

Hi! Paris 2021:

Deep Learning: beyond TensorFlow

(followed by a lab' on i-NNs with TensorFlow)

Edouard Oyallon

edouard.oyallon@lip6.fr

CNRS, LIP6



Intro: Image classification or generation, solved by Deep Learning

A. Engineering Deep Neural Networks

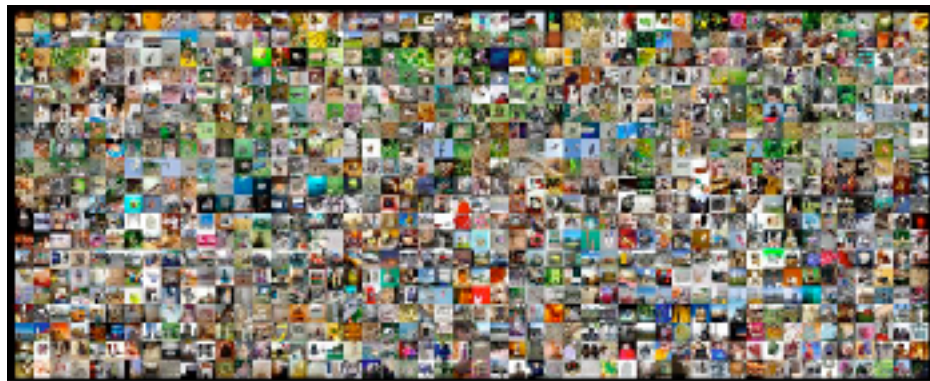
1. Deep Neural Networks models...
2. ... that require many recipes to be trained.

B. Understanding Deep Convolutional Neural Networks

1. Convolutional layers in Convolutional Neural Networks
2. Invariant Representations and Deep Learning.

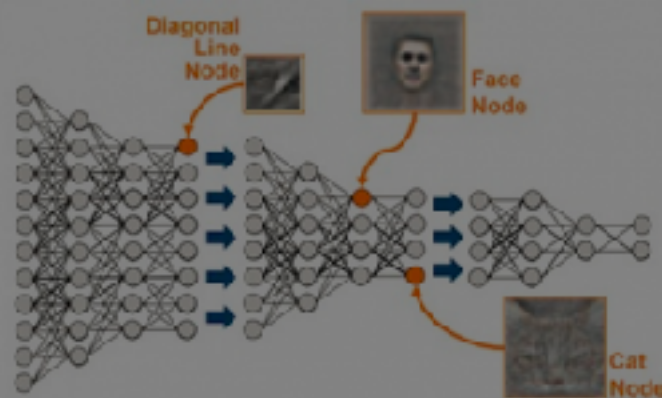
C. Under the hood of Neural Networks

1. Classification mechanisms
2. A mysterious black-box
3. Few results on shallow Neural Networks

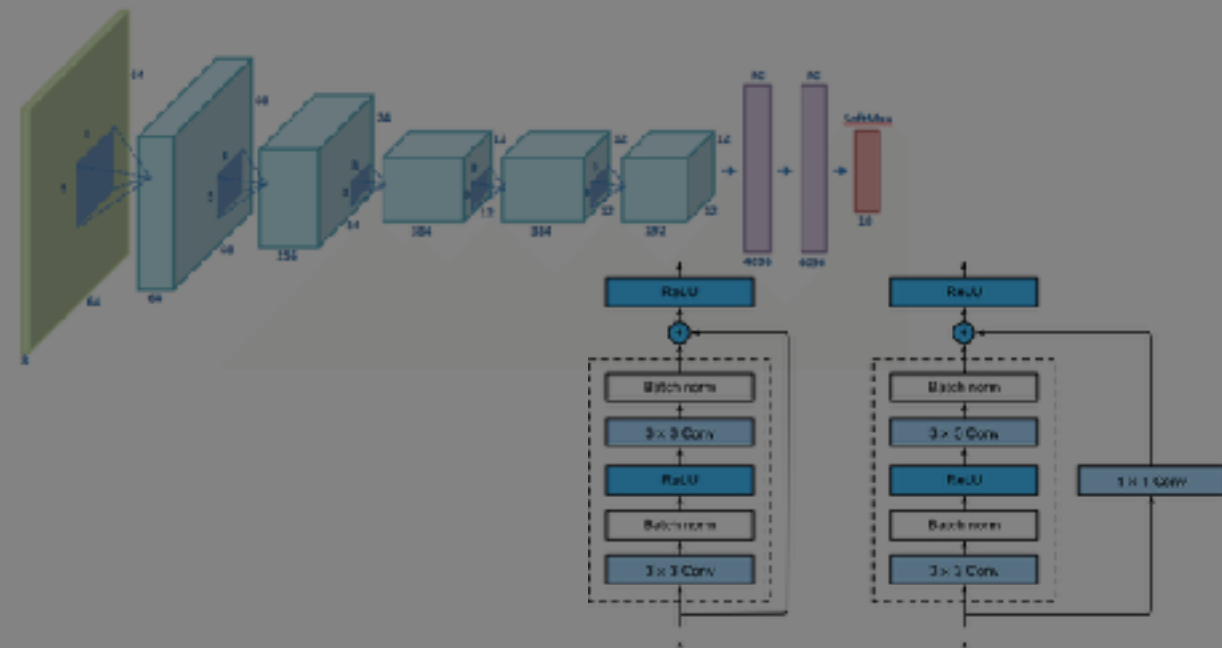
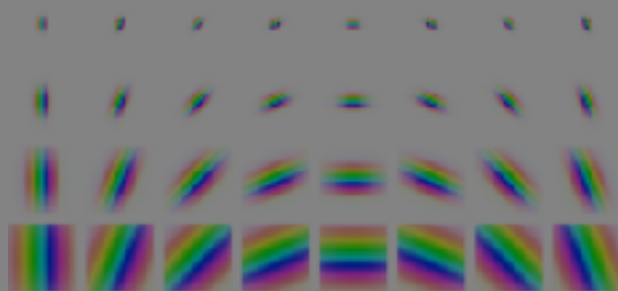


Introduction to high-dimensional tasks

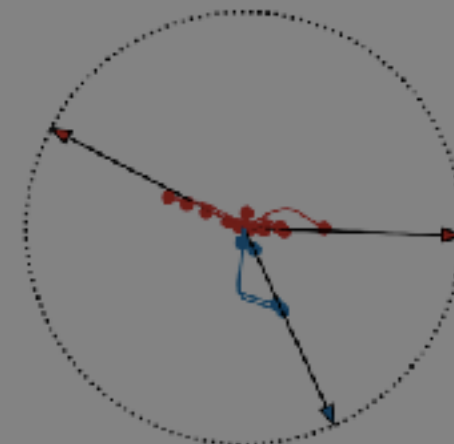
4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5
7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8



Understanding Convolutional Neural Networks



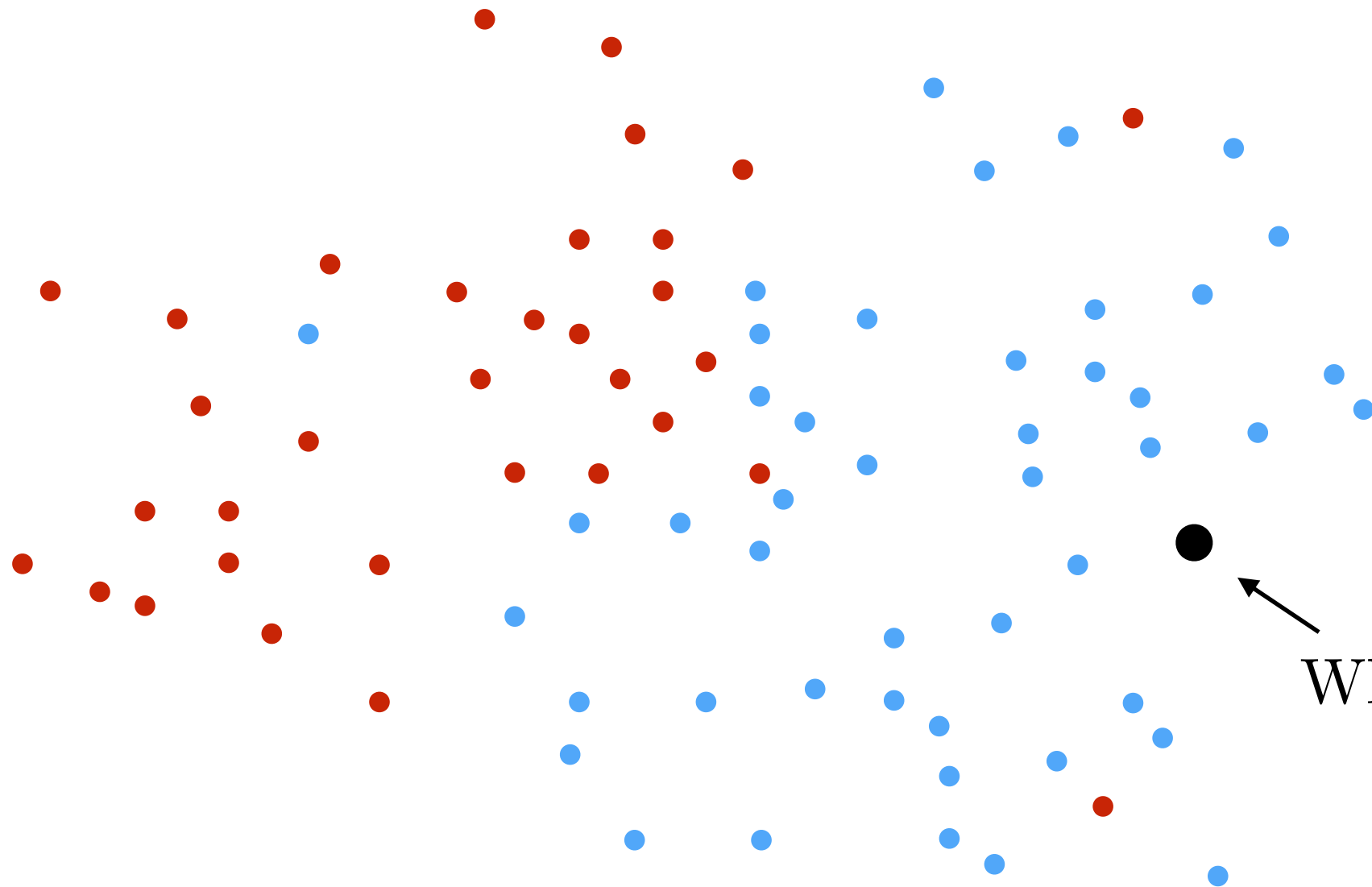
Engineering Deep Neural Networks



Under the Hood of Neural Networks

$$C_f = \int_{\mathbb{R}^D} \|\omega\|_1 |\hat{f}(\omega)| d\omega$$

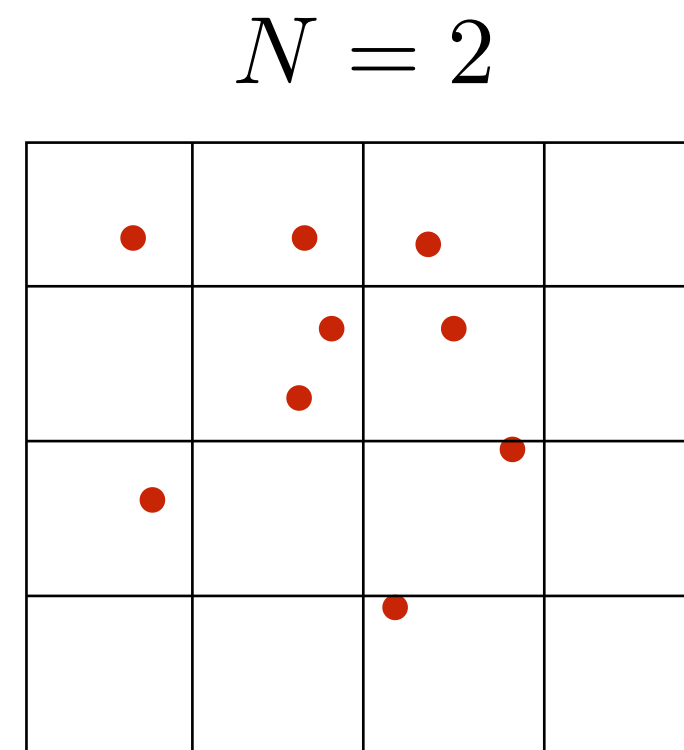
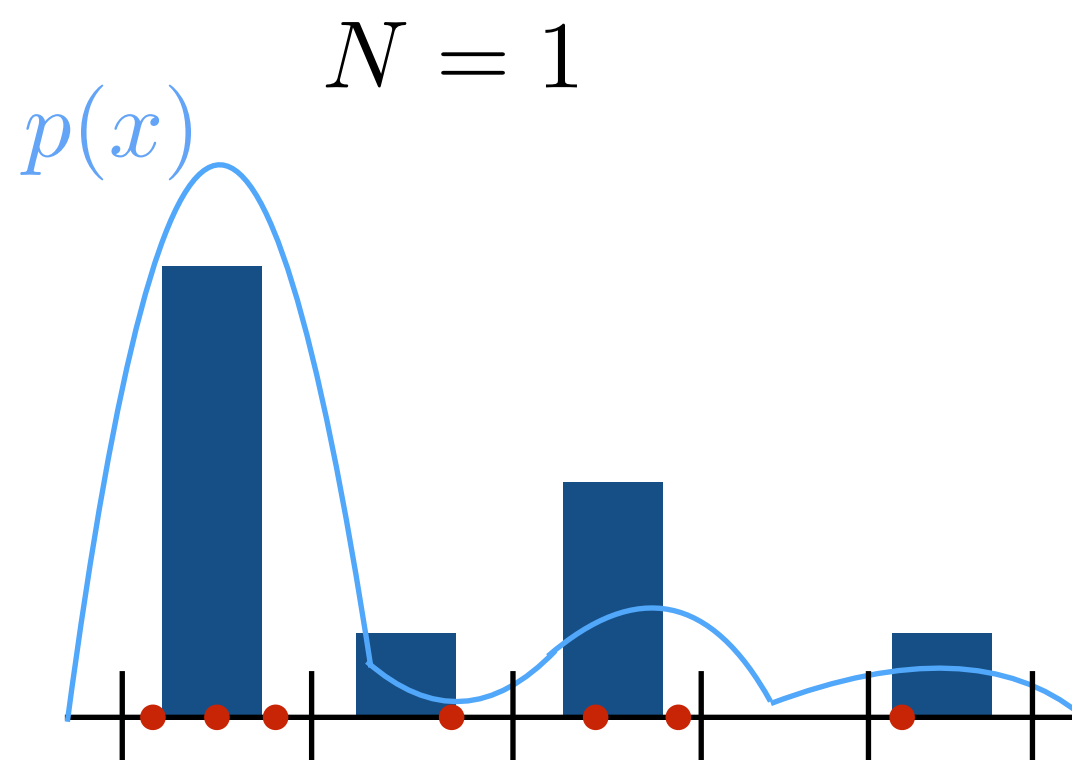
Solving High-dimensional tasks with DNNs



Which color should be this circle?

An example of supervised task: classification

- Pdfs are difficult to estimate in high dimension.

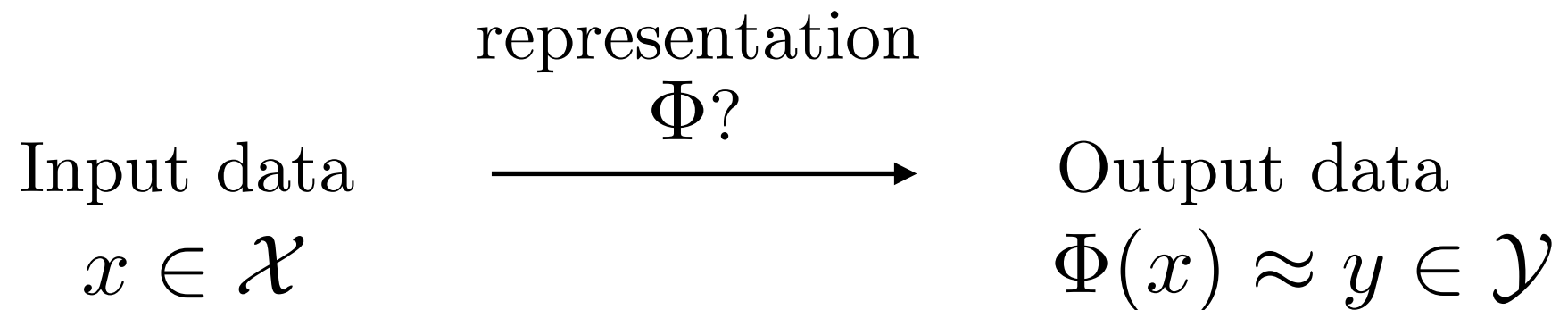


- For a fixed number of points and bin size, as N increases, the bins are likely to be empty.

Curse of dimensionality:
occurs in many machine learning problems

$\mathcal{X} = \mathbb{R}^2$ Samples space

$\mathcal{Y} = \{\bullet, \bullet\}$ Labels



- Estimating a label y from a sample x , by training a model Φ on a training set. Validation of the model is done on a different test set.
- Examples: prediction, regression, classification,...
- Best setting: dimensions of x and y is small, \mathcal{X} large

High Dimensional classification

$$(x_i, y_i) \in \mathbb{R}^{224^2} \times \{1, \dots, 1000\}, i < 10^6 \longrightarrow \hat{y}(x)?$$



"Rhinos"

Estimation problem

Training set to
predict labels



"Rhino"

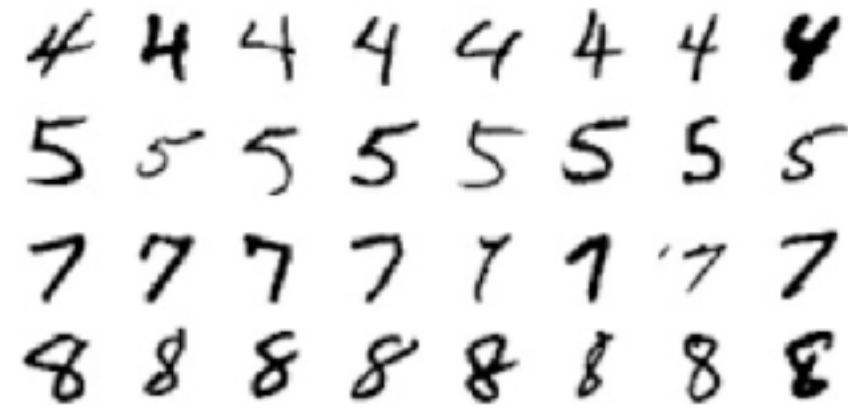


Not a "rhino"

- How to address a supervised task:

1. Propose a model of your data.

Ex.: MNIST (60k samples)



Small deformations
+ Translation

2. Design a representation.

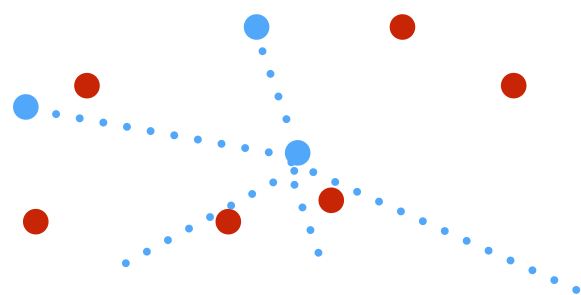
Ex.: SIFT, Bag-of-Words

Achieves translation invariance, linearises deformations.

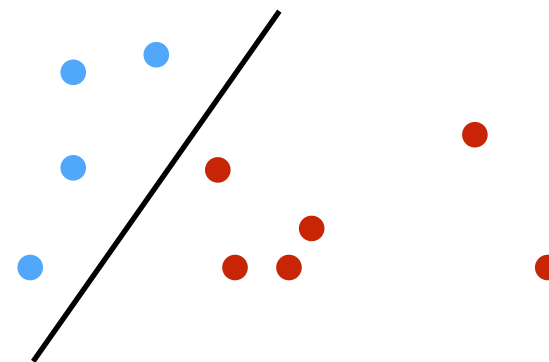
3. Propose a (convex) classifier.

Ex.: Linear SVM.

... Displacement



Φ
+ projection



4. Obtain reasonable performances.

1. No model known on real images
2. Limited *a priori*, except translation invariance
3. Learn each parameters...
4. Obtain the best performances

The reason of their success is unclear...

- ImageNet 2012: (350GB)
1 million training images, 1 000 classes
400 000 test images
Large coloured images of various sizes
- Labels obtained via Amazon Turk (complex process that requires human labelling)

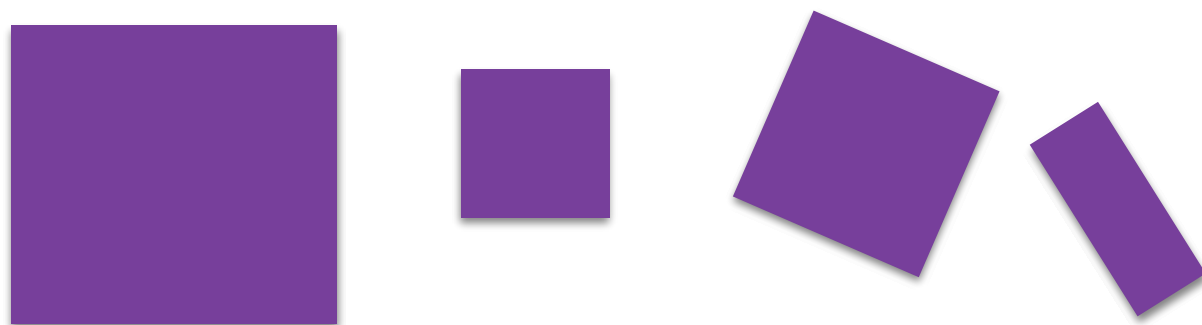
Ref.: image-net.org



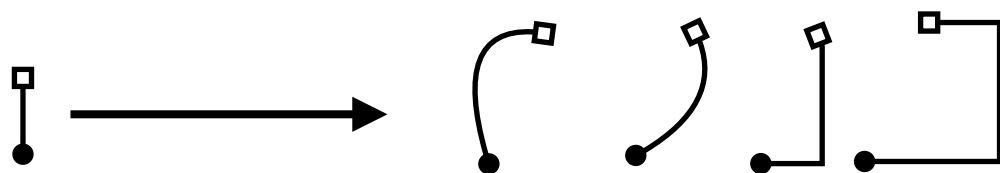
Image variabilities

Geometric variability

Groups acting on images:
translation, rotation, scaling



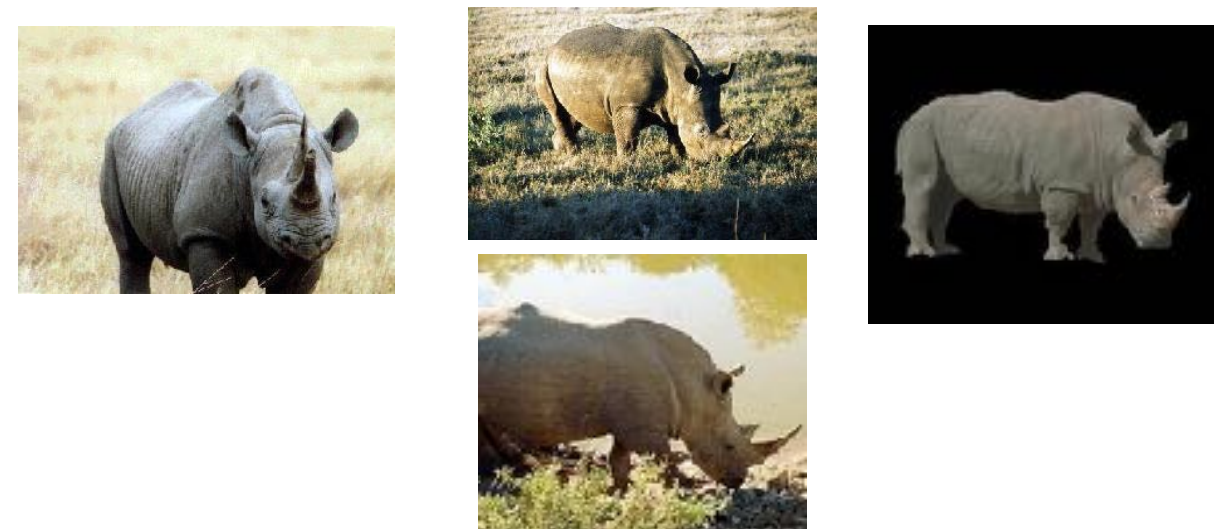
Other sources : luminosity, occlusion,
small deformations



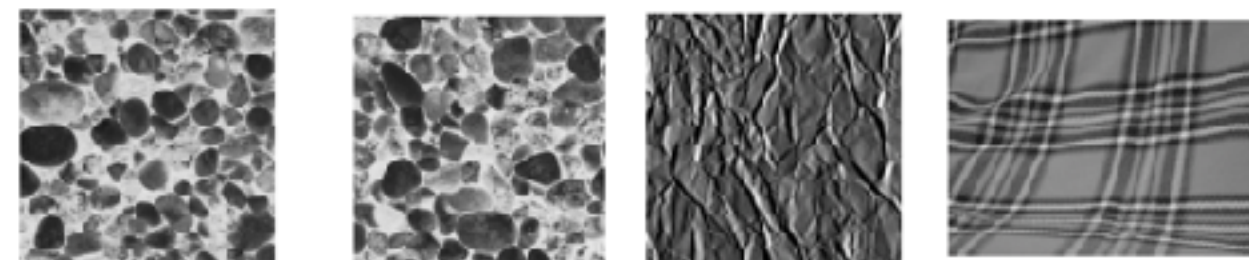
Class variability

Intraclass variability

Not informative



Extraclass variability



High variance: hard to reduce!

- **Invariance** to group G of transformation (e.g. roto-translation):

$$\forall x, \forall g \in G, \Phi(g.x) = \Phi(x)$$

- **Stability** to noise

$$\forall x, y, \|\Phi(x) - \Phi(y)\|_2 \leq \|x - y\|_2$$

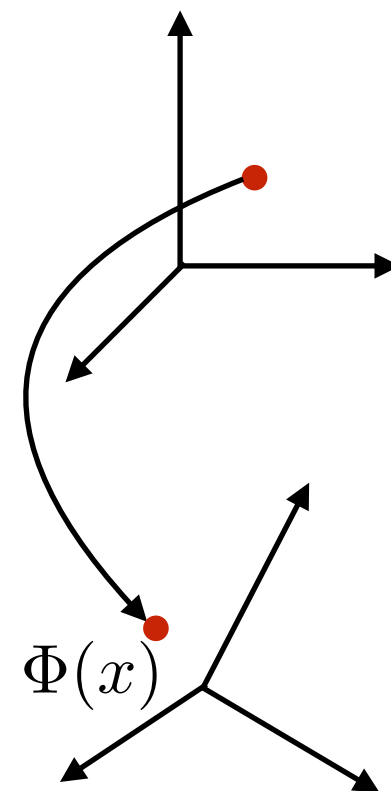
- **Reconstruction** properties

$$y = \Phi(x) \iff x = \Phi^{-1}(y)$$

- **Linear separation** of the different classes

$$\forall i \neq j, \|E(\Phi(X_i)) - E(\Phi(X_j))\|_2 \gg 1$$

$$\forall i, \sigma(\Phi(X_i)) \ll 1 \quad \text{Can be difficult to handcraft..}$$



Years of
research...

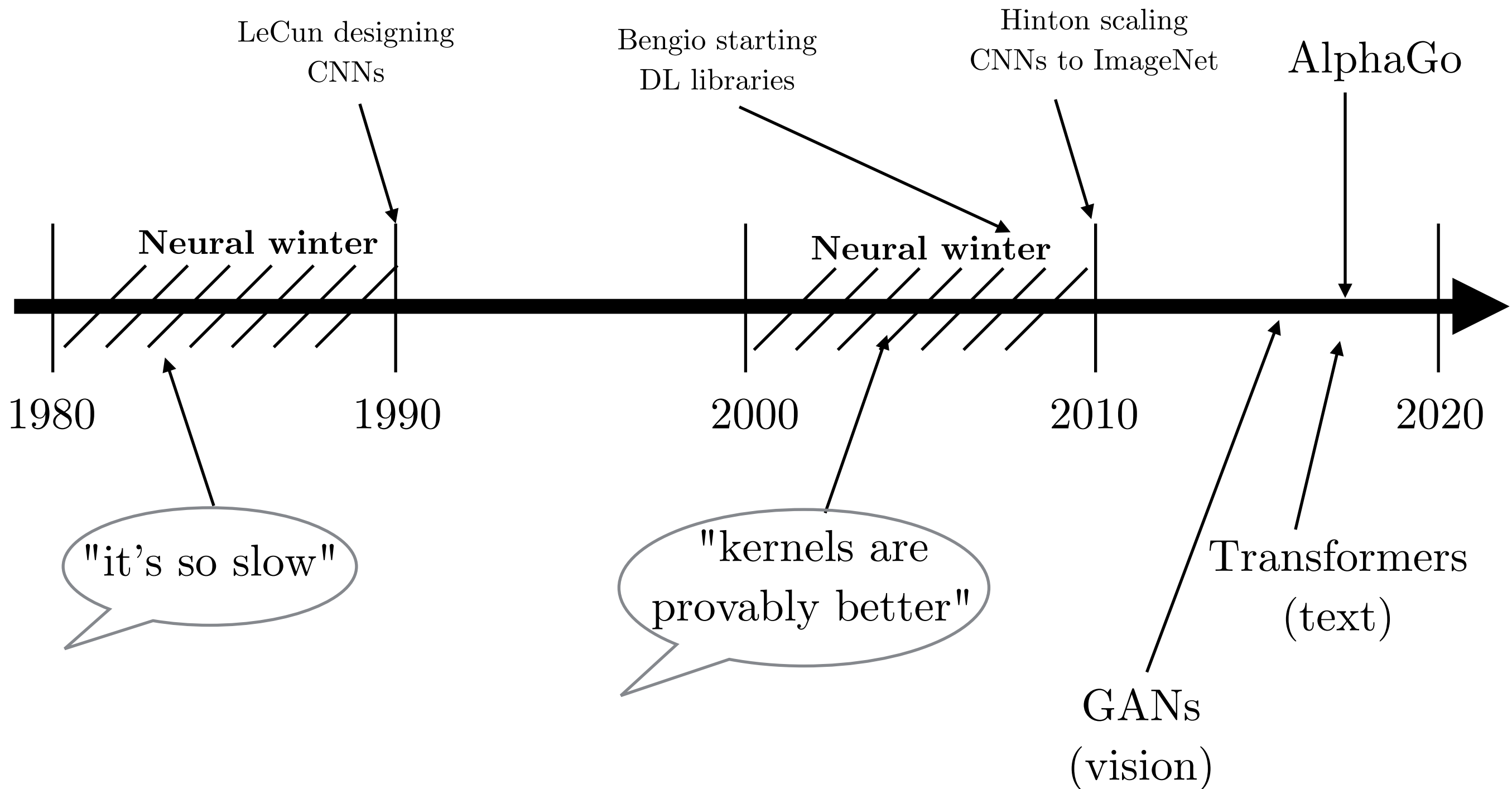


Solving high-dimensional tasks with deep learning

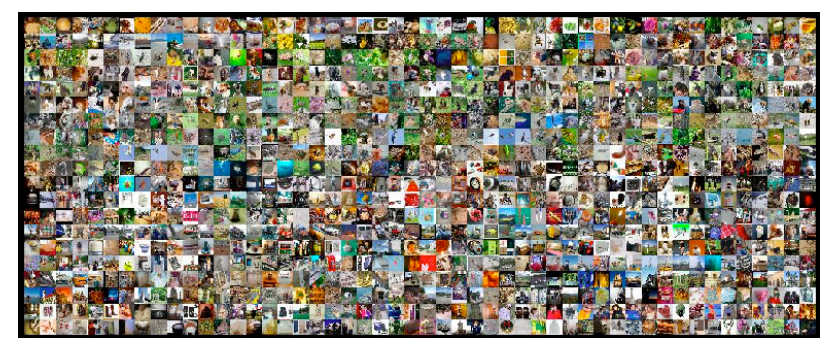
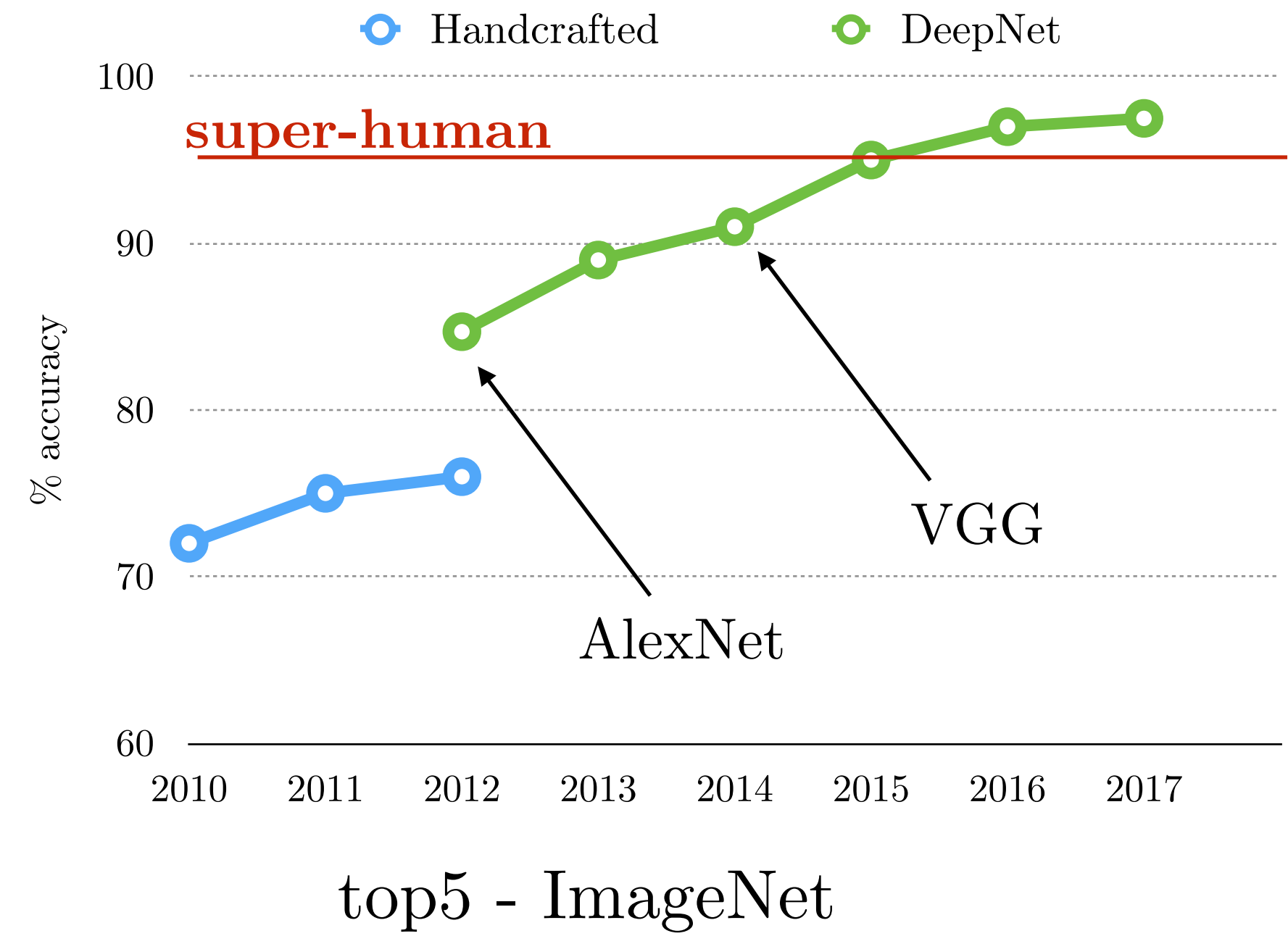
Deep Learning, 2015, Nature, LeCun, Bengio, Hinton

- *Solve* several high dimensional problems that seemed intractable. Impressive benchmarks.
- Requires a *huge* amount of labeled data
- *Generic* and *simple* to deploy (present in many final products) / requires a *large* expertise (highly demanded profiles)
- Handcrafted features are *not required*: the algorithm adapts itself to the specific bias of a task

A biased history of Deep Learning

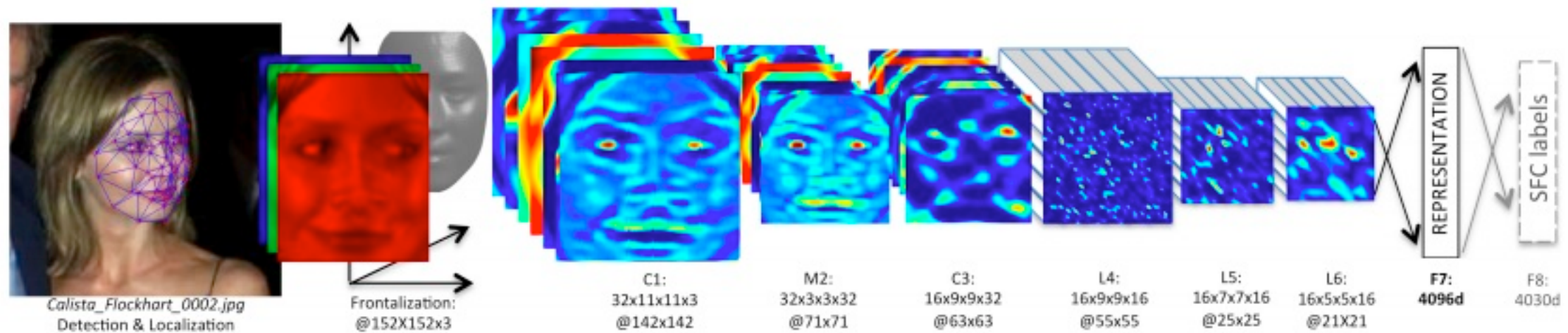


- Accuracies! (you probably heard about it during Prof. Banerjee's talk)

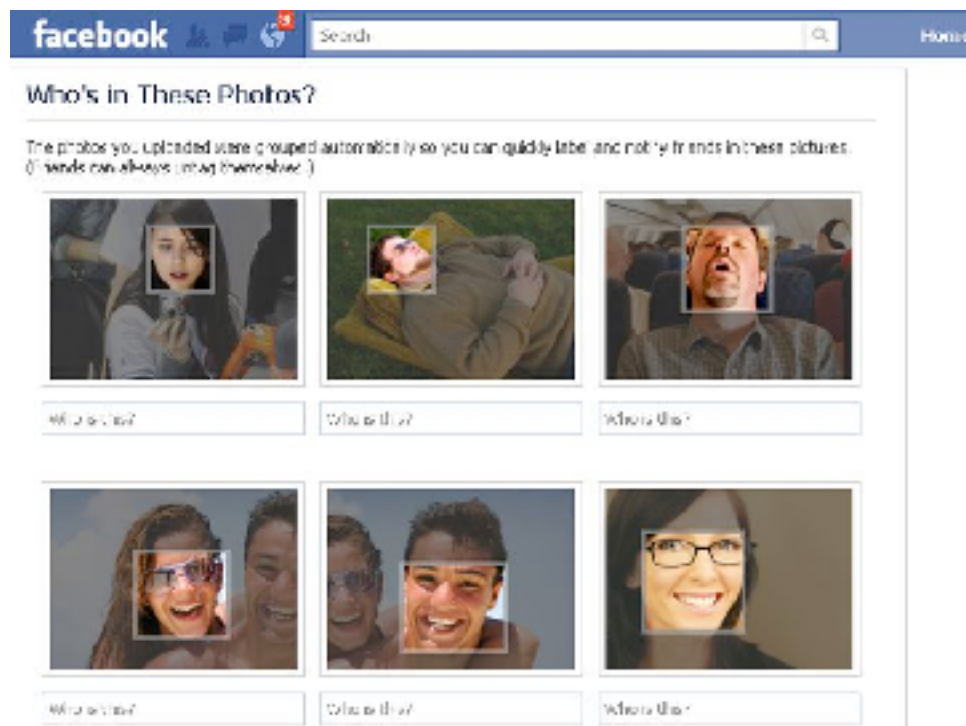


ImageNet:
 1 million training images, 1 000 classes
 400 000 test images
 Large coloured images of various sizes

Theory for good performances?



Are two pictures corresponding to the same person?
Above human performances in rough conditions

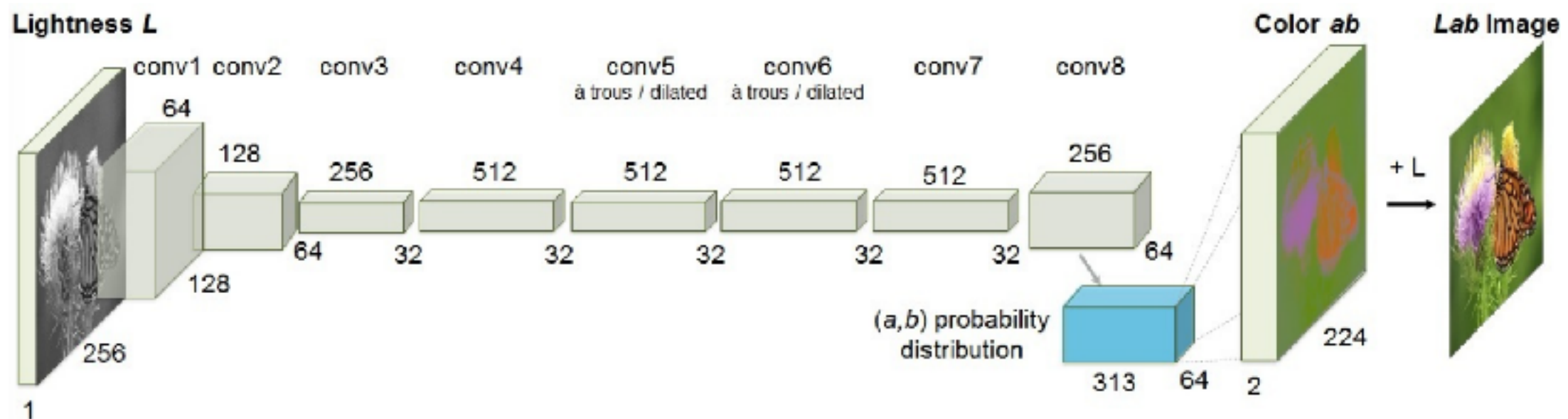


Ref.: DeepFace: Closing the Gap to Human-Level Performance in Face Verification
Taigman et al.





Colorful Image Colorization, Zhang et al.



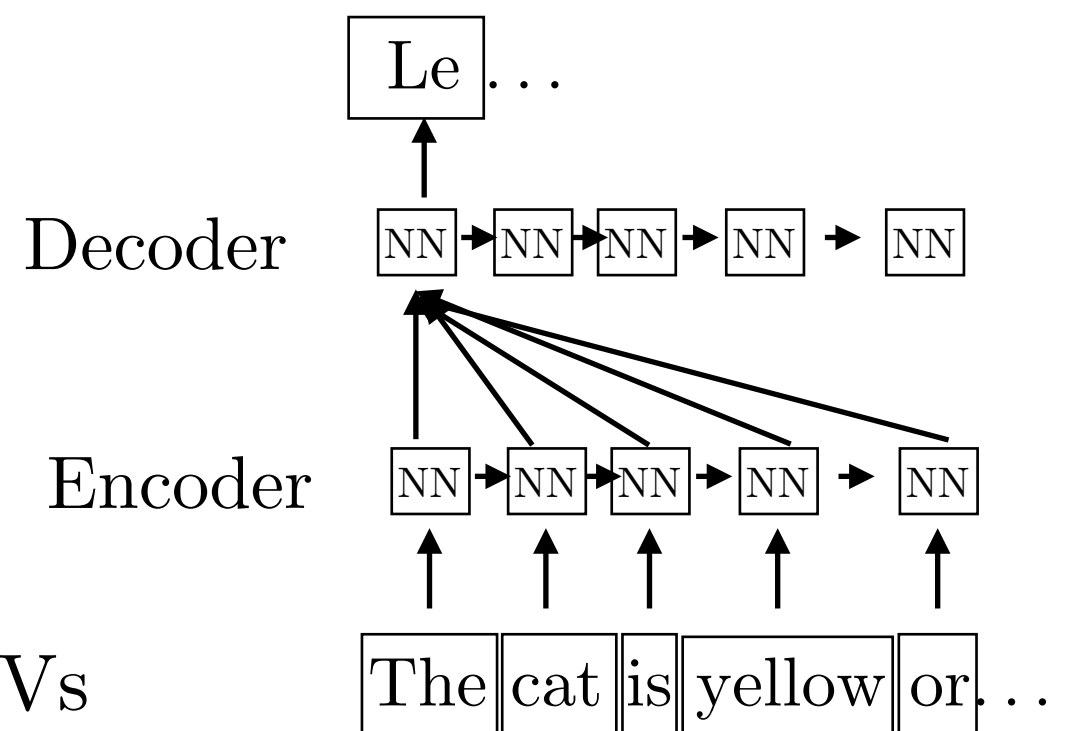
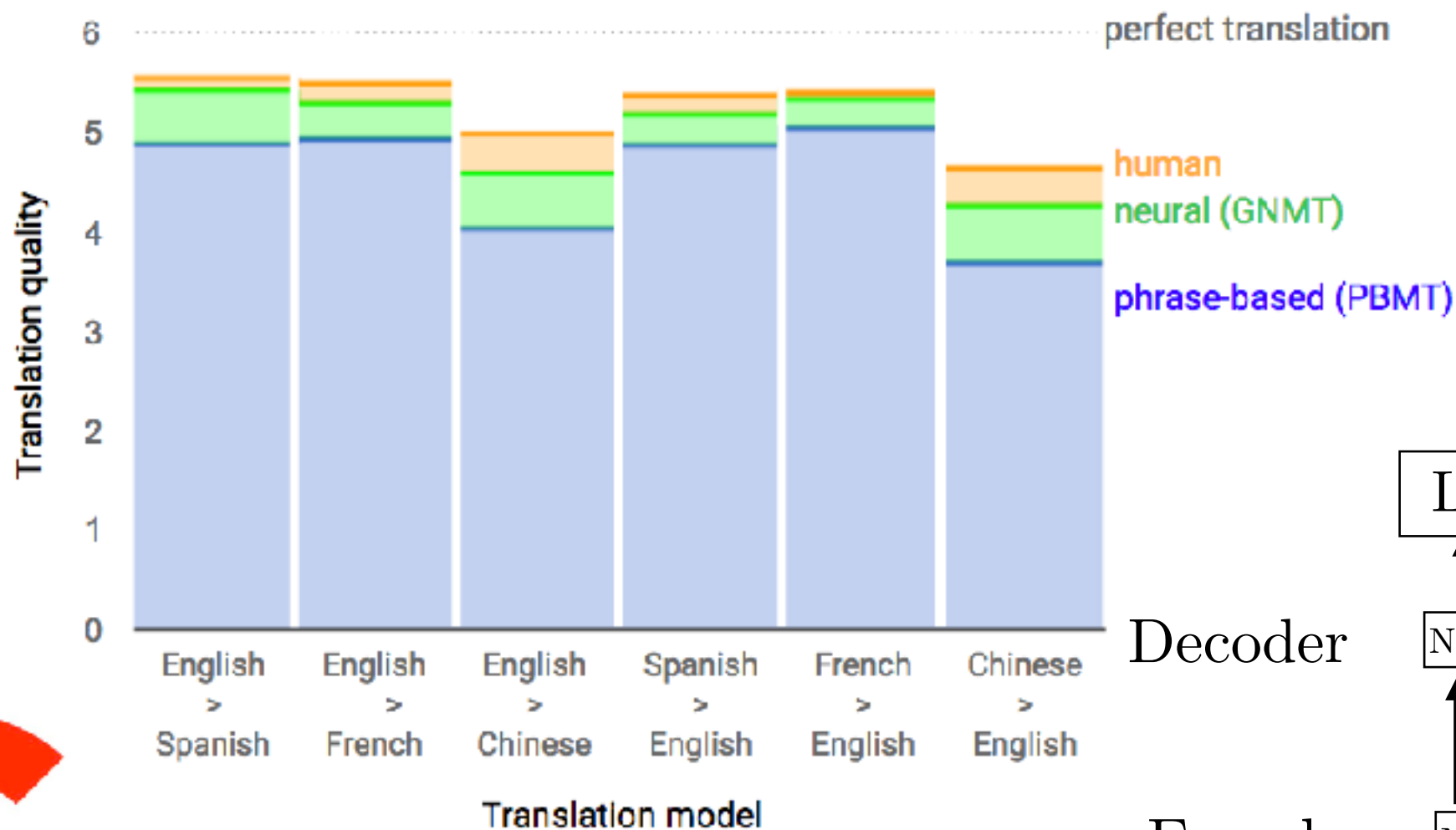
Coloring an image by hand takes several weeks



Spectacular results in face generation.

in text understanding/translations

- Translation (Google uses Recurrent Neural Networks):



Applications for HR: sorting CVs

Surprising results in text, image & (source) code generation

- Generating source code via Recurrent Neural Networks:

<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

```
#define REG_PG      vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)      (func)

#define SWAP_ALLOCATE(nr)      (e)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %%esp, %0, %3" : : "r" (0)); \
    if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
    pC>[1]);

static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
    PUT_PARAM_RAID(2, sel) = get_state_state();
    set_pid_sum((unsigned long)state, current_state_str(),
        (unsigned long)-1->lr_full; low;
}
```

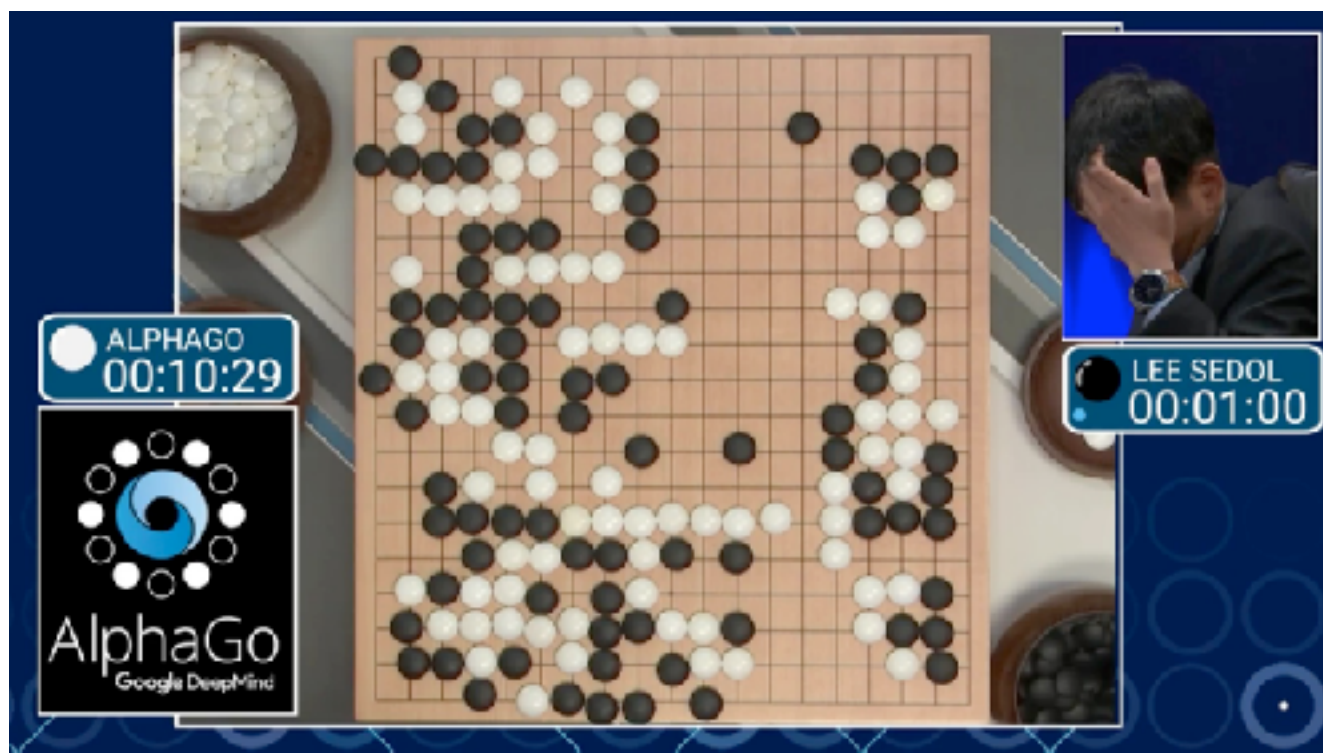
Real one?

Outstanding results with ²⁴

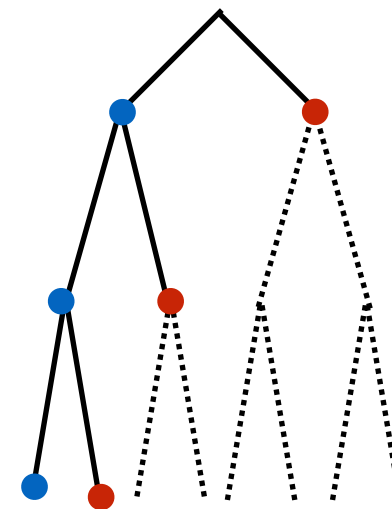
Game Strategy

Game of GO: completely impossible to solve with pure Monte Carlo tree search

Ref.: Mastering the Game of Go with Deep Neural Networks and Tree Search
Silver et al.



● N Roll out with
● Y NNs



NN: computes a proba to win
for each of the 2^{196} nodes

Self driving cars, Starcraft...



Outstanding results in Style Transfer

$$\arg \min_{\tilde{x}} \underbrace{\|\Phi x - \Phi \tilde{x}\|^2}_{\text{Original image}} + \lambda \underbrace{\|\text{Cov}(\Phi y) - \text{Cov}(\Phi \tilde{x})\|^2}_{\text{Style transfer}}$$

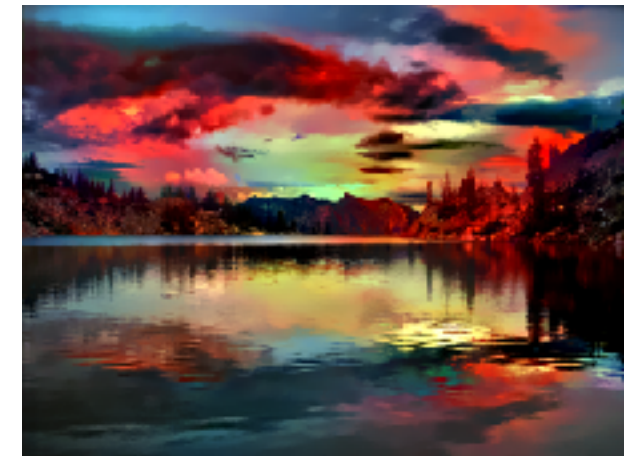
Input
 Φx



Target style
 Φy

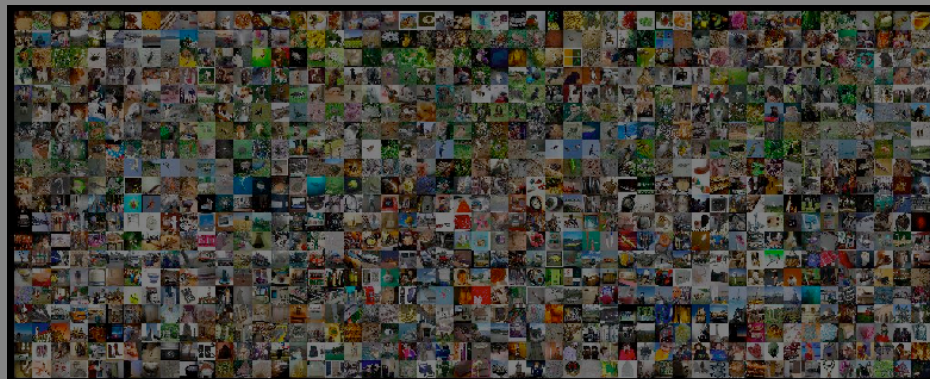


Output
 $\Phi \tilde{x}$

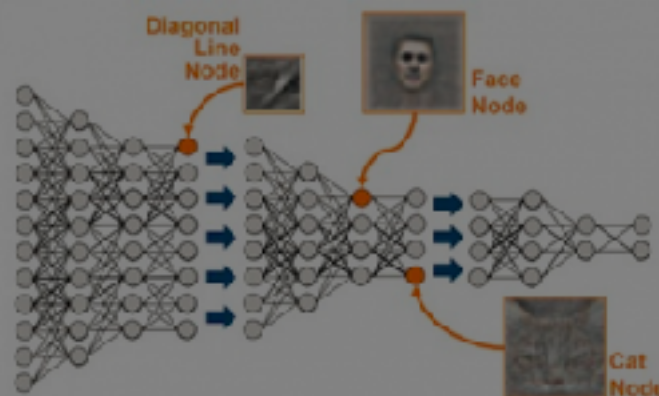
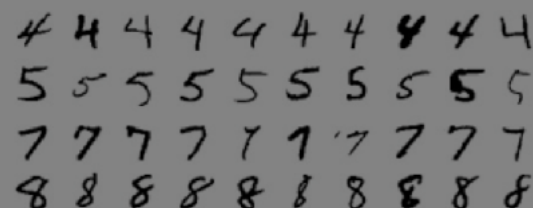


Ref.: Deep Photo Style Transfer, Luan et al.

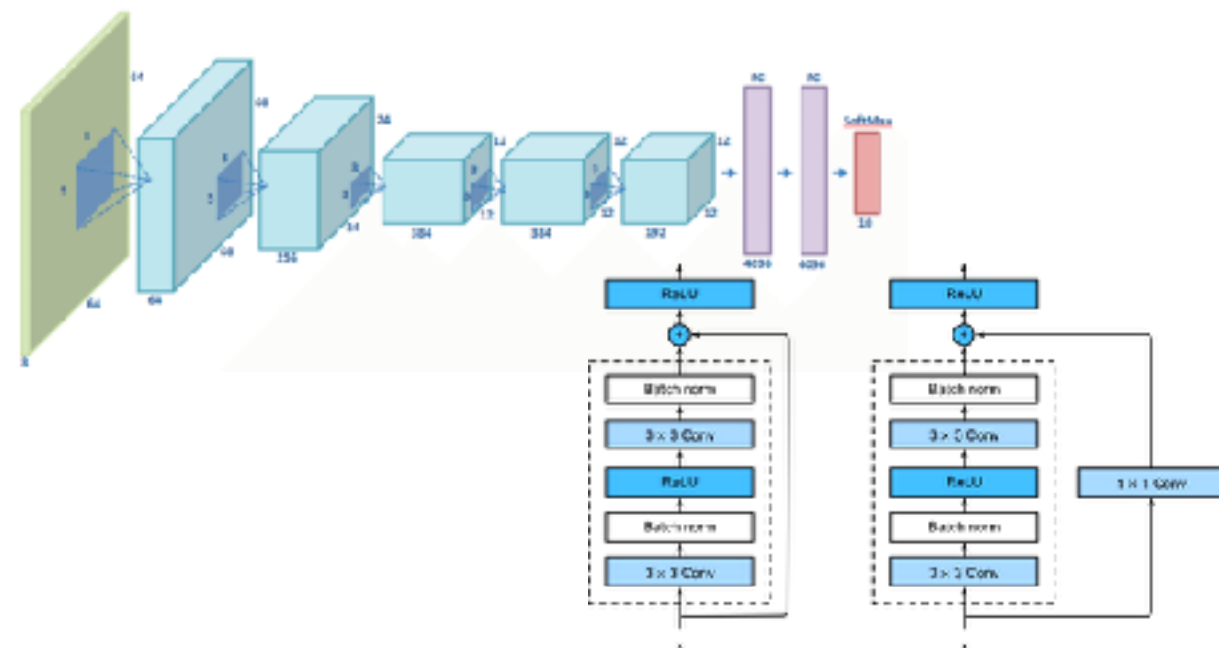
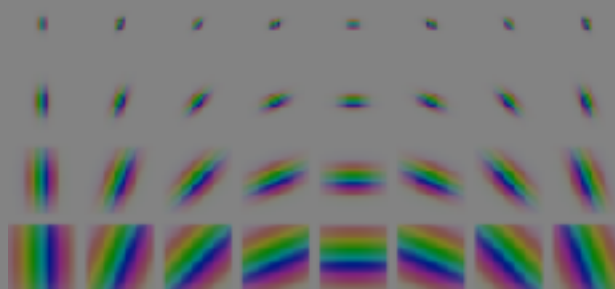
Direct applications in Web design...



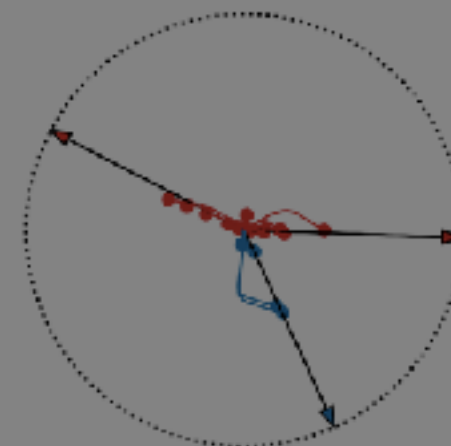
Introduction to high-dimensional tasks



Understanding Convolutional Neural Networks



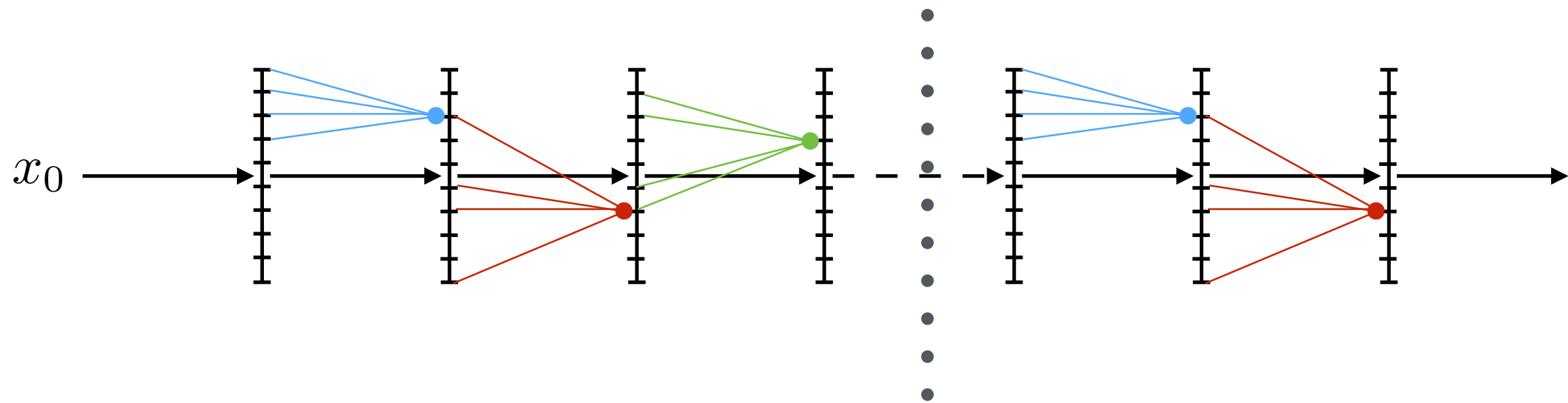
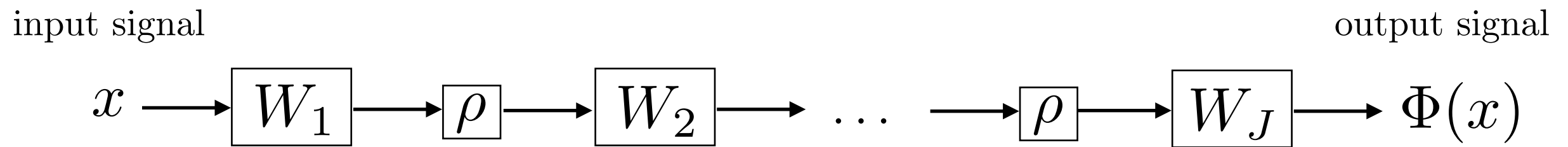
Engineering Deep Neural Networks



Under the Hood of Neural Networks

$$C_f = \int_{\mathbb{R}^D} \|\omega\|_1 |\hat{f}(\omega)| d\omega$$

Neural Networks



$$x_{j+1} = \rho W x_j \longrightarrow x_{j+1,k} = \rho \left(\sum_i W_{j,ki} x_{j,i} \right)$$

No *a priori* is introduced here. Typically used as a classifier.

Note that $\Phi(x; W_1, \dots, W_J)$ is non-convex in x or each W_j

where: $\rho(x) = \max(0, x)$ s.t. $|\rho(x) - \rho(y)| \leq |x - y|$

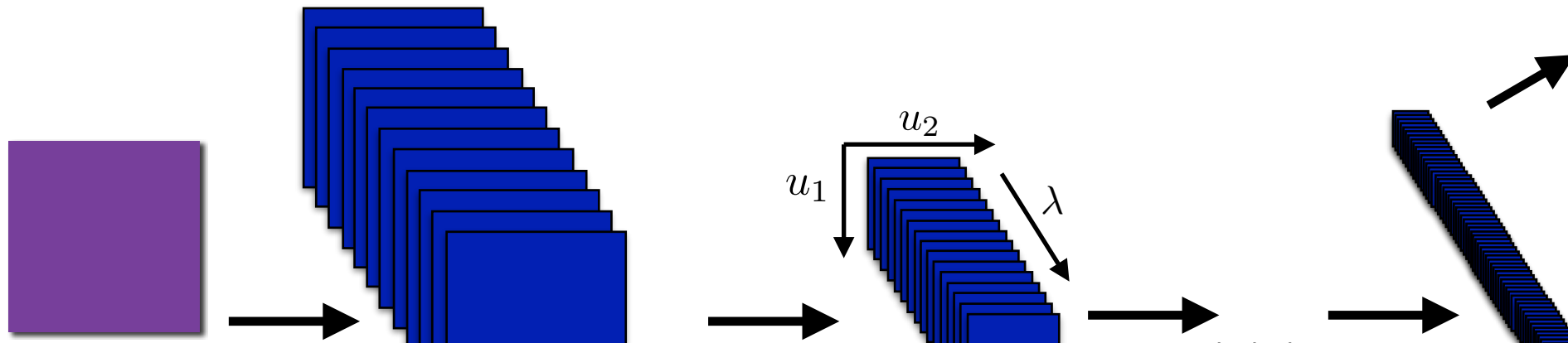
Convolutional Neural Networks

input signal

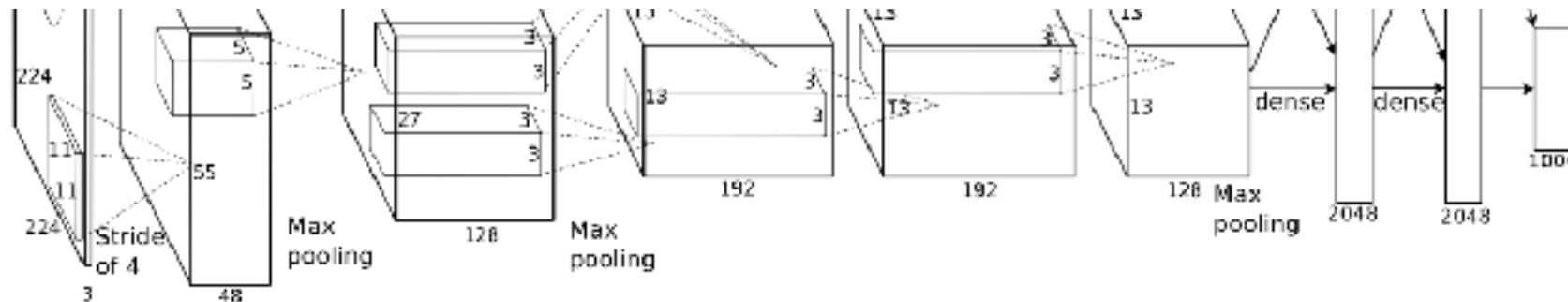
output signal

$$x \rightarrow W_1 \rightarrow \rho \rightarrow W_2 \rightarrow \dots \rightarrow \rho \rightarrow W_J \rightarrow \Phi(x)$$

Schematic



Engineering



Each layer: $x_{j+1} = \rho W_j x_j$

that leads to:
$$x_{j+1}(u, \lambda_{j+1}) = \rho \left(\sum_{\lambda_j} \left(x_j(., \lambda_j) \star w_{\lambda_j, \lambda_{j+1}} \right) (u) \right)$$

learned kernel

Sometimes some "pooling" are incorporated, mainly for speed purposes.

Again, this leads to a non convex loss.

Automatic Differentiation

- In a typical supervised task, one aims at minimizing:

$$L(\Theta) = \frac{1}{n} \sum_{i \leq n} \ell(\Phi_{\Theta}(x_i), y_i) + \frac{\lambda}{2} \|\Theta\|^2 \quad \Theta = (\theta_1, \dots, \theta_J)$$

parameters of each layer \nearrow

\nwarrow ℓ^2 regularisation or weight decay

- For instance, for obtaining a prediction, one minimizes the Cross-Entropy:

$$\ell(\Phi(x), y) = -\Phi(x)[y] + \log \left(\sum_j \exp(\Phi(x)[j]) \right)$$

where:

$$\Phi(x) \in \mathbb{R}^C \quad \text{number of classes} \quad y \in \{1, \dots, C\}$$

At prediction time, one picks:

$$\hat{y} = \arg \max_c \Phi(x)[c]$$

- Typically done via Stochastic Gradient Descent (Prof. Dieuleveut's talk), where the gradient is given by:

$$g(\Theta) = \frac{1}{\mathcal{B}} \sum_{b=1}^{\mathcal{B}} \nabla \ell(\Theta, x_{N^b})$$

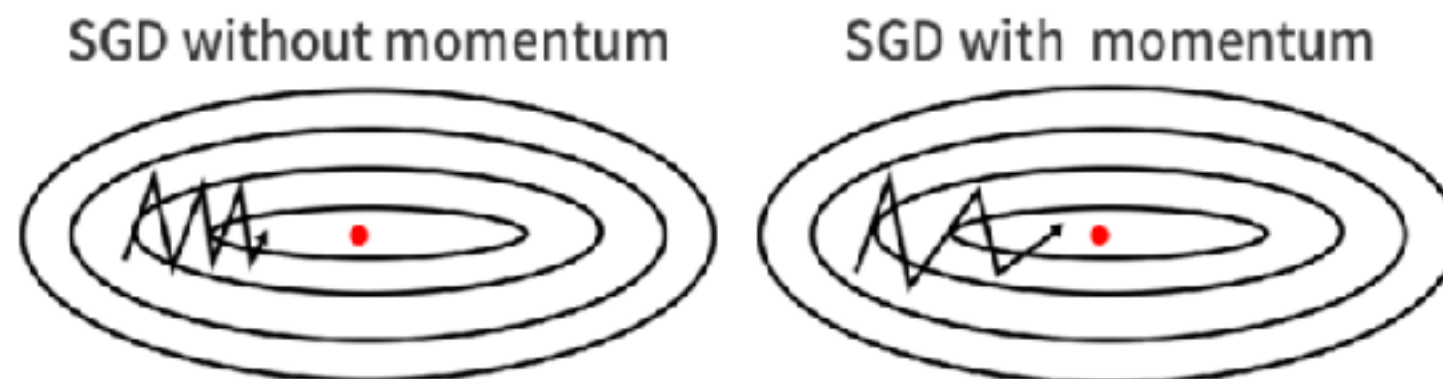
with N^b uniformly sampled from $\{1, \dots, n\}$

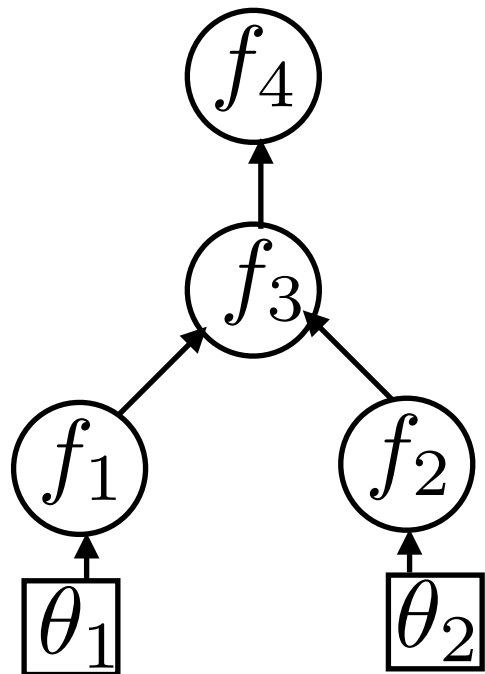
leading to: $\mathbb{E}[g(\Theta)] = \nabla L(\Theta)$

- And one iterates from a random initialisation Θ_0 :

$$\Theta_{t+1} = \Theta_t - \eta_t g(\Theta_t)$$

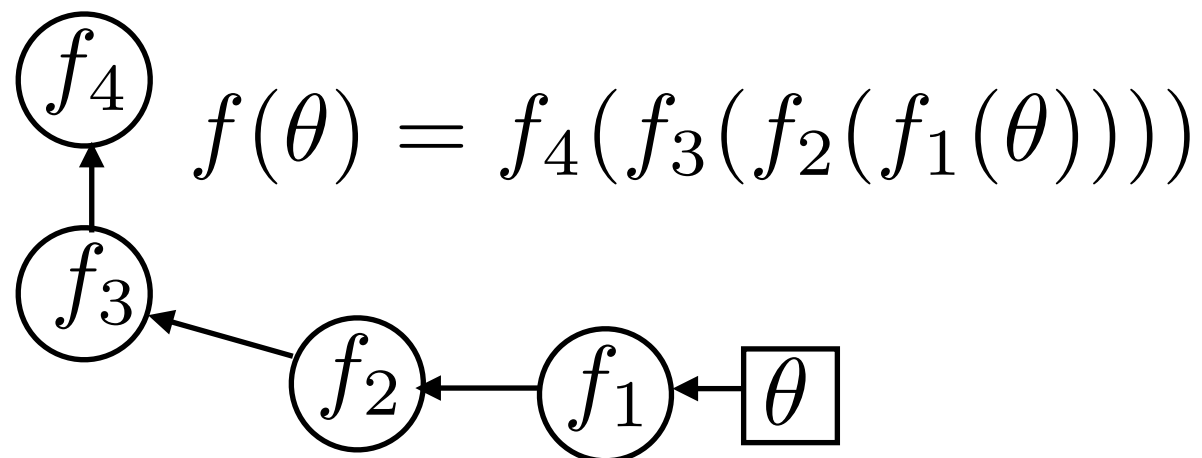
- Many variants of this optimization: Momentum, ADAM, ...





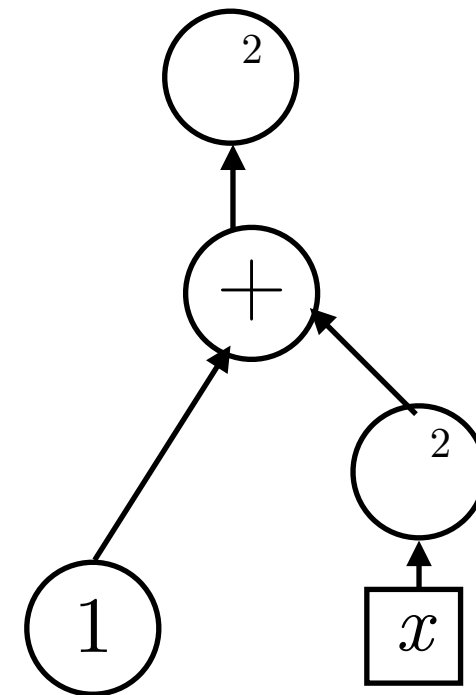
$$f(\Theta) = f_4(f_3(f_1(\theta_1), f_2(\theta_2)))$$

$$\Theta = (\theta_1, \theta_2)$$



$$f(\theta) = f_4(f_3(f_2(f_1(\theta))))$$

$$f(x) = (1 + x^2)^2$$



How to compute fast gradients?

$f_i, \partial f_i$ computed in $\mathcal{O}(??)$

- No *a priori* on the operations involved: finite scheme difference

- For a gradient $f'(\theta) = \frac{f(\theta + \delta\theta) - f(\theta)}{\delta\theta} + o(1)$
Yet: can be unstable and requires multiple calls to f

- Back-propagation algorithm for a tree:

Ref.: Gabriel Peyré's slides at the Mathematical
Morning coffee
Mathieu's Blondel teaching about AD

$$(f \circ g)(x) \in \mathbb{R} \text{ leads to } \nabla(f \circ g)(x) = [\partial g](x)^T \nabla f(g(x))$$

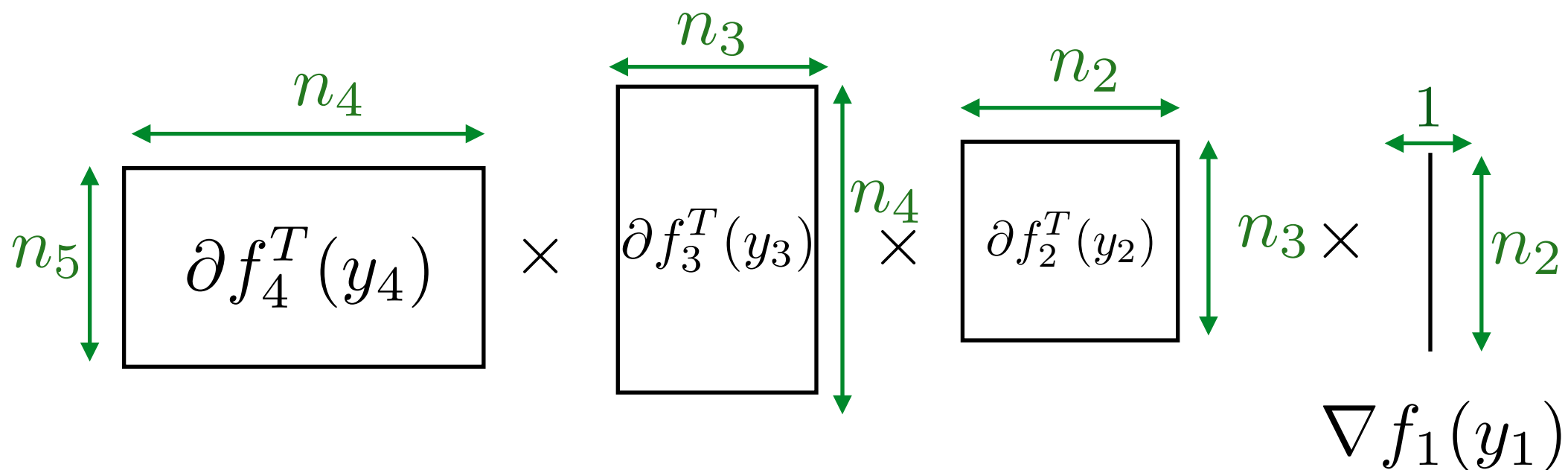
$$(f_1 \circ f_2 \circ f_3)(x) \in \mathbb{R} \text{ leads to}$$

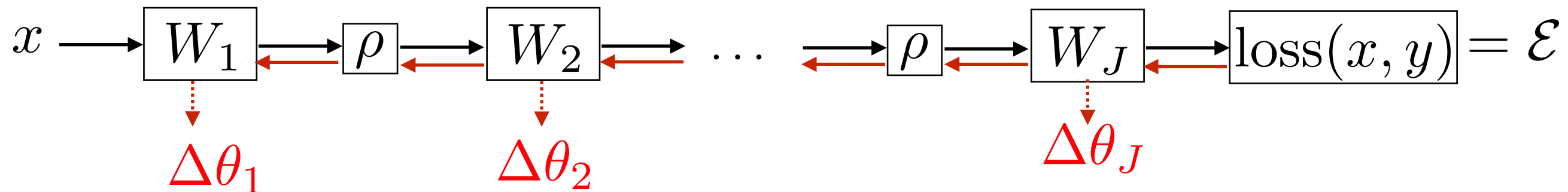
$$\nabla(f_1 \circ f_2 \circ f_3)(x) = \underbrace{[\partial f_3](x)}^\top \underbrace{[\partial f_2](f_1(x))}^\top \underbrace{\nabla f_1(f_1 \circ f_2(x))}^\top$$

How to compute this quantity efficiently?

Fast-computations

- Assuming that: $f_i : \mathbb{R}^{n_i} \rightarrow \mathbb{R}^{n_{i+1}}$
- Computing each Jacobian is about $\mathcal{O}(n_i n_{i+1})$
- Multiplying a matrix $m \times n$ with a $n \times p$ costs $\mathcal{O}(mnp)$
- It also induces a substantial memory saving...



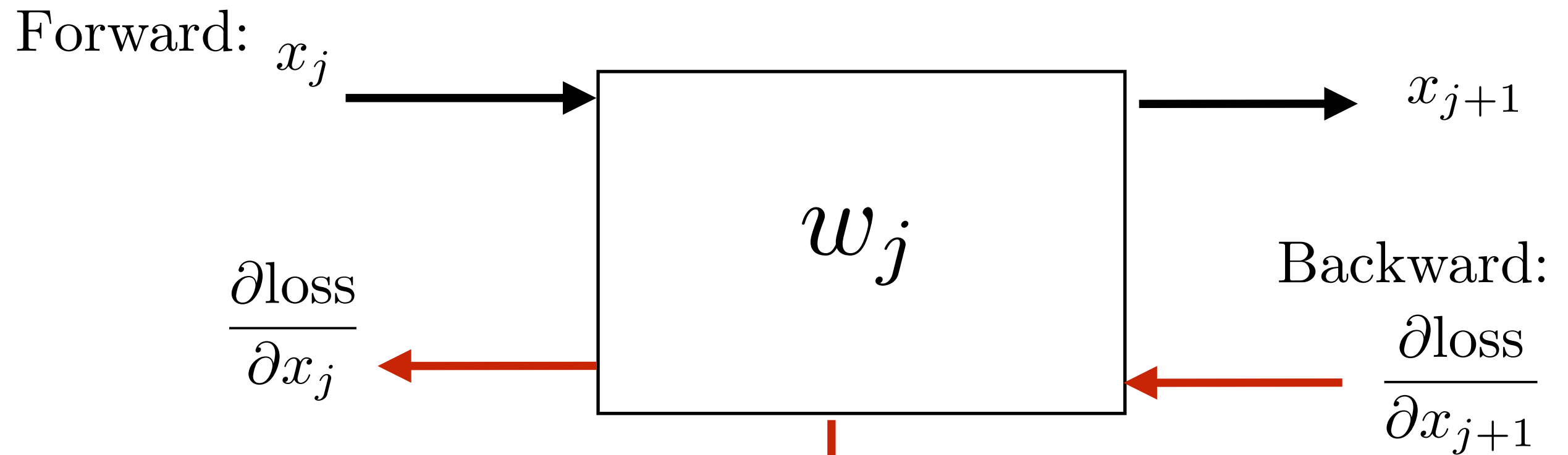


$$\leftarrow \nabla_{x_j}(\mathcal{E}) = \frac{\partial(\rho W_j)^T}{\partial x_j} \nabla_{x_{j+1}}(\mathcal{E}) \quad \downarrow \quad \nabla_{\theta_j}(\mathcal{E}) = \frac{\partial(\rho W_j)^T}{\partial \theta_j} \nabla_{x_{j+1}}(\mathcal{E})$$

- Automatic-differentiation fits well Deep Learning!
- Note the lock that can make distributed optimization difficult

Ref.: Decoupled Neural Interfaces using Synthetic Gradients, Jaderberg et al, 2017

- How is it implemented?



$$\frac{\partial \text{loss}}{\partial w_j} = \frac{\partial \text{loss}}{\partial x_j} \frac{\partial x_j}{\partial w_j}$$

$\frac{\partial \text{loss}}{\partial w_j}$ Gradient w.r.t. parameters

$$\frac{\partial \text{loss}}{\partial x_j} = \frac{\partial x_{j+1}}{\partial x_j} \frac{\partial \text{loss}}{\partial x_{j+1}}$$



(GPUs were for video-games which require fast graphic rendering!)

- Deep learning algorithms rely a lot on **linear** operations. (ex: convolutions)
- CUDA routines permit to implement efficiently linear algebra routines: min. speed up of **100**.
- GPUs are now super mainstream...
- It's not unusual to have a 1TB GPU memory.



Training Pipeline

- Once the model $\Phi(x; \theta)$ and the loss ℓ is fixed the model is trained via mini-batch:

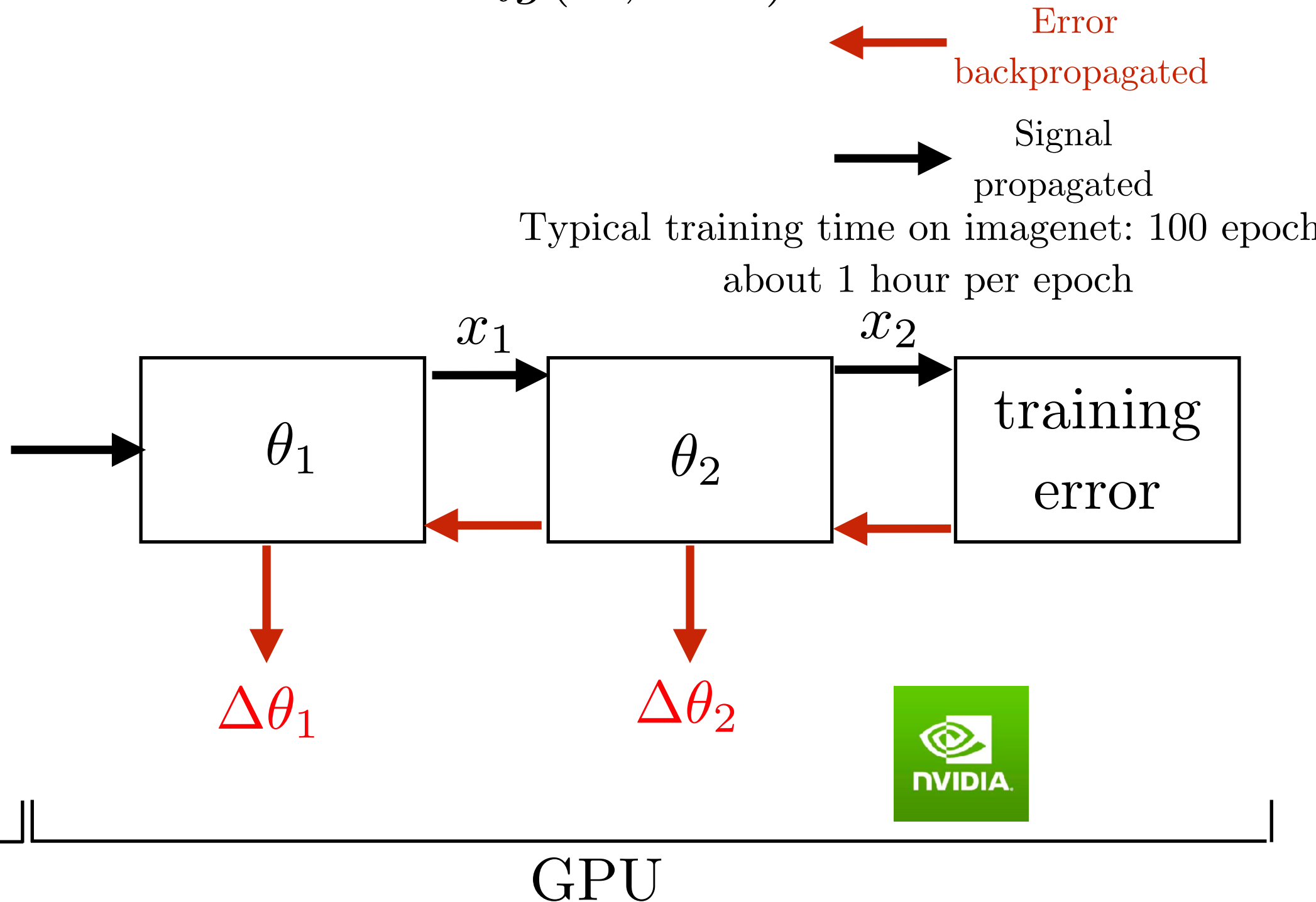
Splitting dataset
into batches of size

$$\theta^{t+1} = \theta^t - \alpha_t g(\theta^t, \text{data})$$

256

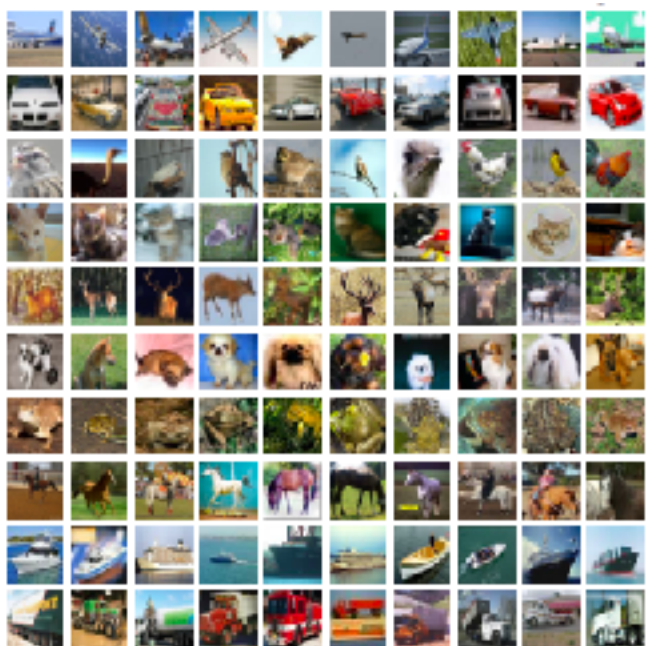
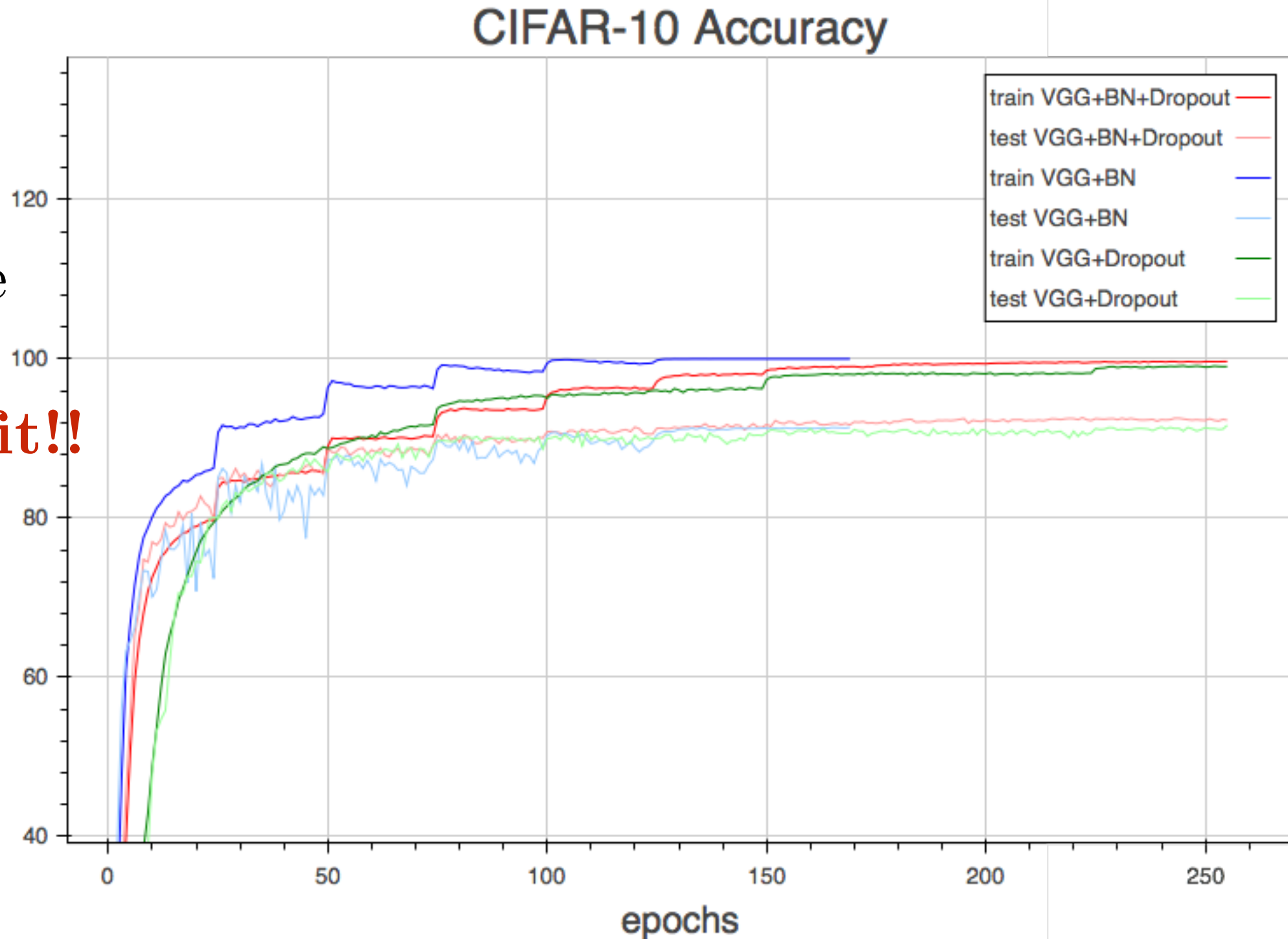


CPU



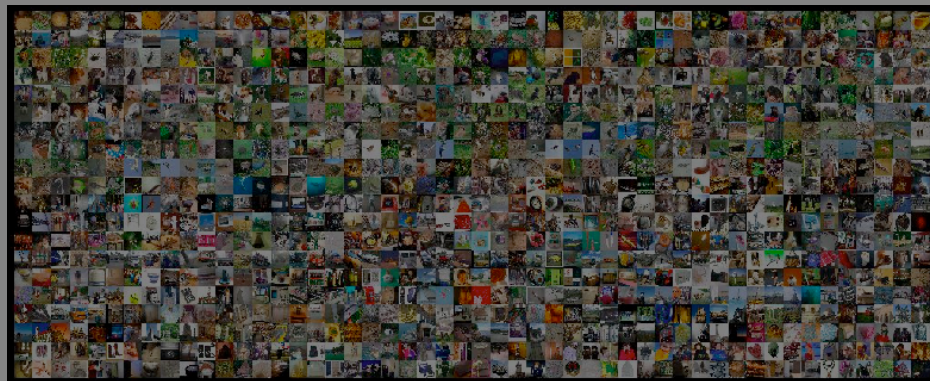


- Batch-normalization
- Data augmentation
- Dropout
- Learning rate
- **Let it overfit!!**

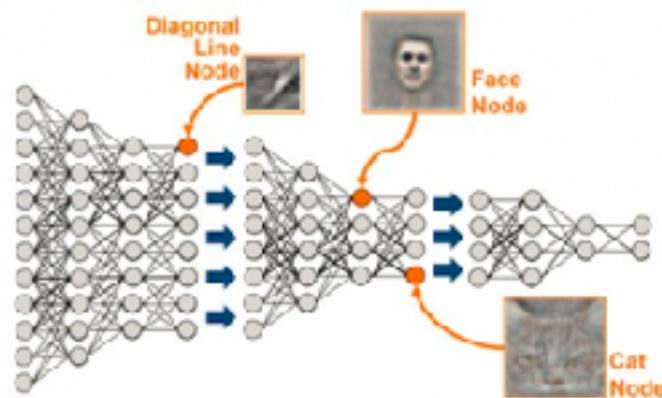
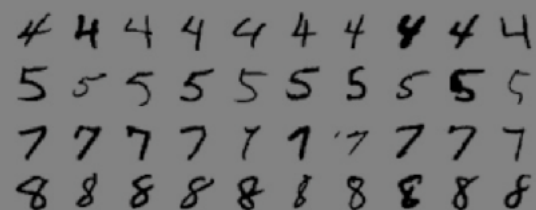


The strength of DL

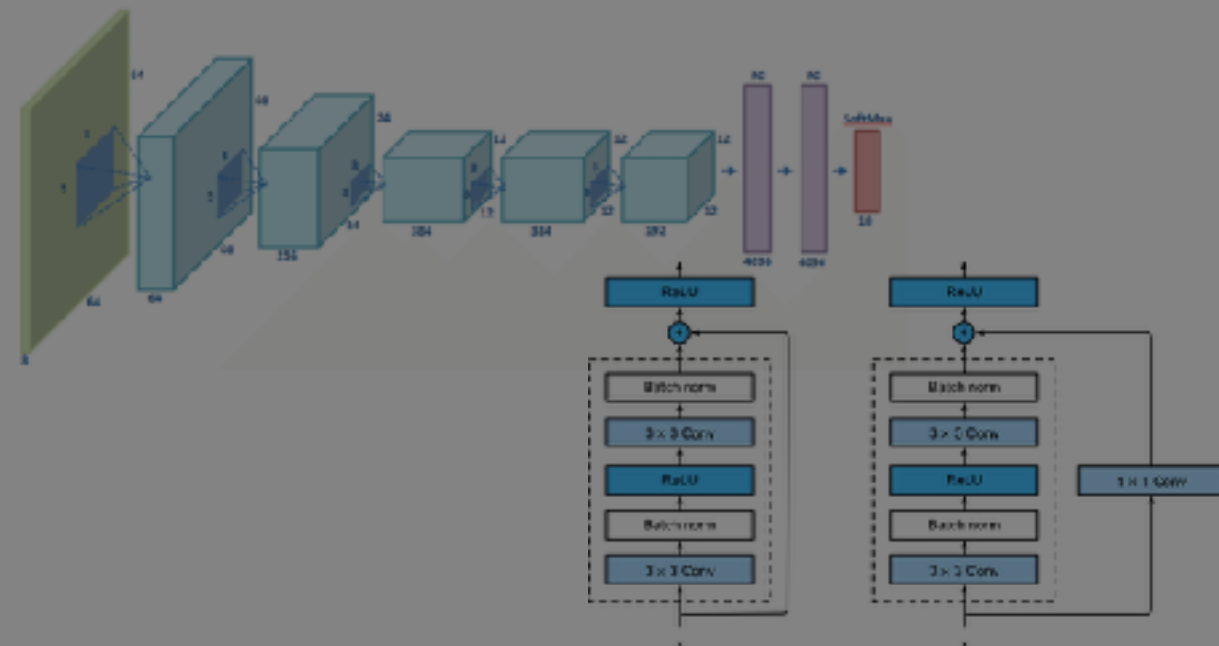
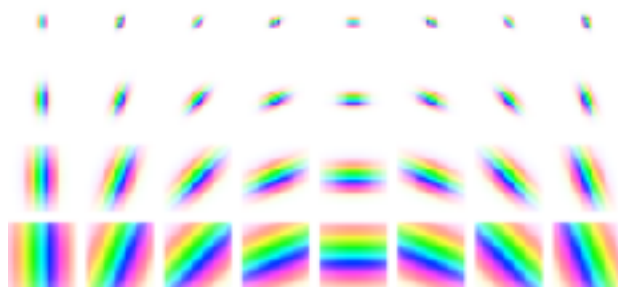
- Data? Computer power? Not only:
- **Flexibility&modularity**: quickly benchmarking non-linearity, layer dimension, losses, batch size, learning rate schedule...
- Is it overfitting? Clearly, yet the representations learned are empirically excellent and used in many cryptic applications.



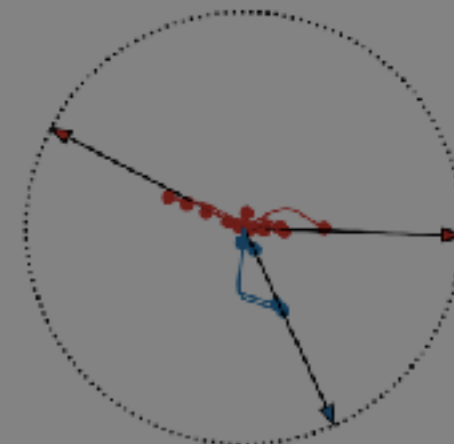
Introduction to
high-dimensional tasks



Understanding Convolutional
Neural Networks



Engineering Deep Neural Networks



Under the Hood of Neural
Networks

$$C_f = \int_{\mathbb{R}^D} \|\omega\|_1 |\hat{f}(\omega)| d\omega$$

Convolutional layers in CNNs

$$(a \star b)(u) = \int a(u - v)b(v) dv$$

Let a linear operator: $W : \ell^2(\{1, \dots, n\}^k)^{d_1} \rightarrow \ell^2(\{1, \dots, n\}^k)^{d_2}$

number of channels

$$L_1 : \ell^2(\{1, \dots, n\}^k)^{d_1} \rightarrow \ell^2(\{1, \dots, n\}^k)^{d_1}$$

$$L_2 : \ell^2(\{1, \dots, n\}^k)^{d_2} \rightarrow \ell^2(\{1, \dots, n\}^k)^{d_2}$$

size of an image

$$\text{where } L_i x[u, j] = x[u + 1, j]$$

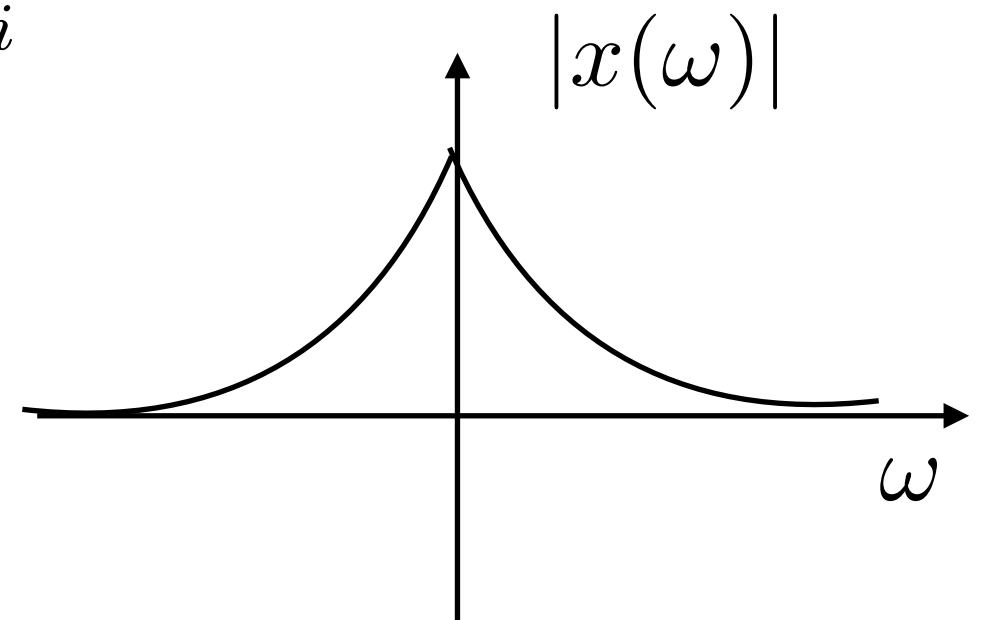
translation operator

- Lemma:**

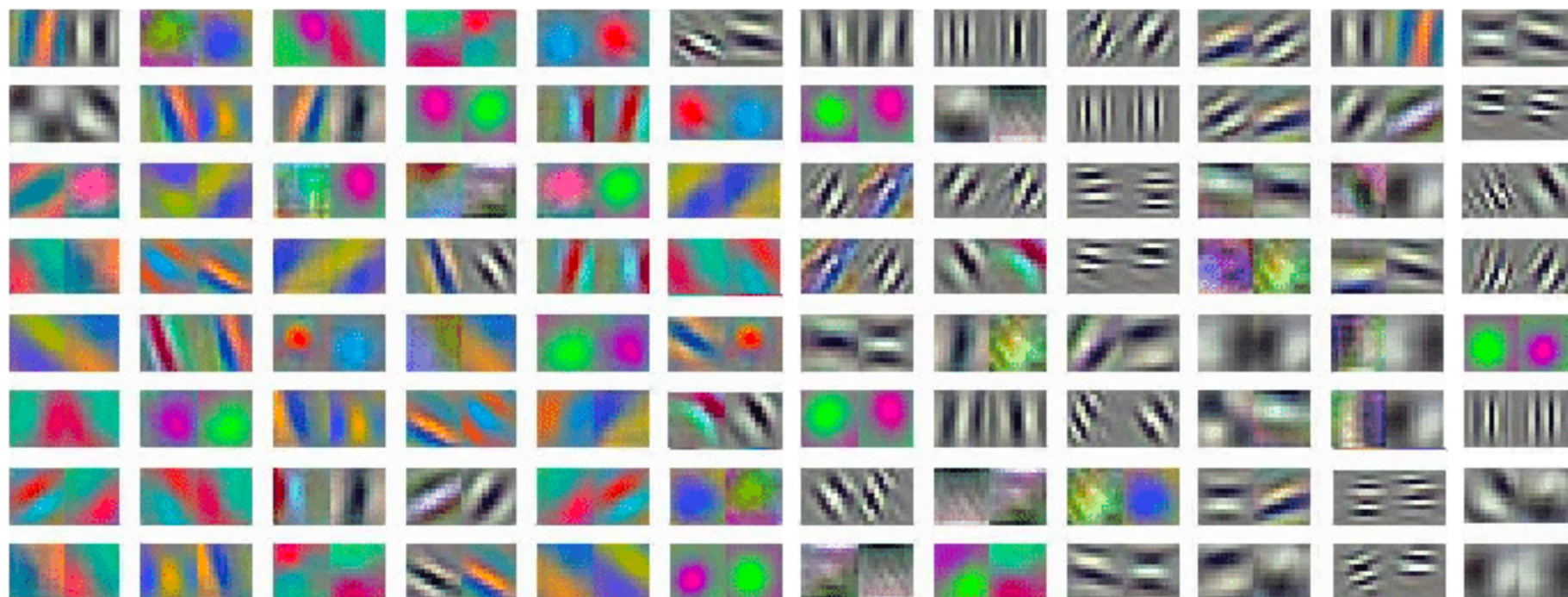
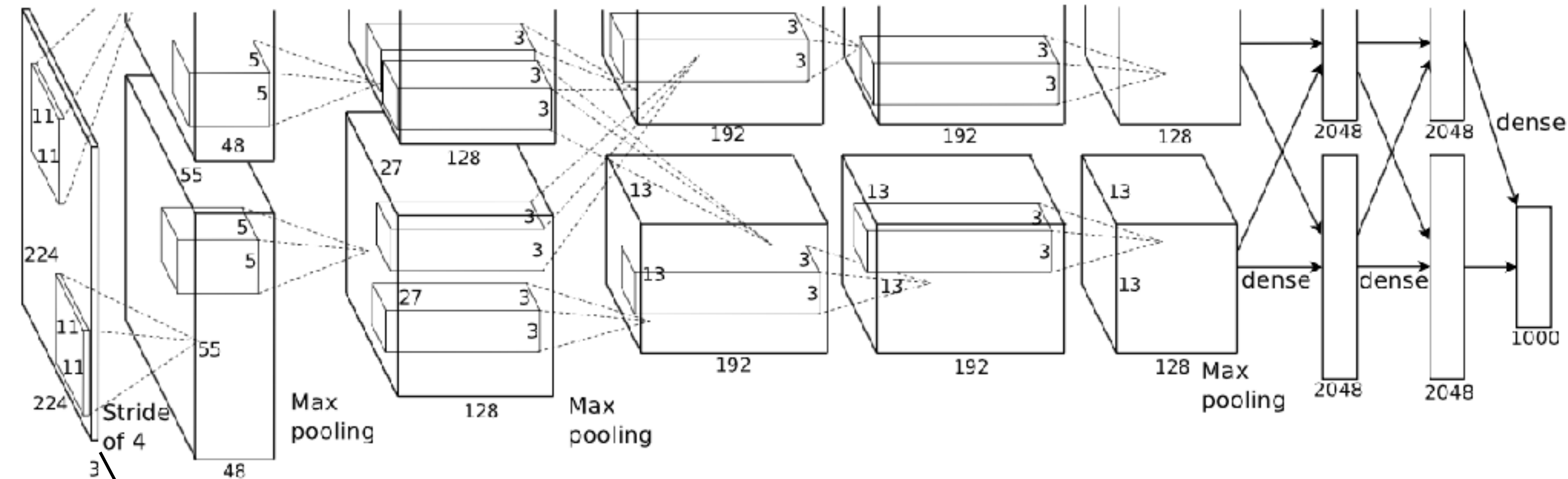
$$WL_1 = L_2W \iff Wx[u, j] = \sum_i (k_{i,j} \star^u x)[u, i]$$

- Fourier analysis!!

$$\widehat{x \star y}(\omega) = \hat{x}(\omega)\hat{y}(\omega) \longrightarrow$$

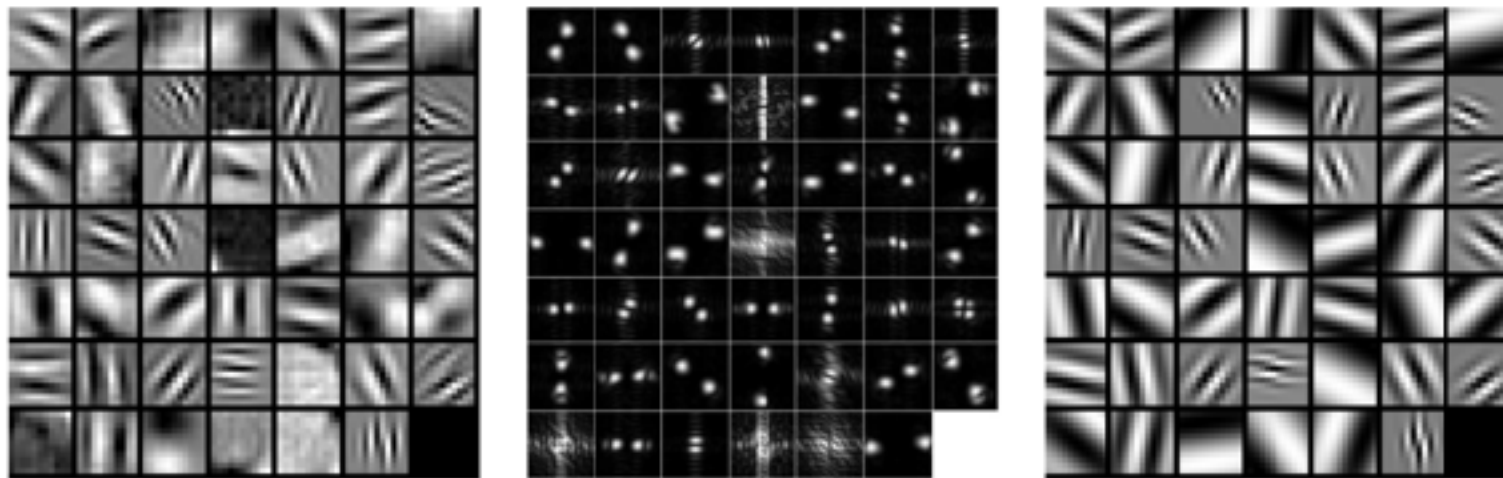


Regularity corresponds to a fast decay



A short analysis of an AlexNet

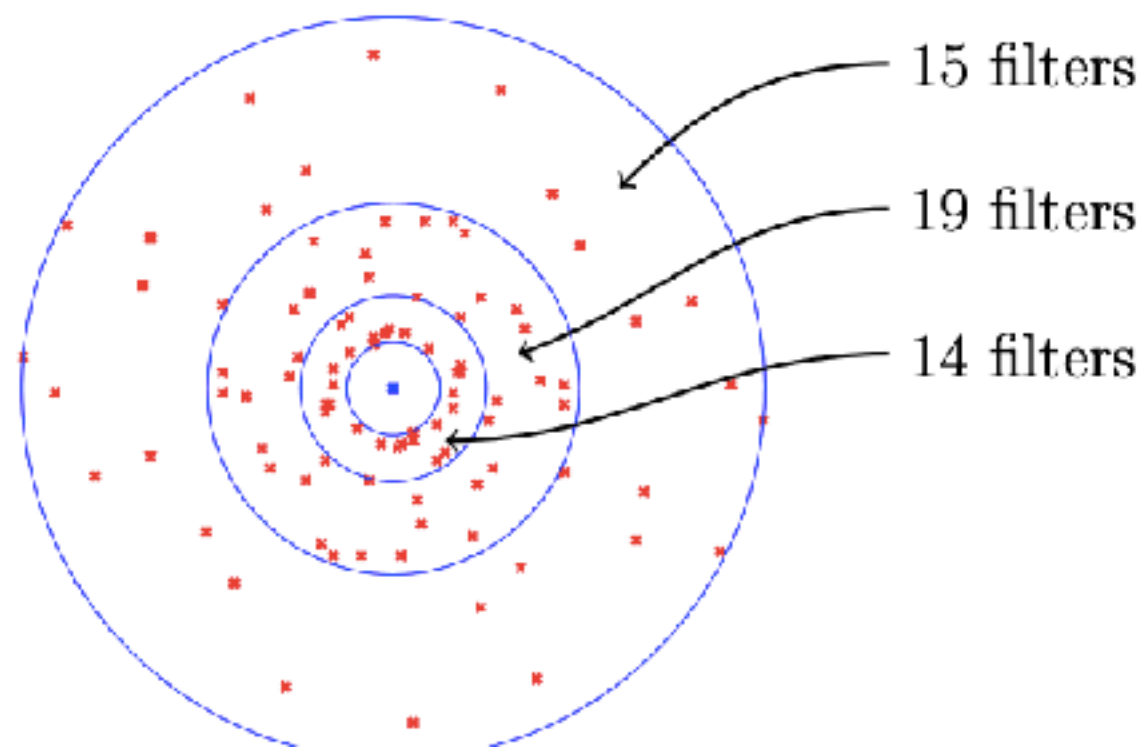
$$\psi_{C,D,\xi}(u) = C e^{-u^T D u} e^{i u^T \xi}$$



Ref.: I Waldspurger's phd

- Consider Gabor filters and fit the model.


This principle is core
in many models
(V1, Scattering, ...)

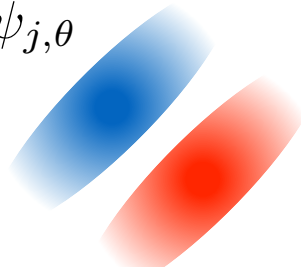


- ψ is a wavelet iff $\int \psi(u)du = 0$ and $\int |\psi|^2(u)du < \infty$
- Typically localised in space and frequency.

- Rotation, dilation of a wavelets:

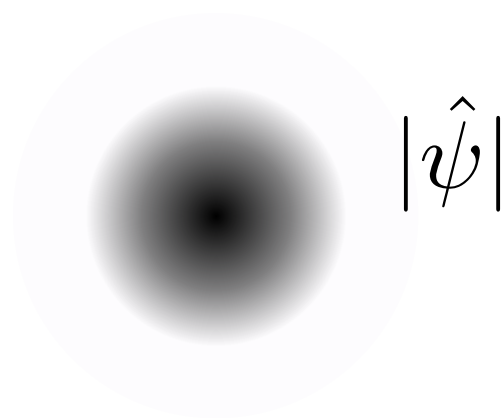
$$\psi_{j,\theta} = \frac{1}{2^{2j}} \psi\left(\frac{x_\theta(u)}{2^j}\right)$$

ψ


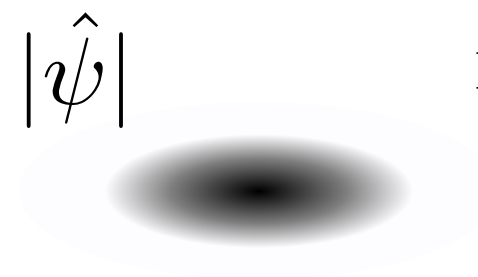
$\psi_{j,\theta}$


Group action!
- Design wavelets selective to **rotation** variabilities.

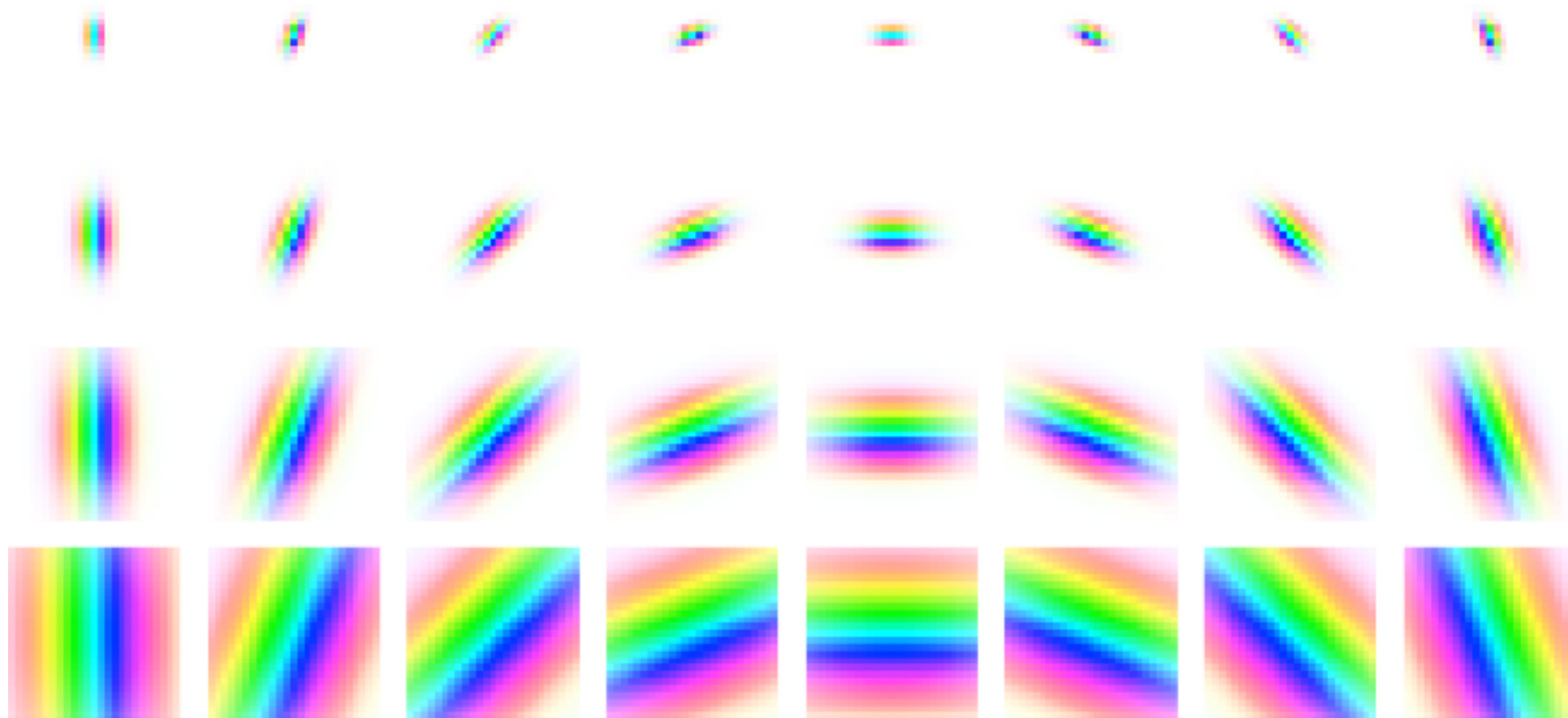
Isotropic



VS



Non-Isotropic



$$\psi(u) = \frac{1}{2\pi\sigma} e^{-\frac{\|u\|^2}{2\sigma}} (e^{i\xi \cdot u} - \kappa)$$

$$\phi(u) = \frac{1}{2\pi\sigma} e^{-\frac{\|u\|^2}{2\sigma}}$$

The Gabor wavelet

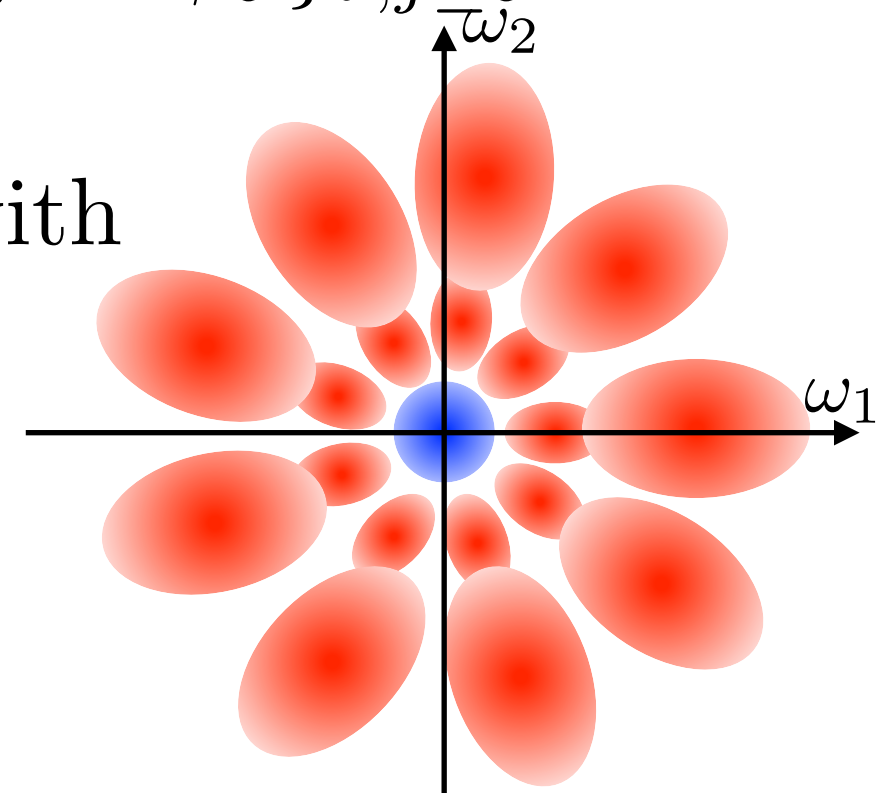
(for sake of simplicity, formula
are given in the isotropic case)

- Wavelet transform: $Wx = \{x \star \psi_{j,\theta}, x \star \phi_J\}_{\theta, j \leq J}$
- Isometric and linear operator of L^2 with

$$\|Wx\|^2 = \sum_{\theta, j \leq J} \int |x \star \psi_{j,\theta}|^2 + \int x \star \phi_J^2$$
- Covariant with translation L_a

$$WL_a = L_a W$$
- Nearly commutes with diffeomorphisms

$$\|[W, L_\tau]\| \leq C \|\nabla \tau\|$$
- A good baseline to describe a signal (here, an image)!

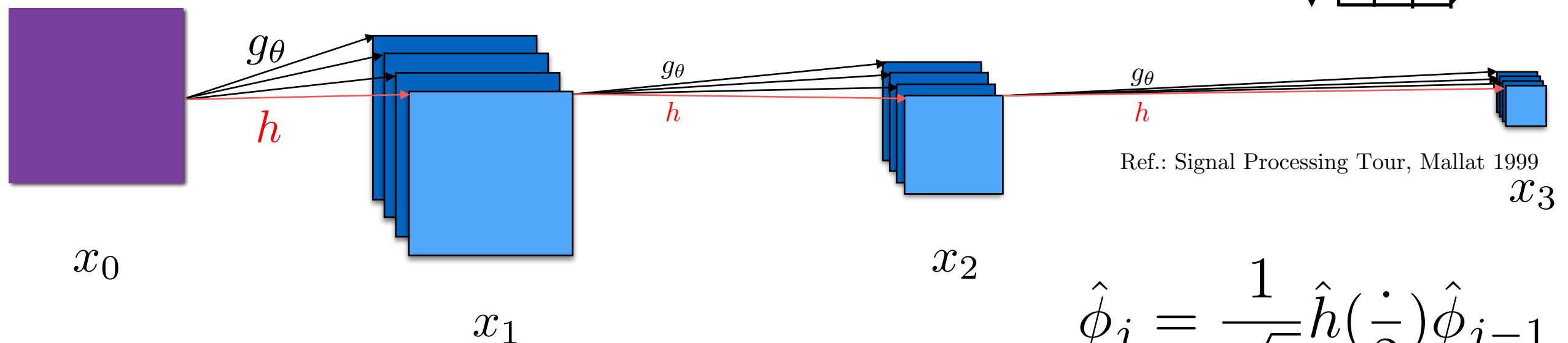
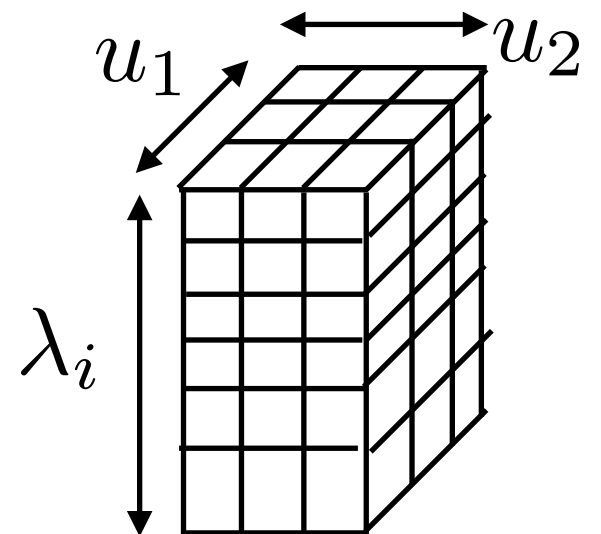


Ref.: Group Invariant Scattering, Mallat S

- Very often, the filters of a CNN have a small support (3x3) and are interlaced with downsampling.

$$y[n, \lambda_{i+1}] = \sum_i x[., \lambda_i] \star k_{\lambda_{i+1}, \lambda_i}[2n]$$

- Similar to a Wavelet Transform.



$$\hat{\phi}_j = \frac{1}{\sqrt{2}} \hat{h}\left(\frac{\cdot}{2}\right) \hat{\phi}_{j-1}$$

$$\hat{\psi}_{j,\theta} = \frac{1}{\sqrt{2}} \hat{g}_\theta\left(\frac{\cdot}{2}\right) \hat{\phi}_{j-1}$$

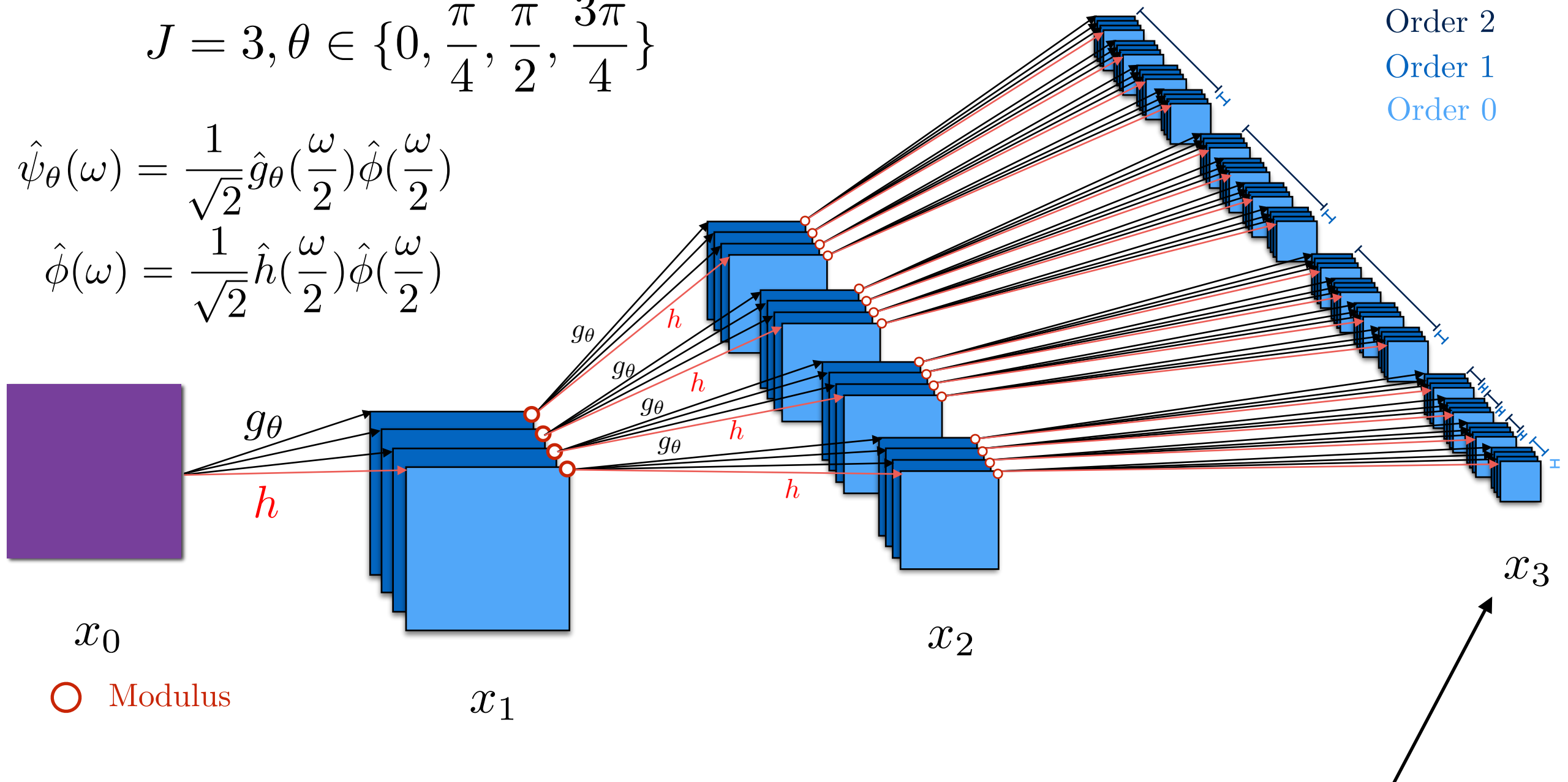
- Except the sum isn't separable.

From wavelet to Scattering

$$J = 3, \theta \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}$$

$$\hat{\psi}_\theta(\omega) = \frac{1}{\sqrt{2}} \hat{g}_\theta\left(\frac{\omega}{2}\right) \hat{\phi}\left(\frac{\omega}{2}\right)$$

$$\hat{\phi}(\omega) = \frac{1}{\sqrt{2}} \hat{h}\left(\frac{\omega}{2}\right) \hat{\phi}\left(\frac{\omega}{2}\right)$$



Has many
invariance properties

$$h \geq 0$$

Scattering as a CNN

Scattering coefficients
are only at the output

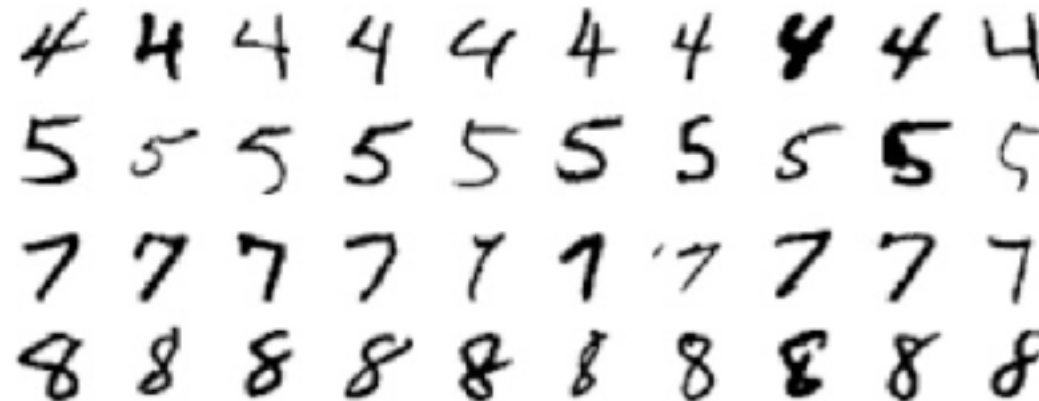
cnrs LIP MLIA

A successful representation⁵³ in image classification

Ref.: Invariant Convolutional Scattering Network, J. Bruna and S Mallat

- Successfully used in several applications:

- Digits



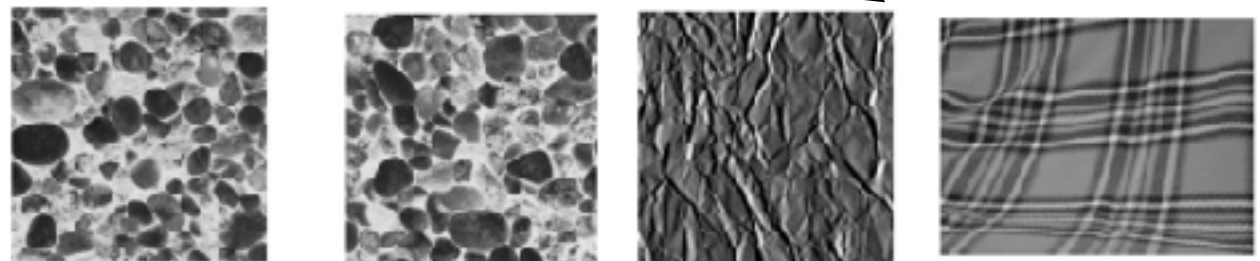
All variabilities
are known

Small deformations
+ Translation

Rotation+Scale

- Textures

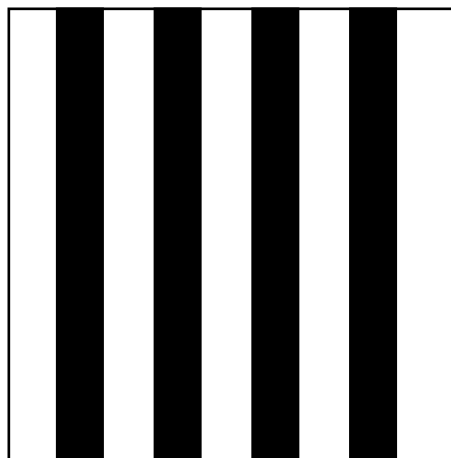
Ref.: Rotation, Scaling and Deformation Invariant Scattering
for texture discrimination, Sifre L and Mallat S.



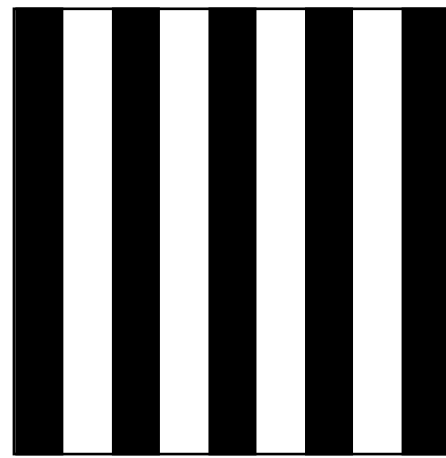
- The design of the scattering transform is guided by the invariance to Euclidean groups and deformations
- To which extent can we compete with other architectures on more complex problems (e.g. variabilities are more complex)? **(still open!)**

Invariant Representations and Deep Learning

Translation



x



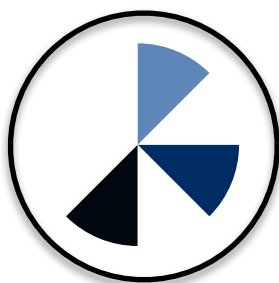
y

Deformations

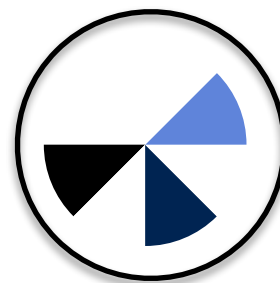
$$L_\tau x(u) = x(u - \tau(u))$$

$$\|x - y\|_2 = 2$$

Rotation

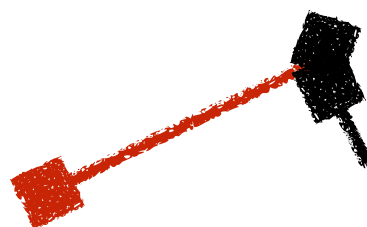
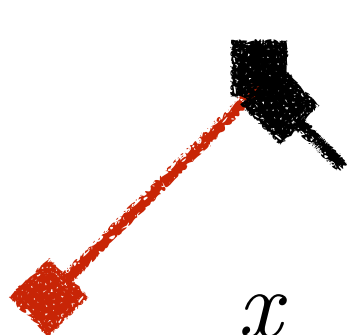


x



y

Averaging is the key
to get invariants



High dimensionality issues



Group invariance

- The notion of convolution can be easily extended on a compact group or a Lie group G via a Haar measure.
- It is the only measure invariant by (left) translations, i.e., $L_*\mu = \mu$ which allows to introduce:

$$L^2(G, \mu) = \{f, \int_G |f|^2 d\mu < \infty\}$$

- And thus the convolution operation:

$$a \star b(g) = \int_G a(\tilde{g})b(\tilde{g}^{-1}g)d\mu(g)$$

- and some Fourier analysis (on Lie groups):

$$\begin{array}{l} \rho : G \rightarrow L^2(G) = \oplus_{\omega} E_{\omega} \\ g \rightarrow \mathcal{L}_g \end{array} \quad \begin{array}{l} \swarrow \\ \text{invariant subspace} \\ \text{of the representation} \end{array}$$

- We say that L is covariant with W if $WL = LW$
example: convolutions!
- We say that A is invariant to L if $AL = A$
- If W (e.g., convolution), ρ (e.g., point-wise non-linearity) are covariant and if A is invariant to L then

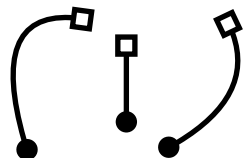
$$\Phi x = AW_J \rho W_{J-1} \rho W_{J-2} \dots W_1 x$$

is invariant. Indeed:

$$\Phi Lx = ALW_J \rho \dots W_1 x = \Phi x$$

- It is also possible to have only an approximate covariance and one measure it via the norm of:

$$[W, L] = WL - LW$$

example: deformation 

Symmetry group hypothesis

Ref.: Understanding deep
convolutional networks
S Mallat

- To each classification problem corresponds a
canonic and unique symmetry group G :

$$\forall x, \forall g \in G, \Phi x = \Phi g.x$$

High dimensional

- We hypothesise there exists **Lie** groups and CNNs
such that:

$$G_0 \subset G_1 \subset \dots \subset G_J \subset G$$

$$\forall g_j \in G_j, \phi_j(g_j.x) = \phi_j(x) \text{ where } x_j = \phi_j(x)$$

- Examples are given by the euclidean group:

$$G_0 = \mathbb{R}^2, G_1 = G_0 \rtimes SL_2(\mathbb{R})$$

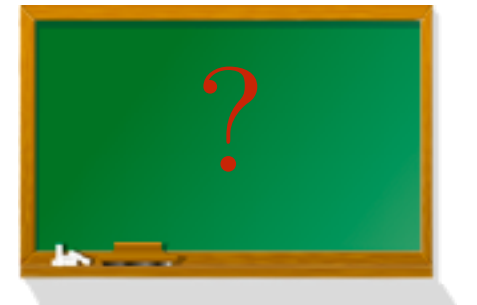
An example: the roto-translation

- If the convolution is defined on G , G' , one can extend it to $G \times G', G \ltimes G'$.

Ref.: PhD of L. Sifre

- Roto-translation (or rigid motions) is a non commutative group:

$$(u, \theta).(\tilde{u}, \tilde{\theta}) = (u + r_{\theta}\tilde{u}, \theta + \tilde{\theta})$$



- ... and this leads to the following convolution:

$$(Y \circledast \Psi)(g) = \int_{g'} Y(g') \Psi(g'^{-1}g) dg$$

- Interestingly, CNNs often incorporate some poolings \mathcal{P} , which satisfy for $\|I - \mathcal{L}\| \ll 1$: $\mathcal{P}\mathcal{L} \approx \mathcal{P}$.
- It allows to progressively induce more invariance. (and it's very similar to a Wavelet Transform)
- Similarly, the non-linearity is point-wise. Interestingly, point-wise non-linearity are the only non-linearity ρ that commutes with deformations, ie

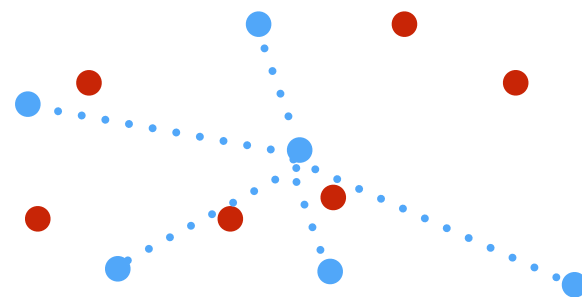
$$\rho L = L\rho \quad \text{iff} \quad \forall x = (x_1, \dots, x_d), \rho(x) = (\rho(x_1), \dots, \rho(x_d))$$

- Weak differentiability property:

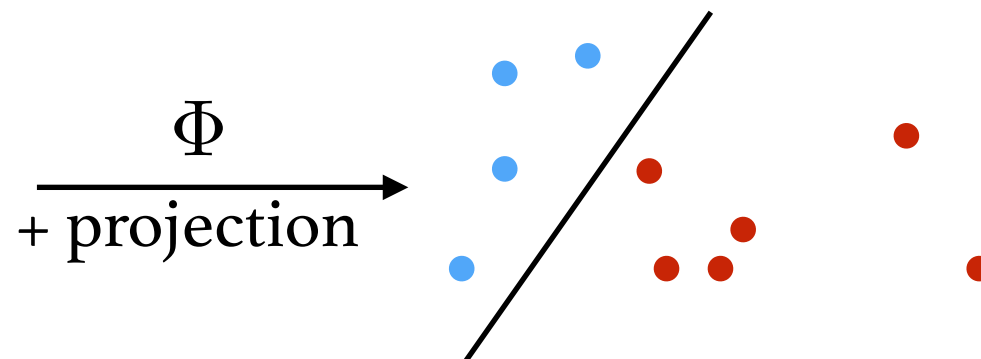
$$\sup_L \frac{\|\Phi Lx - \Phi x\|}{\|Lx - x\|} < \infty \Rightarrow \exists \text{ "weak" } \partial_x \Phi$$

$$\Rightarrow \Phi Lx \approx \Phi x + \underbrace{\partial_x \Phi L}_{\text{A linear operator}} + o(\|L\|)$$

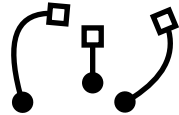
... Displacement L



- A linear projection (to kill L) build an invariant



Deformations
 $L_\tau x(u) = x(u - \tau(u))$



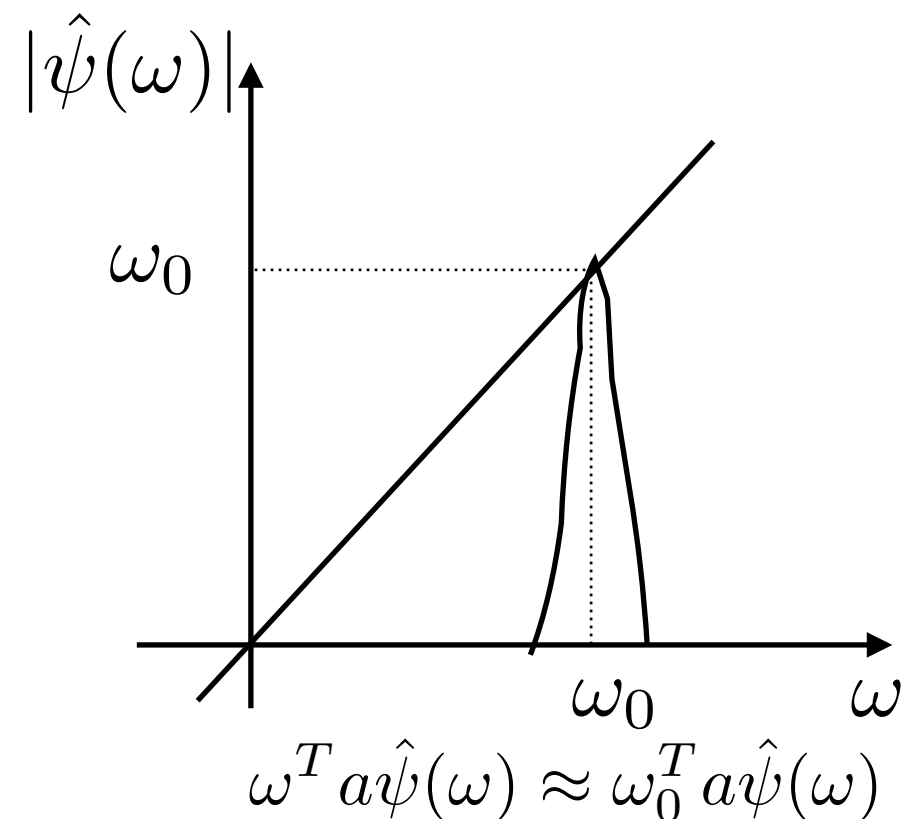
- Analytic wavelets permit to build stable invariants

to:

- small translations by a :

$$\widehat{L_a x \star \psi}(\omega) = e^{i\omega^T a} \hat{x}(\omega) \hat{\psi}(\omega)$$

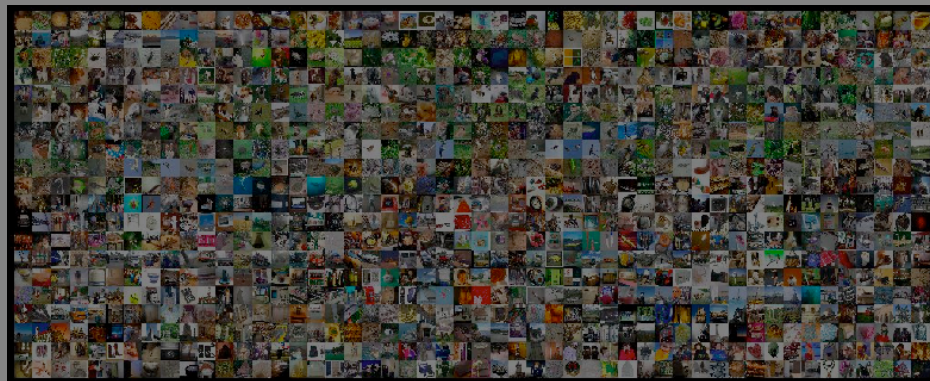
Ref.: Group Invariant Scattering, Mallat S



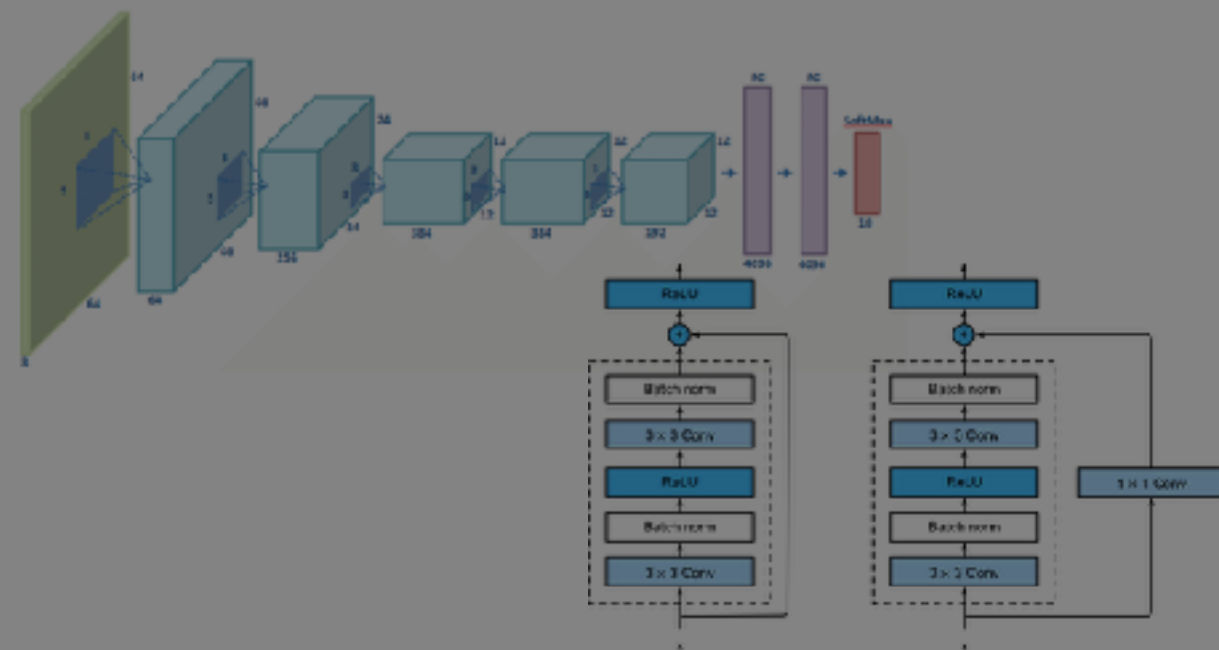
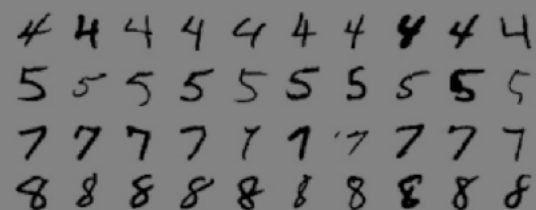
The variability corresponds to a phase multiplication!

- small deformations:

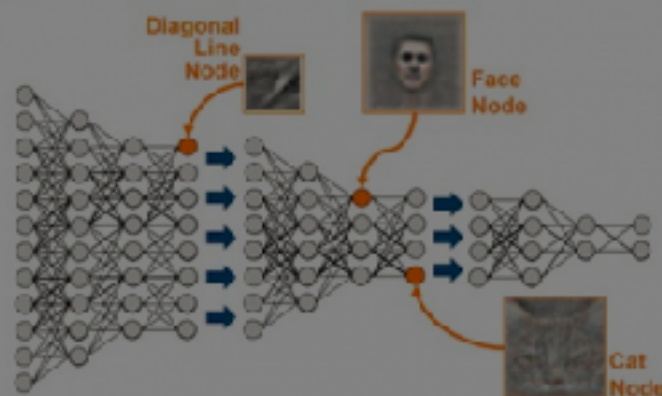
$$\|(L_\tau x) \star \psi - L_\tau(x \star \psi)\| \leq C \nabla \|\tau\|_\infty$$



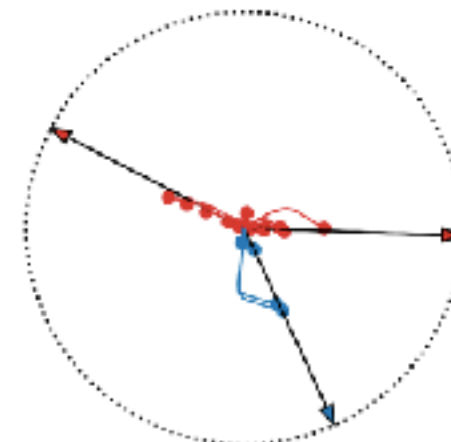
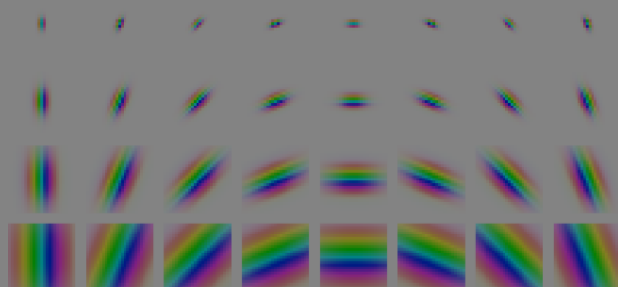
Introduction to high-dimensional tasks



Engineering Deep Neural Networks



Understanding Convolutional Neural Networks

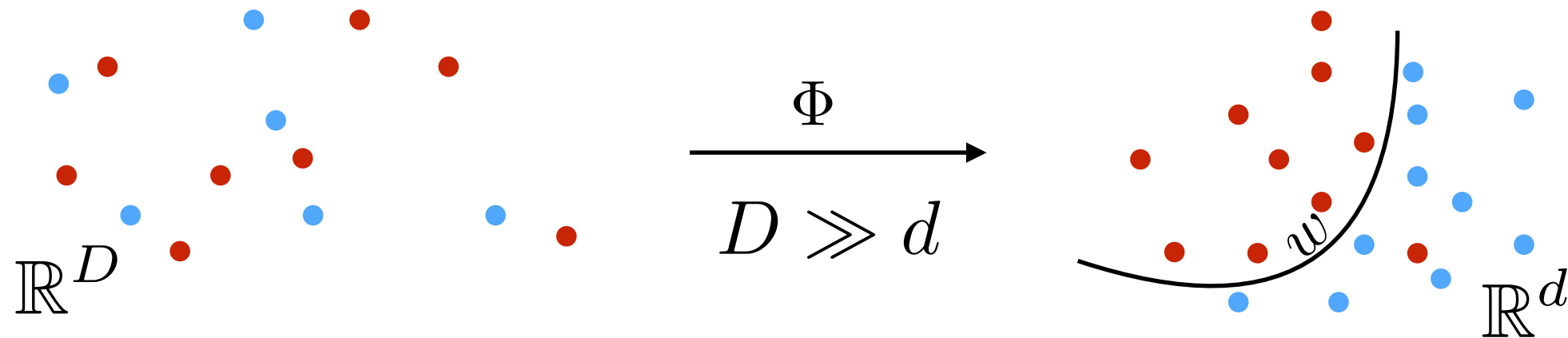


Under the Hood of Neural Networks

$$C_f = \int_{\mathbb{R}^D} \|\omega\|_1 |\hat{f}(\omega)| d\omega$$

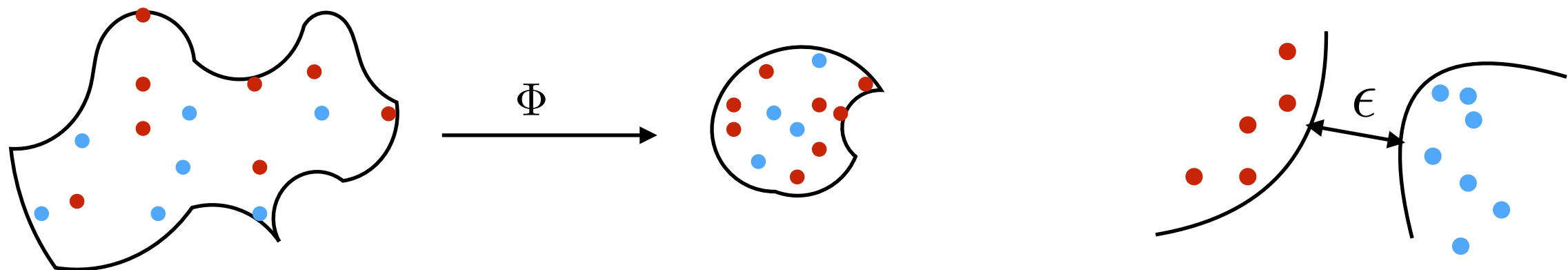
Classification mechanism

- **Objective:** building a representation Φx of x such that a simple (say euclidean) classifier \hat{y} can estimate the label y :



- Designing Φ : must be regular with respect to the class:

$$\|\Phi x - \Phi x'\| \lll 1 \Rightarrow \hat{y}(x) = \hat{y}(x')$$
- **Necessary** dimensionality reduction and separation to break the curse of dimensionality:

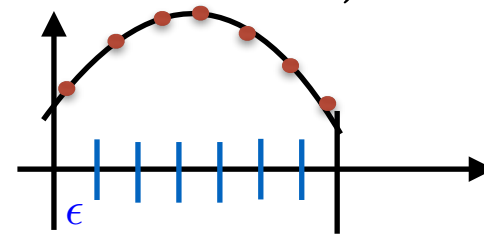


Model on the data: low

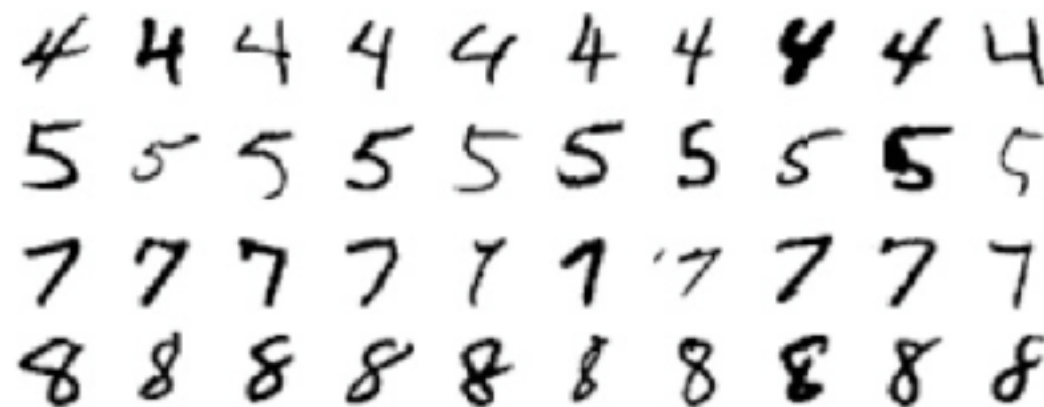
dimensional manifold hypothesis?

- Low dimensional manifold: dimension up to 6. Not higher:

Property: if $f : \mathbb{R}^D \rightarrow [0, 1]$ is 1-Lipschitz, then let $N_\epsilon = \arg \inf_N \sup_{i \leq N} (|f(x) - f(x_i)| < \epsilon)$.
Then $N_\epsilon = \mathcal{O}(\epsilon^{-D})$



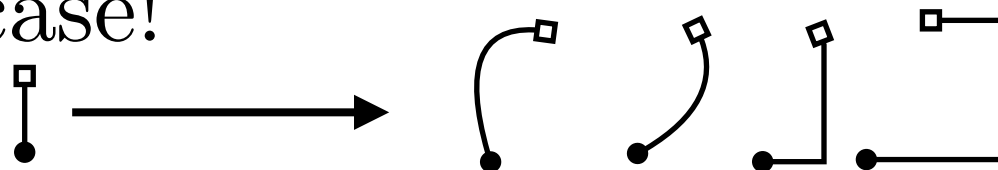
- Can be true for MNIST...



All variabilities
are known

Small "limited" deformations
+ Translation

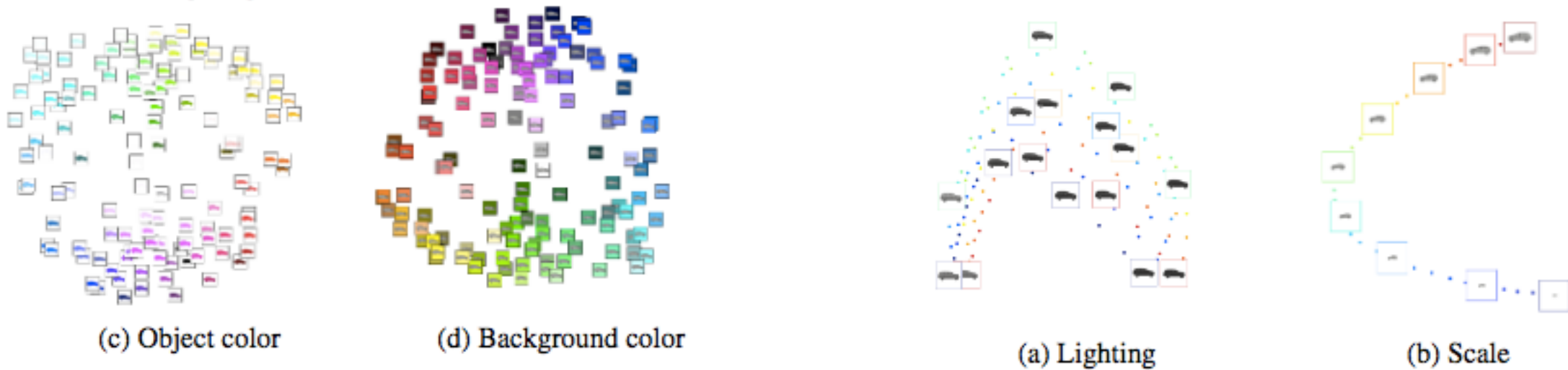
- Yet high dimensional deformations are an issue in the general case!



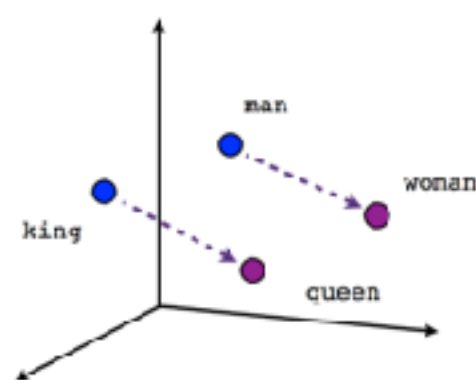
Flattening the space: progressive manifold?

- Parametrize variability on synthetic data: $L_\theta, \theta \in \mathbb{R}^d$ and observe it after PCA

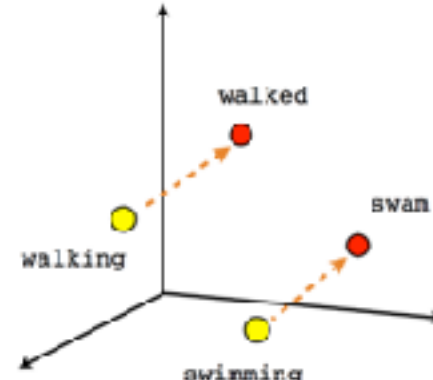
Ref.: Understanding deep features with computer-generated imagery, M Aubry, B Russel



- Data tends to live on flattened space. Tangent space?



Male-Female



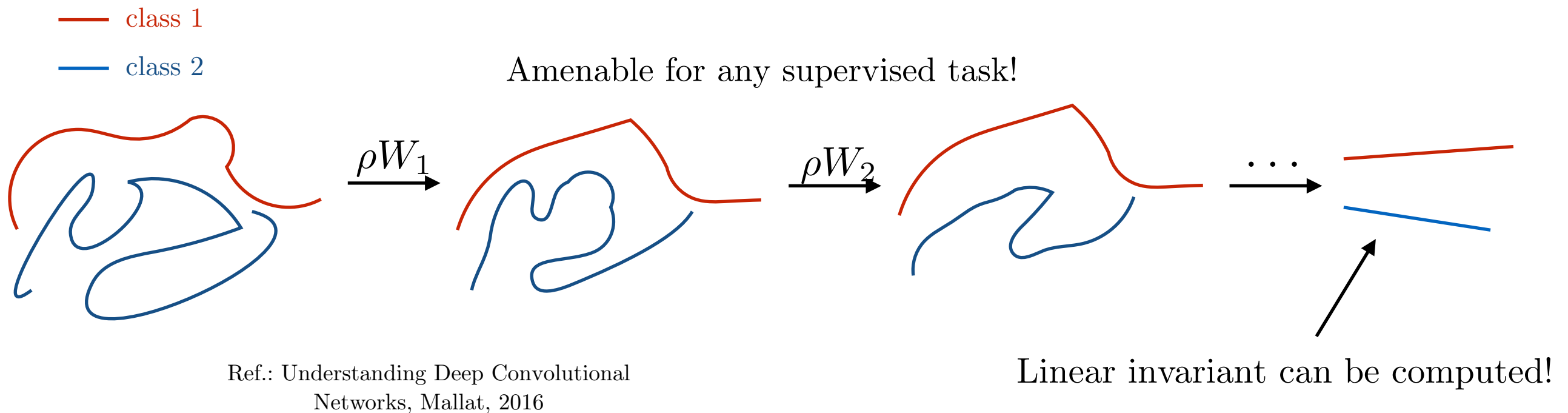
Verb tense



Country-Capital

Difficult to find evidences of such phenomenon

Mechanism proposal: Flattening the level sets



- How to linearize? Ex.: Gâteaux differentiability

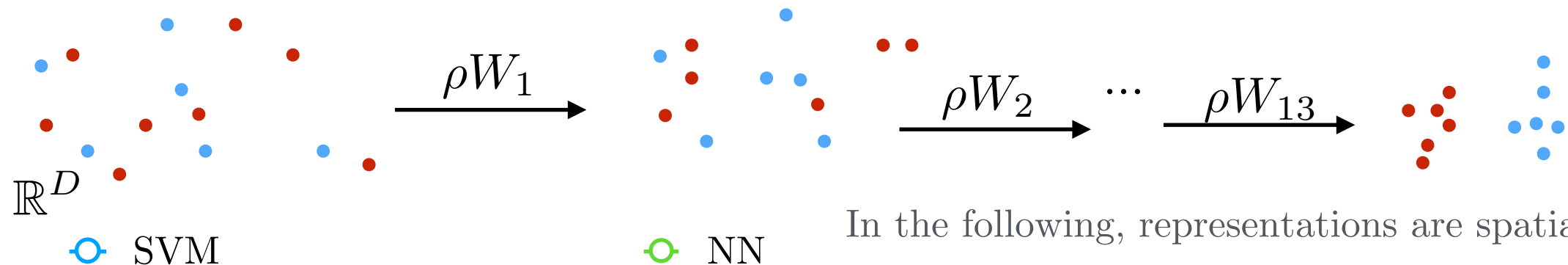
$$\exists C_x, \sup_{\mathcal{T}} \frac{\|\Phi x - \Phi \mathcal{T} x\|}{\|\mathcal{T}\|} < C_x \Rightarrow \exists \partial \Phi_x : \Phi \mathcal{T} x \approx \Phi x + \partial \Phi_x \cdot \mathcal{T}$$

- However, exhibiting \mathcal{T} can be difficult. (*curse of dimensionality*)

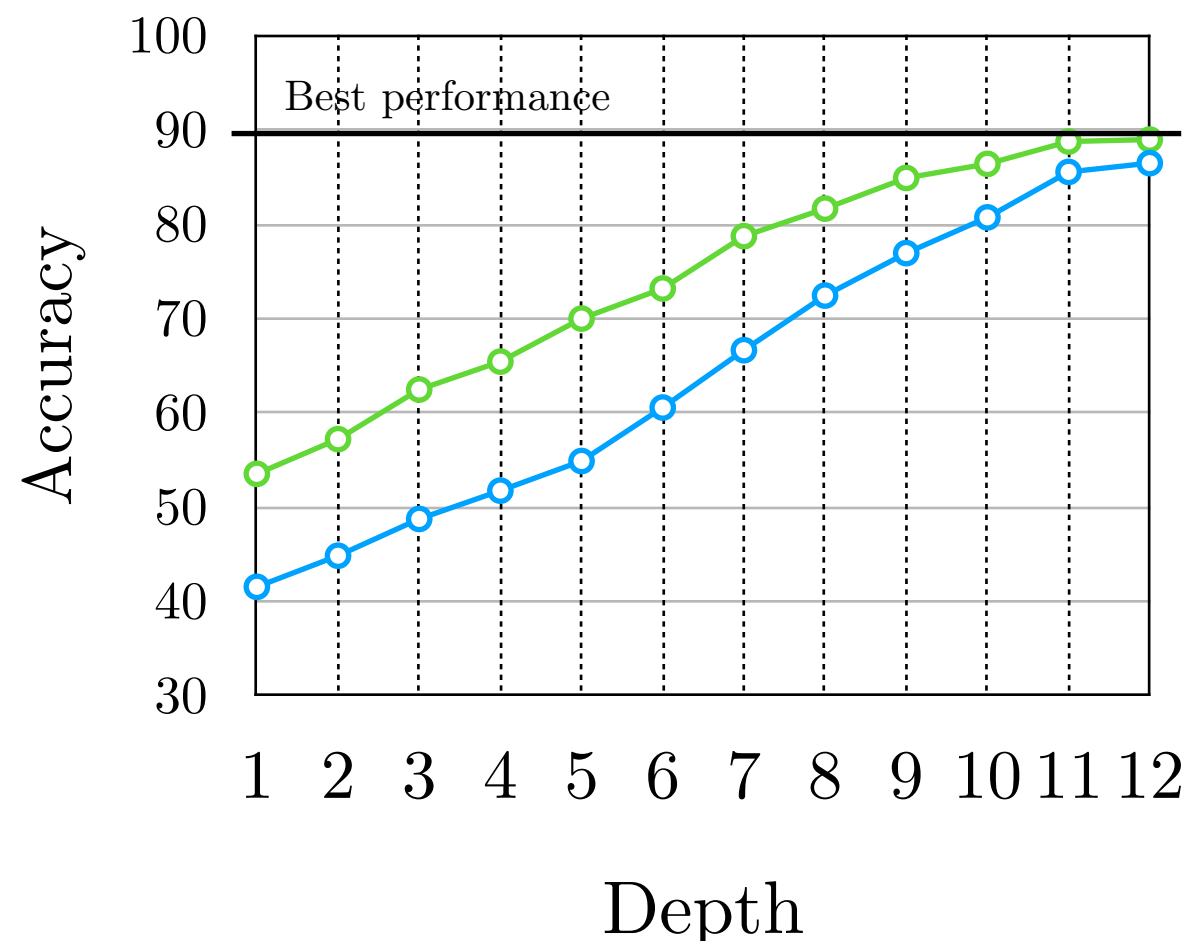
Ex.: linear translations $\mathcal{T}_a(x)(u) \triangleq x(u + a)$, yet non linear case?

Empirical observation: Progressive separability

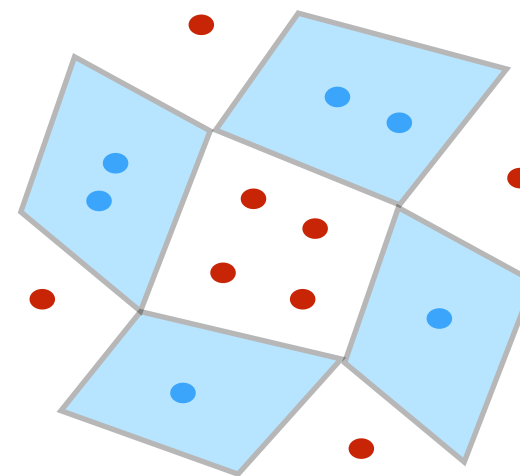
- Typical CNN exhibits a progressive contraction & separation, w.r.t. the depth:



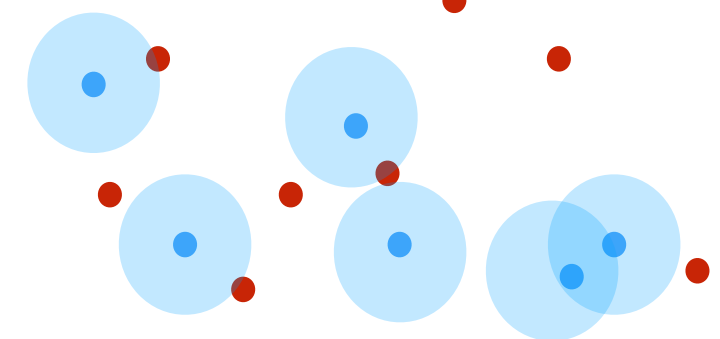
In the following, representations are spatially averaged.



Nearest Neighbor (NN)



Gaussian SVM



Localised classifiers

Ref.: Building a Regular Decision Boundary with Deep Networks, EO

- How can we explain it?

A mysterious black-box

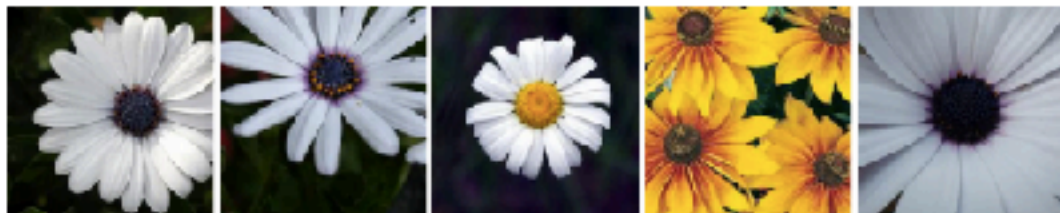
Concept of neuron?

- Consider: $v \in \mathbb{R}^{1000}$, $x_v = \arg \max_{x \in \mathcal{D}} \langle \Phi x, v \rangle$ ← dataset

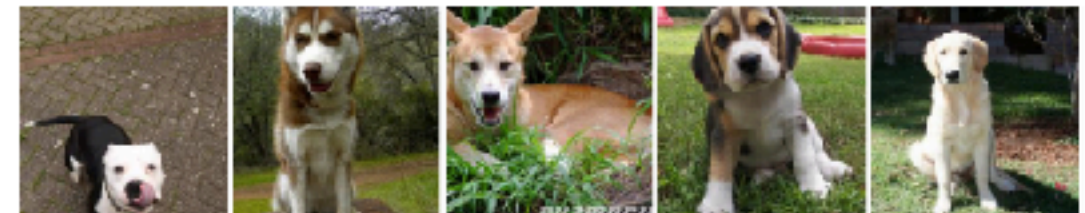
- Claim 1: $v = (0, \dots, 0, 1, 0, \dots, 0)$ has a semantic meaning

Ref.: Intriguing properties of Deep Neural Networks, Szegedy et al.

- Claim 2: any unit norm v has a semantic meaning.



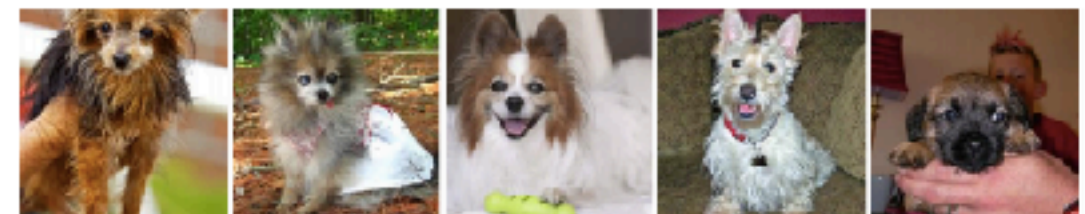
(a) Direction sensitive to white, spread flowers.



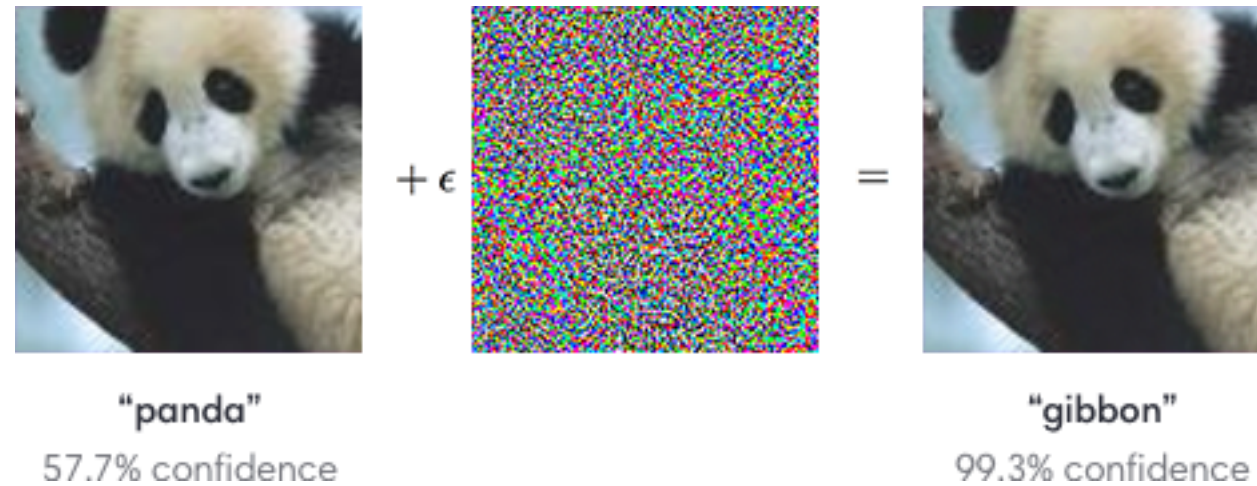
(b) Direction sensitive to white dogs.



(c) Direction sensitive to spread shapes.



(d) Direction sensitive to dogs with brown heads.



- NNs are super sensitive to input noise
- Indeed, the NN is at most $\|W_1\| \dots \|W_J\|$ -Lipschitz

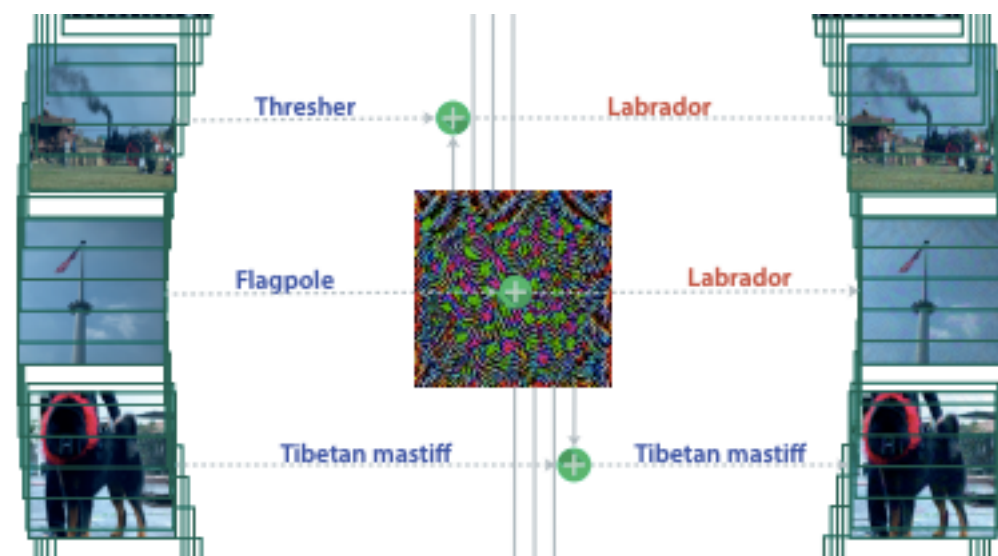
Ref.: Lipschitz Regularity of deep neural networks, Scaman and Virmaux

$$\inf_{\Phi(x) \neq \Phi(x+\epsilon)} \|\epsilon\|$$

Or even for every class, there are algorithms with parameters (ϵ, κ) s.t.:

$$\begin{cases} \mathbb{P}(\Phi(X + \delta) \neq \Phi(X)) \geq 1 - \kappa \\ \|\delta\| \leq \epsilon \end{cases}$$

Ref.: Universal adversarial perturbations, Moosavi et al.

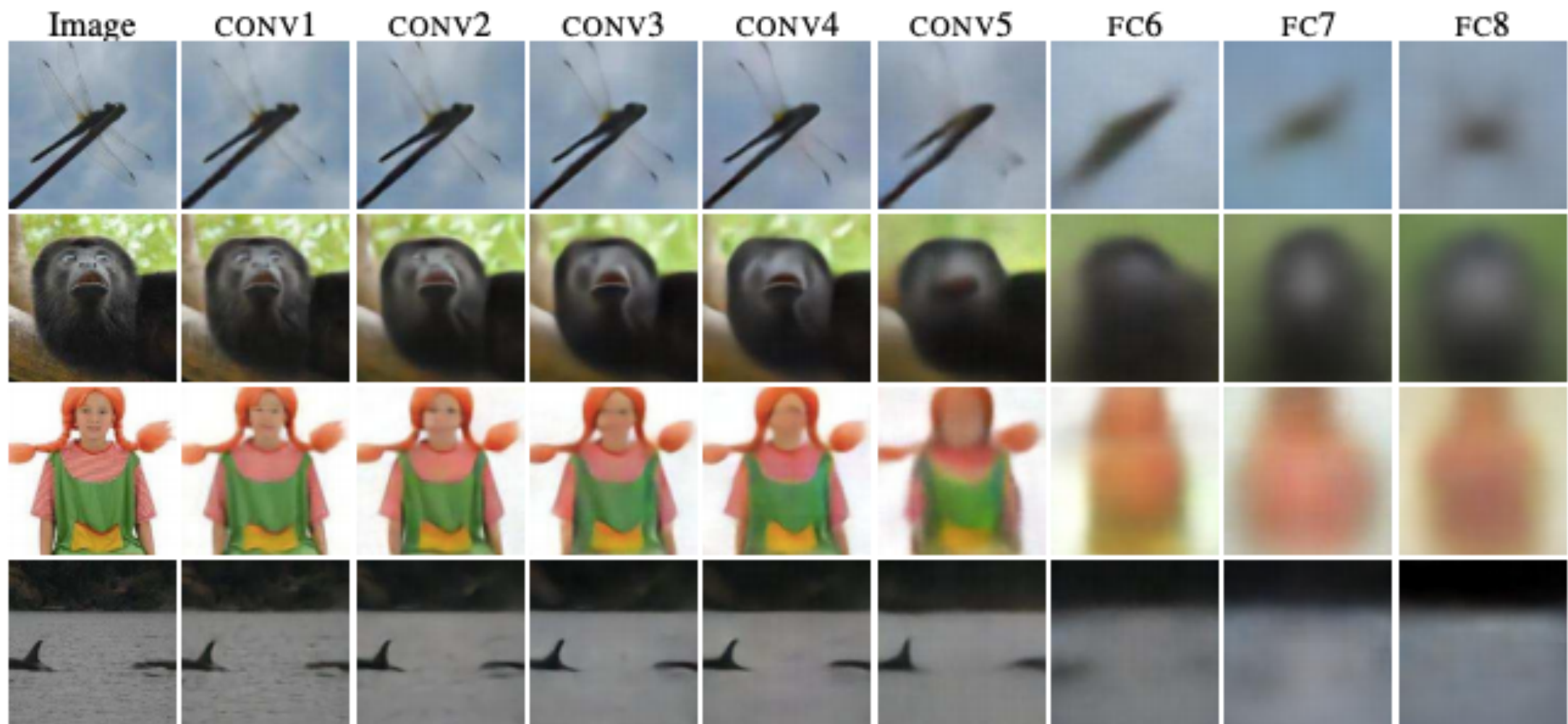


Reconstruction from a given layer?

$$X_0 \longrightarrow \boxed{W_1} \longrightarrow \boxed{\rho} \longrightarrow \boxed{W_2} \quad \dots \quad \longrightarrow \boxed{\rho} \longrightarrow \boxed{W_J} \longrightarrow \Phi(x)$$

Learn the operators!

$$\varphi_j \longrightarrow \boxed{\tilde{W}_j} \longrightarrow \boxed{\rho} \longrightarrow \boxed{\tilde{W}_{j-1}} \longrightarrow \dots \longrightarrow \boxed{\rho} \longrightarrow \boxed{\tilde{W}_1} \longrightarrow X(\varphi_j)$$



Reducing mutual information (Information bottleneck)

- Reducing the information sounds relevant:

$$I(X; Y) = \int_{\mathbb{R}^2} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy = H(X) - H(X|Y)$$

Ref.: Opening the Black Box of Deep Neural Networks via Information, R Shwartz-Ziv and N Tishby

Measures the dependancy between variables

$$I(X; \Phi_1 X) \geq I(X; \Phi_2 X) \geq \dots \geq I(X; \Phi_J X)$$

"Compress" X

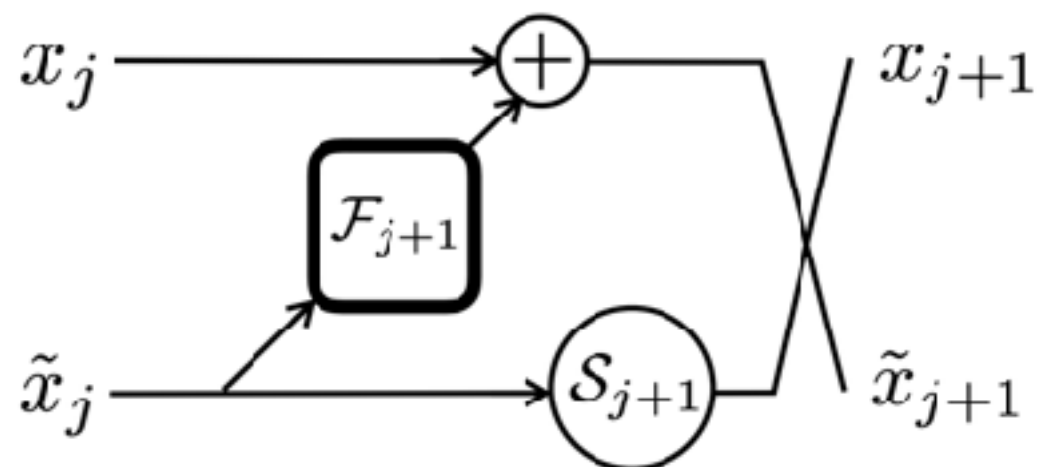
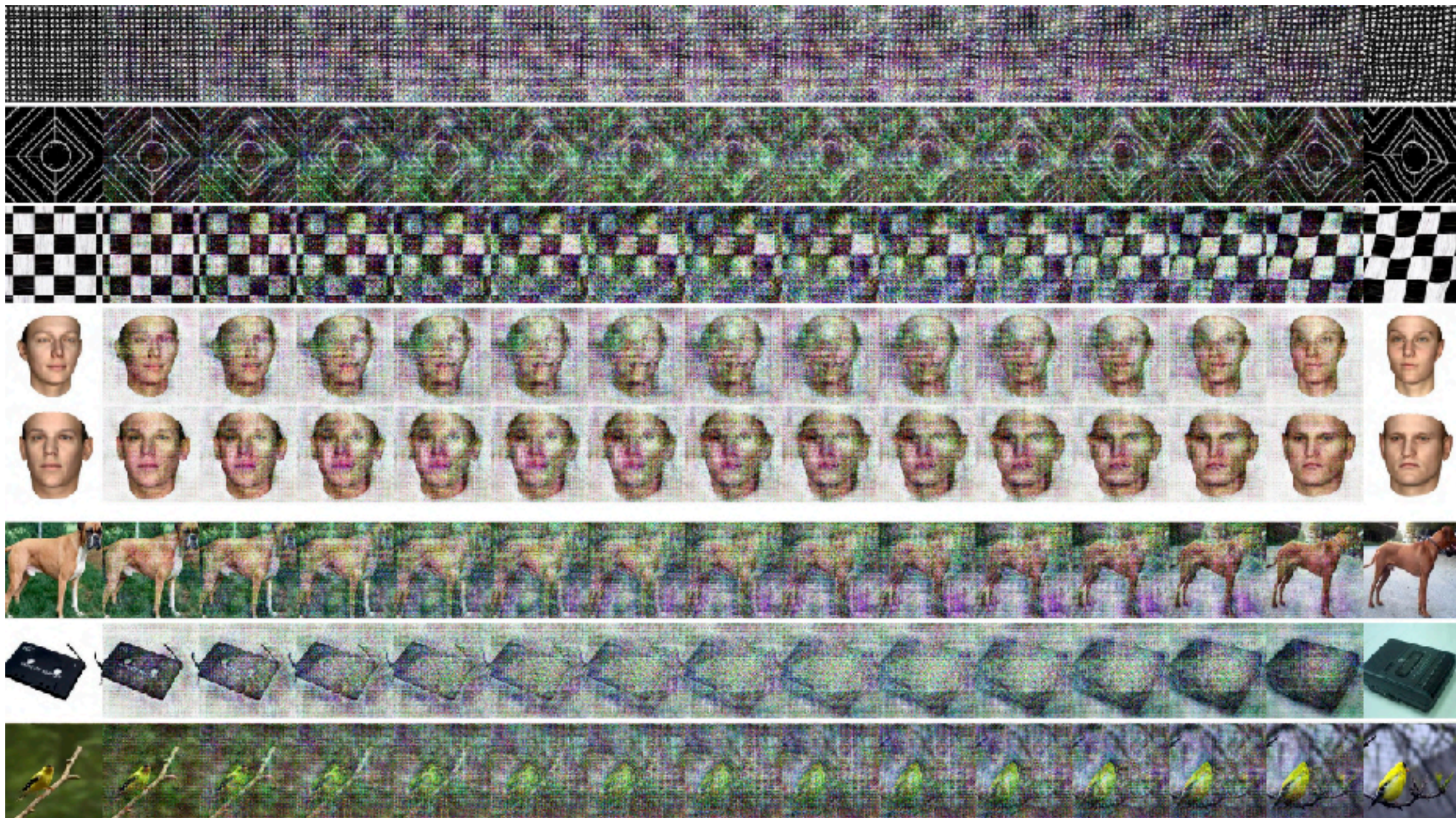
$$I(X; Y) \geq I(\Phi_1 X; Y) \geq \dots \geq I(\Phi_J X; Y)$$

... but "reveal" Y

They propose to introduce:

$$\Phi_{j,\lambda} = \arg \inf_{\Phi} I(\Phi_{j-1} X, \Phi_j X) - \lambda I(\Phi_j X, Y)$$

- But one can easily build invertible CNNs...



Ref.: i-Revnet, depp invertible
networks Jacobsen, Smeulder and
EO

One study case:
1-hidden layer NNs

- 1-hidden layer neural networks are one of the simplest instance of Neural Networks. A real valued 1-NN writes:

$$\Phi(x) = \sum_{i \leq n} \alpha_i \rho(\langle x, \theta_i \rangle) \quad \text{where } x \in \mathcal{B}(0, 1), (\alpha_i, \theta_i) \in \mathbb{R} \times \mathbb{R}^d$$

- If ρ is homogeneous, i.e. $\forall \lambda > 0, \forall x \in \mathbb{R}, \rho(\lambda x) = \lambda \rho(x)$, we get:

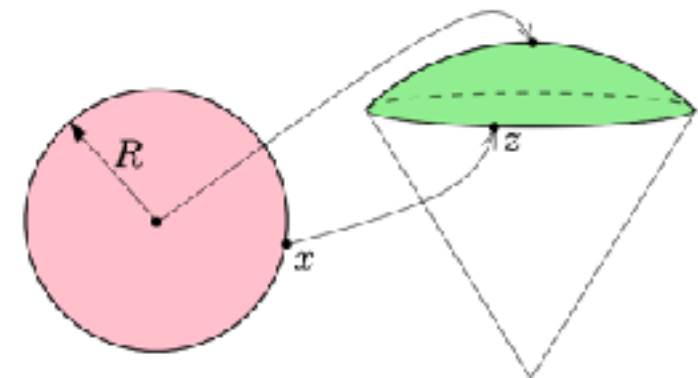
$$\Phi(x) = \sum_{i \leq n} \alpha_i \|\theta_i\| \rho\left(\left\langle x, \frac{\theta_i}{\|\theta_i\|} \right\rangle\right)$$

ReLU!

$$= \int_{\mathcal{S}^{d-1}} \rho(\langle x, \theta \rangle) d\mu_n(\theta) \quad \text{with } \mu_n = \sum_{i \leq n} \alpha_i \|\theta_i\| \delta_{\frac{\theta_i}{\|\theta_i\|}}$$

- Then let $|t| = \sqrt{1 - \|x\|^2}$ and let:

$$\tilde{\Phi}((x, t)) \triangleq |t| \Phi\left(\frac{x}{t}\right) = \Phi(x) \quad \text{if } t > 0$$



- We thus have a function parametrised by a measure of the sphere and defined over the sphere \mathcal{S}^{d-1} !

- Instead to consider finite measures $\mu \in \mathcal{M}(\mathcal{S}^{d-1})$, consider as a reference measure the uniform measure σ and:

$$L^2(\mathcal{S}^{d-1}) = \left\{ \int_{\mathcal{S}^{d-1}} |p|^2 d\sigma < \infty \right\}$$

which is of the type: $d\mu = p d\sigma$ (no dirac!)

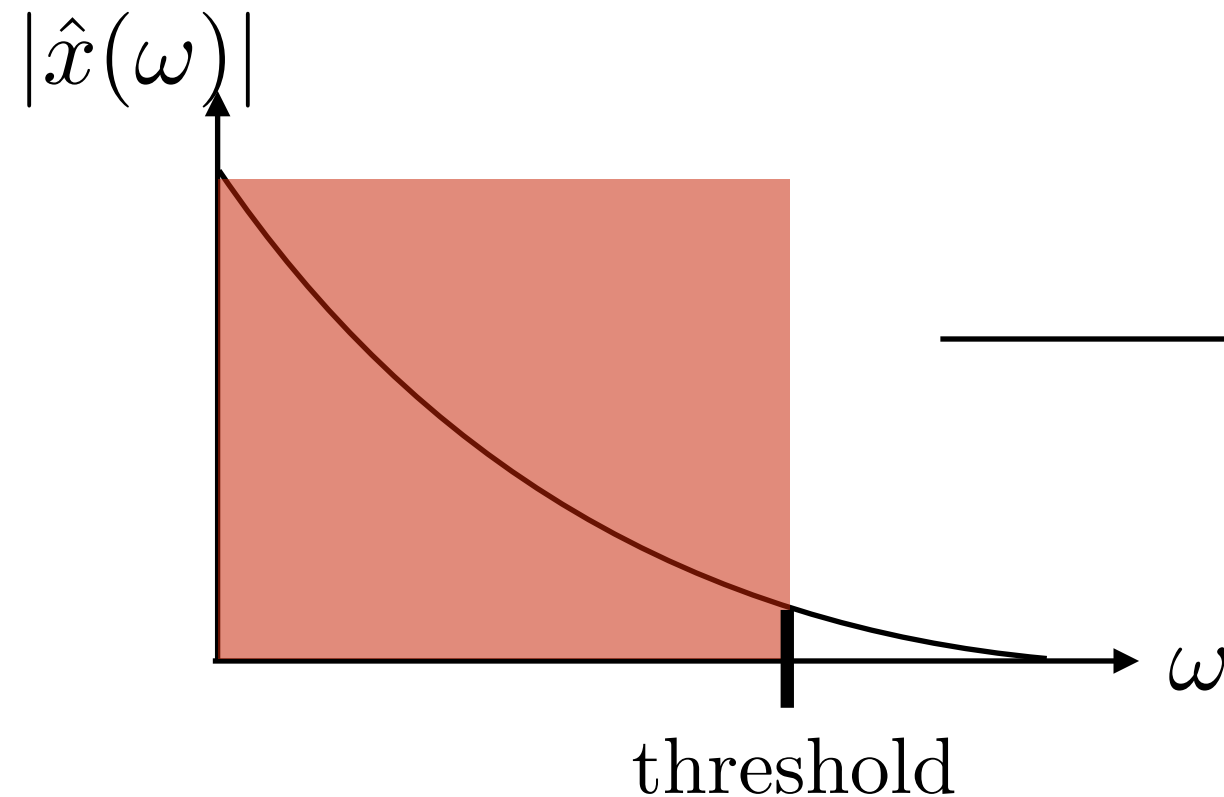
- And we study approximation of f regular enough of the type:
 $\forall x \in \mathcal{S}^{d-1},$

$$f(x) \approx \int_{\mathcal{S}^{d-1}} \rho(\langle x, \theta \rangle) p(\theta) d\sigma = \rho \circledast p(x)$$

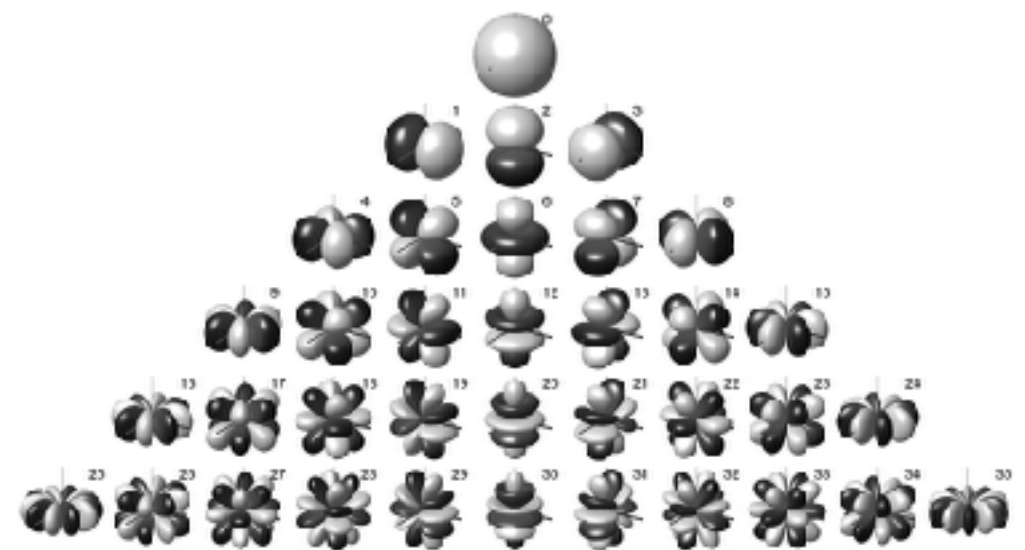
convolutions!

well defined because
 ρ is bounded

regularity to define



Fourier analysis-like
can be done on \mathcal{S}^{d-1}
via spherical harmonics



Allows to derive
approximation bounds
of Lipschitz
functions with $\rho \circledast p$

Rates: for functions p of $L^2(\mathcal{S}^{d-1})$ and $\epsilon > 0$, we can find a finitely supported μ such that for $x \in \mathcal{S}^{d-1}$:

Ref.: Breaking the curse of dimensionality with
convex neural networks, F Bach

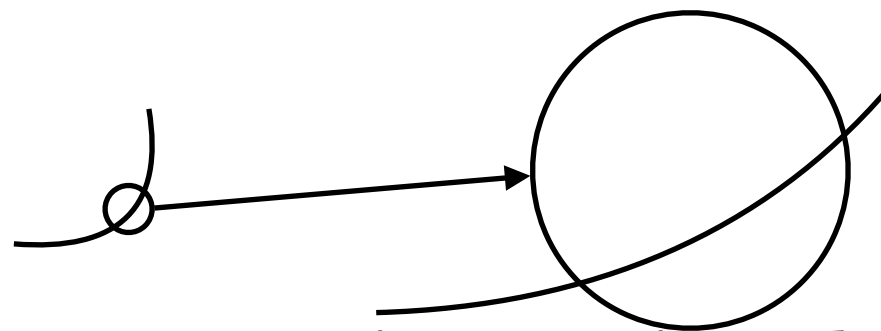
$$|(\rho \circledast p)(x) - \int_{\mathcal{S}^{d-1}} \rho(\langle x, \theta \rangle) d\mu(\theta)| \leq \epsilon \|p\|$$

and $\#(\text{Support}(\mu)) \leq C(d) \epsilon^{-\frac{2d}{d+3}}$

- **Lazy training:** the idea that neural networks behaves like their linearised counter part due to rescaling effects in the asymptotic regime.

linearisation: $\Phi(\Theta) \approx \Phi(\Theta_0) + (\Theta - \Theta_0)^\top \nabla \Phi(\Theta_0)$

$$\Phi(x) = \gamma(n) \sum_{i \leq n} \alpha_i \rho(\langle x, \theta_i \rangle) \text{ then } \gamma(n) = o\left(\frac{1}{\sqrt{n}}\right) \Rightarrow \text{laziness}$$

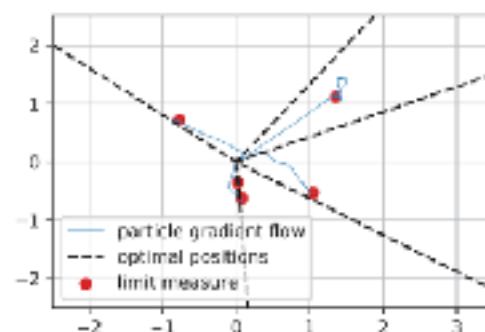


Ref.: On Lazy Training in Differentiable Programming, Chizat et al.

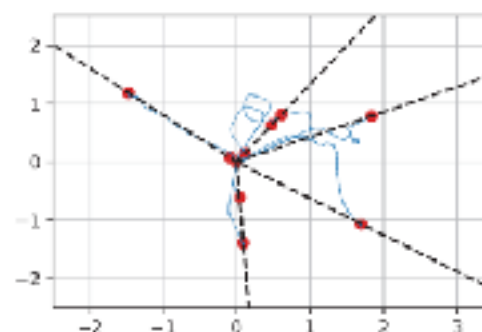
- **Optimization of 1-NNs via optimal transport:** many asymptotic results (global optimum!) if the non-linearity is homogeneous.

f^* has 5 neurons

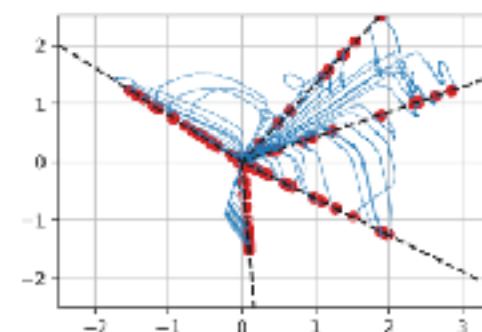
Ref.: On the global convergence of gradient descent for over-parameterized models using optimal transport, Chizat and Bach



5 neurons



10 neurons



100 neurons

- Under non-restrictive assumptions (e.g., satisfied by ReLU) on ρ , there exists constant $c, C > 0$, such that for any dimension d , there exists a measure μ and $g : \mathbb{R}^d \rightarrow \mathbb{R}$ such that
- g is bounded, with support in $\mathcal{B}(0, C\sqrt{d})$ and can be approximate by a 3 layers NN with a polynomial width.
- BUT any 2 layers NN g such that $\int |f - g|^2 d\mu \leq c$ has an exponential width.

More about my research

- Asynchronous Distributed Optimization (might have some funded positions soon)
- Interpretability in Deep Learning
- Interferometric Graph Transform (\approx Scattering for Graphs)
- Tabular data

- Deep learning is a super excited topic because of it solves many tasks...
- ... yet much has to be done to understand it much better.
- **Resources:**
<https://edouardoyallon.github.io>, papers, codes
<https://edouardoyallon.github.io/MAP670R-2020/notes.pdf>, more about 1-NNs and Deep Learning theory!

Thanks for your attention!