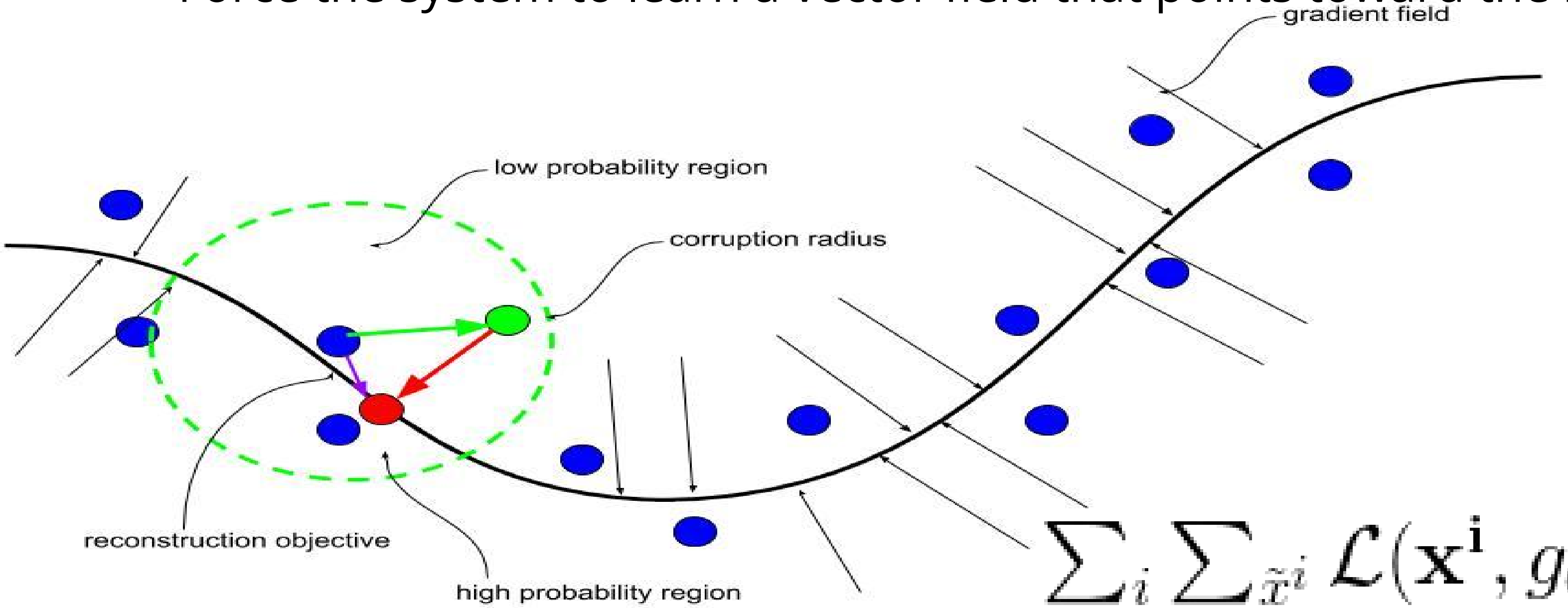
- input: X code: $h = W^T X$

- loss: $L(X;W) = \|W h - X\|^2 + \lambda \sum_j |h_j|$

Denoising Autoencoders

[P. Vincent et al. 2010]

- Learn to reconstruct a corrupted input $\tilde{\mathbf{x}}^{\mathbf{i}} \sim C(\tilde{\mathbf{x}}^{\mathbf{i}}|\mathbf{x}^{\mathbf{i}}) = \mathcal{N}(\mathbf{x}^{\mathbf{i}}, \sigma^2\mathbf{I_d})$
- Force the system to learn a vector field that points toward the manifold.



$$\sum_i \sum_{\tilde{x}^i} \mathcal{L}(\mathbf{x}^{\mathbf{i}}, g_\theta(f_\theta(\tilde{\mathbf{x}}^{\mathbf{i}})))$$

Contractive Autoencoders

[S. Rifai et al. 2010]

- Penalize high curvature of the manifold in the latent space

  → Penalize high values of the terms of the Jacobian of the coder

$$\sum_i \mathcal{L}(\mathbf{x^i}, g_\theta(f_\theta(\mathbf{x^i}))) + \lambda||\mathbf{J}(\mathbf{x^i})||_{\mathcal{F}}^2 \qquad ||\mathbf{J}(\mathbf{x^i})||_{\mathcal{F}}^2 = \sum_j \sum_k \left(\frac{\partial f_{\theta_k}}{\partial \mathbf{x_j}}|_{\mathbf{x^i}} \right)^2$$

- Example: Sigmoidal Contractive Autoencoder

$$||\mathbf{J}(\mathbf{x^i})||_{\mathcal{F}}^2 = \sum_k \{ \boxed{(f_\theta(\mathbf{x^i})_k(1-f_\theta(\mathbf{x^i})_k))^2} \sum_j \mathbf{W_{jk}}^2 \}$$

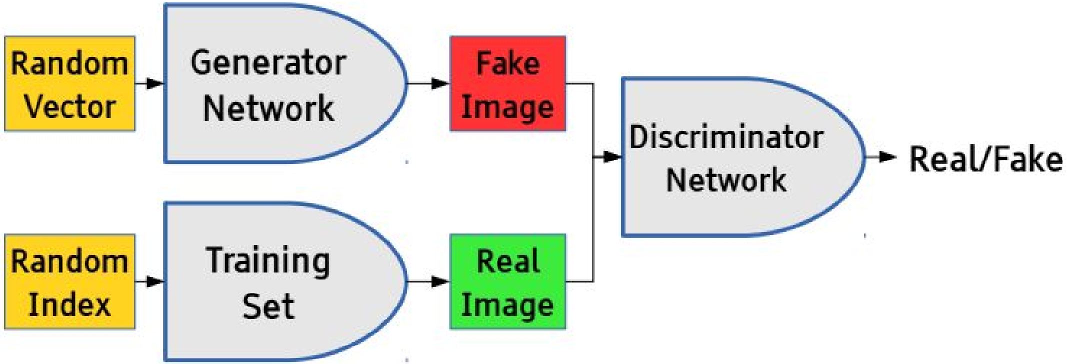
- Allows to learn robust features while learning to reconstruct the input

  → Contracts the input space in "interesting" directions of variation = Manifold Learning

Generative Adversarial Network (GAN)

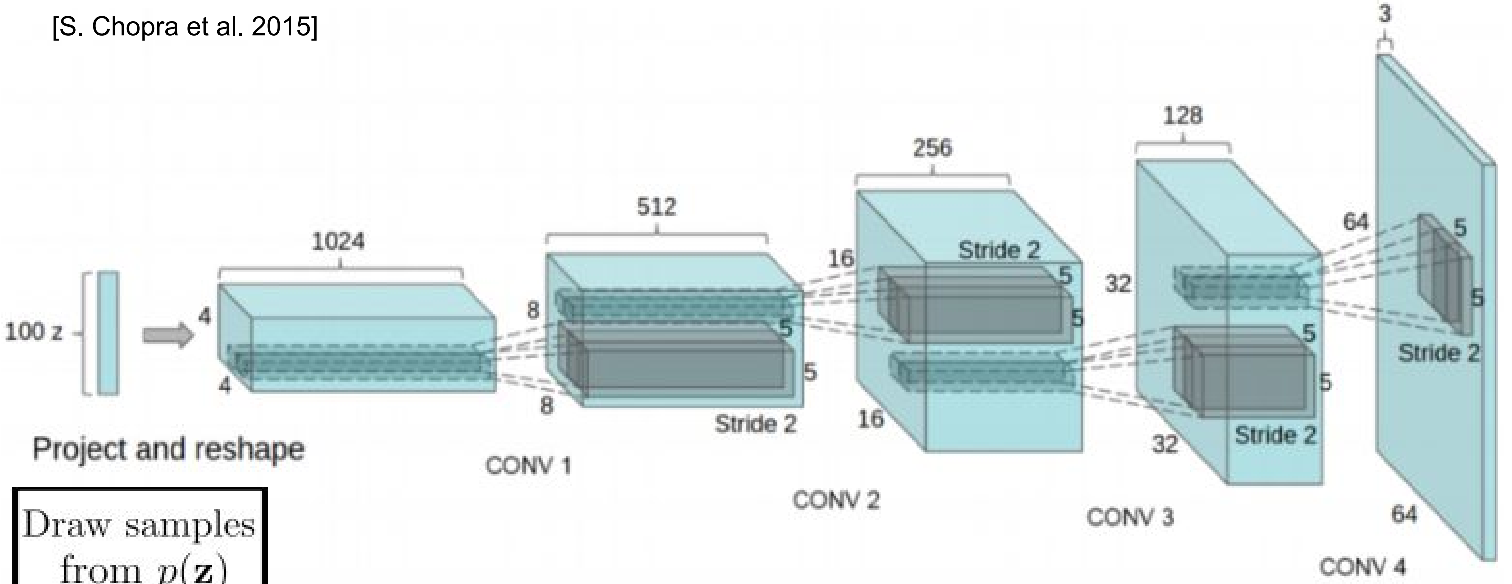
[I. GoodFellow et al. 2014]

- [Goodfellow et al. NIPS 2014]
- Generator net maps random numbers to image
- Discriminator learns to tell real from fake images.
- Generator can cheat: it knows the gradient of the output of the discriminator with respect to its input



CNN Generative Adversarial Network (GAN)

[S. Chopra et al. 2015]



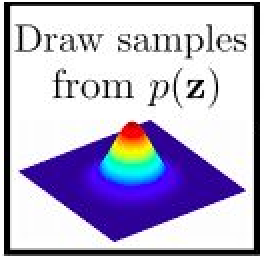


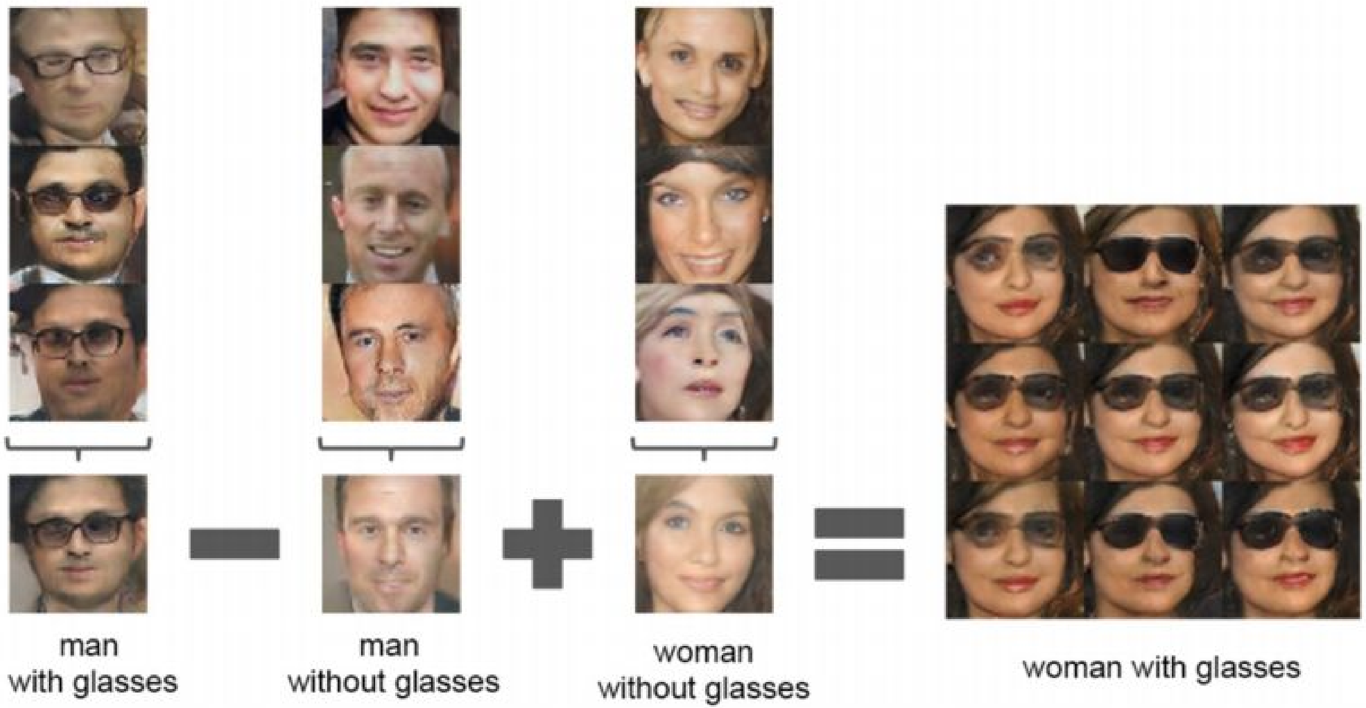
Image Generation with GAN

Image Generation with GAN

Image Generation with GAN

Logic with Deep Learning

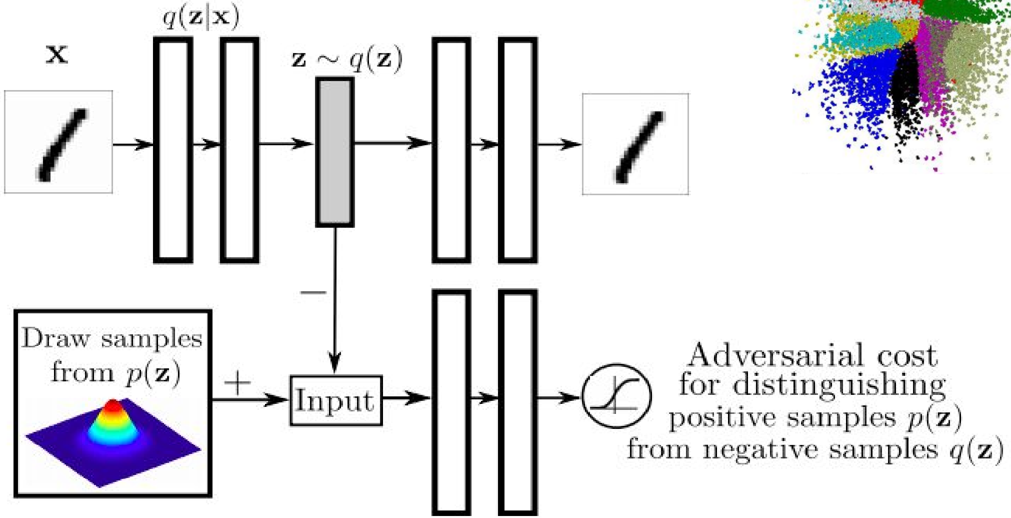
– [Radford, Metz, Chintala 2015]

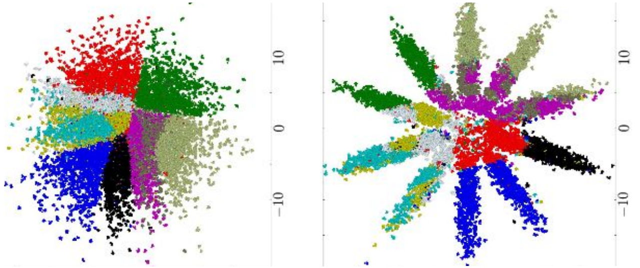


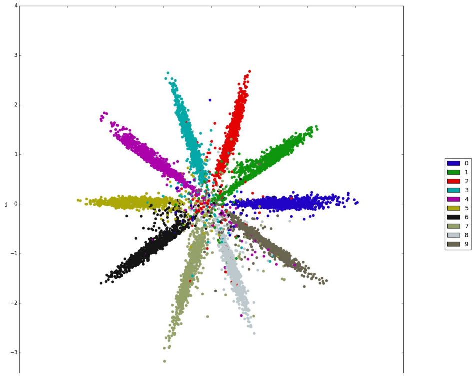
Adversarial Autoencoders

[A. Makhzani et al. 2015]

- Use adversarial regularisation to force the shape of the distribution in the latent space.

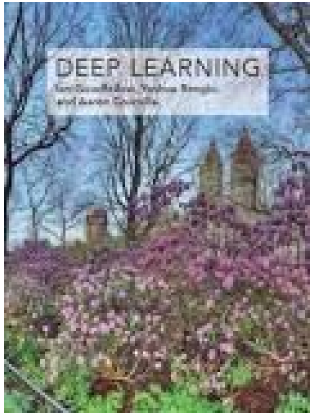






Ressources

- Cours de Yann LeCun au Collège de France
- Intervention de Stéphane Mallat au Collège de France
- The Deep Learning Book (The Holy Bible)



Questions