

One Signal Processing view on Deep Learning

Edouard Oyallon

edouard.oyallon@lip6.fr

CNRS, LIP6



Overview of the lecture

Intro: Image classification or generation, some challenging tasks.

Intro: Image classification or generation, some challenging tasks.

A. Fighting the curse of dimensionality with Deep Neural Networks.

1. A fantastic tool to empirically solve high dimensional tasks...
2. ... that requires many recipes to be trained.

Intro: Image classification or generation, some challenging tasks.

A. Fighting the curse of dimensionality with Deep Neural Networks.

1. A fantastic tool to empirically solve high dimensional tasks...
2. ... that requires many recipes to be trained.

B. Interpretability in deep learning.

1. Under the hood of neural networks.
2. Invariant Representations and Deep Neural Networks.

Intro: Image classification or generation, some challenging tasks.

A. Fighting the curse of dimensionality with Deep Neural Networks.

1. A fantastic tool to empirically solve high dimensional tasks...
2. ... that requires many recipes to be trained.

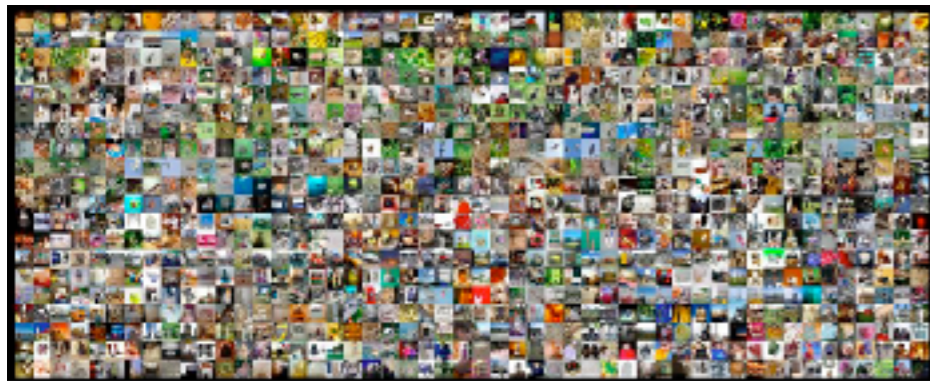
B. Interpretability in deep learning.

1. Under the hood of neural networks.
2. Invariant Representations and Deep Neural Networks.

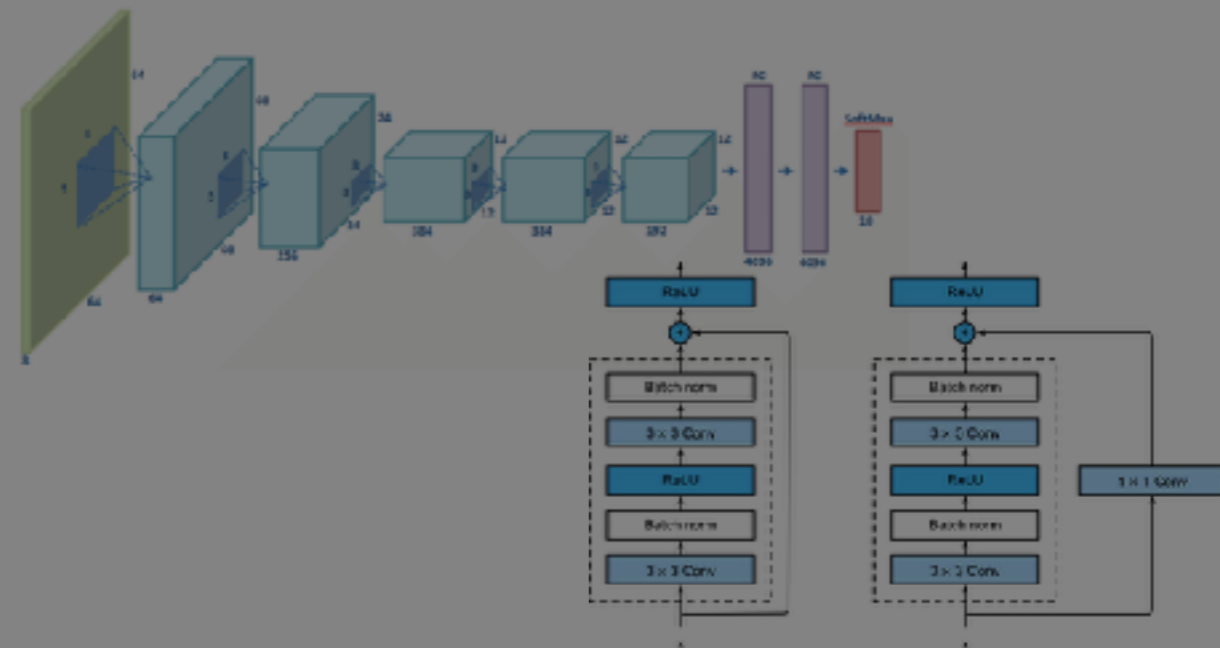
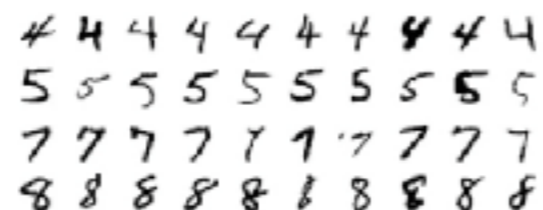
C. Statistical learning results.

1. An opaque black-box from the learning theory perspective.
2. Sometimes, well understood: 1 hidden-layer Neural Networks

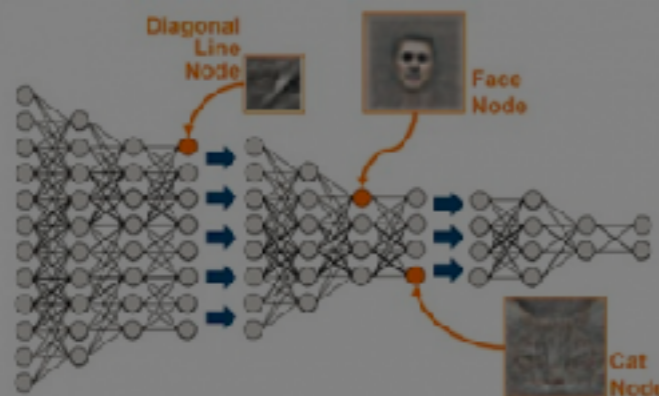
- A. Make your own Invertible Neural Networks
- B. A tutorial to the Scattering Transform
- C. (if we have time) Get insights on a pretrained model.



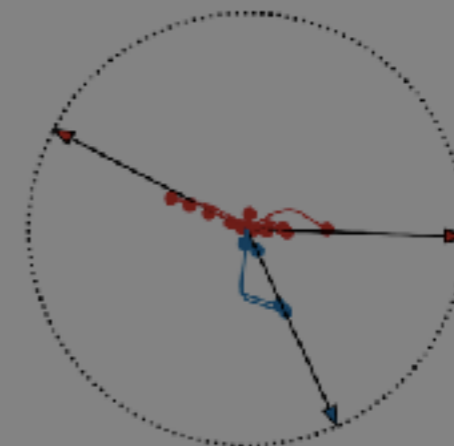
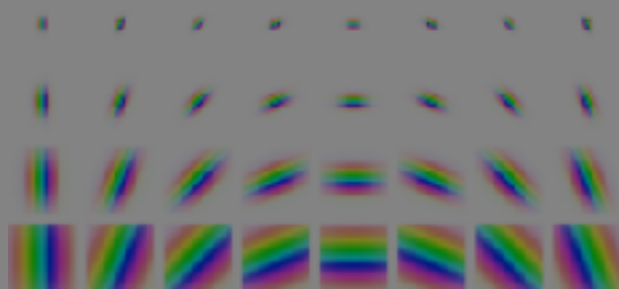
Introduction to image classification



Fighting the curse of dimensionality with Deep Neural Networks

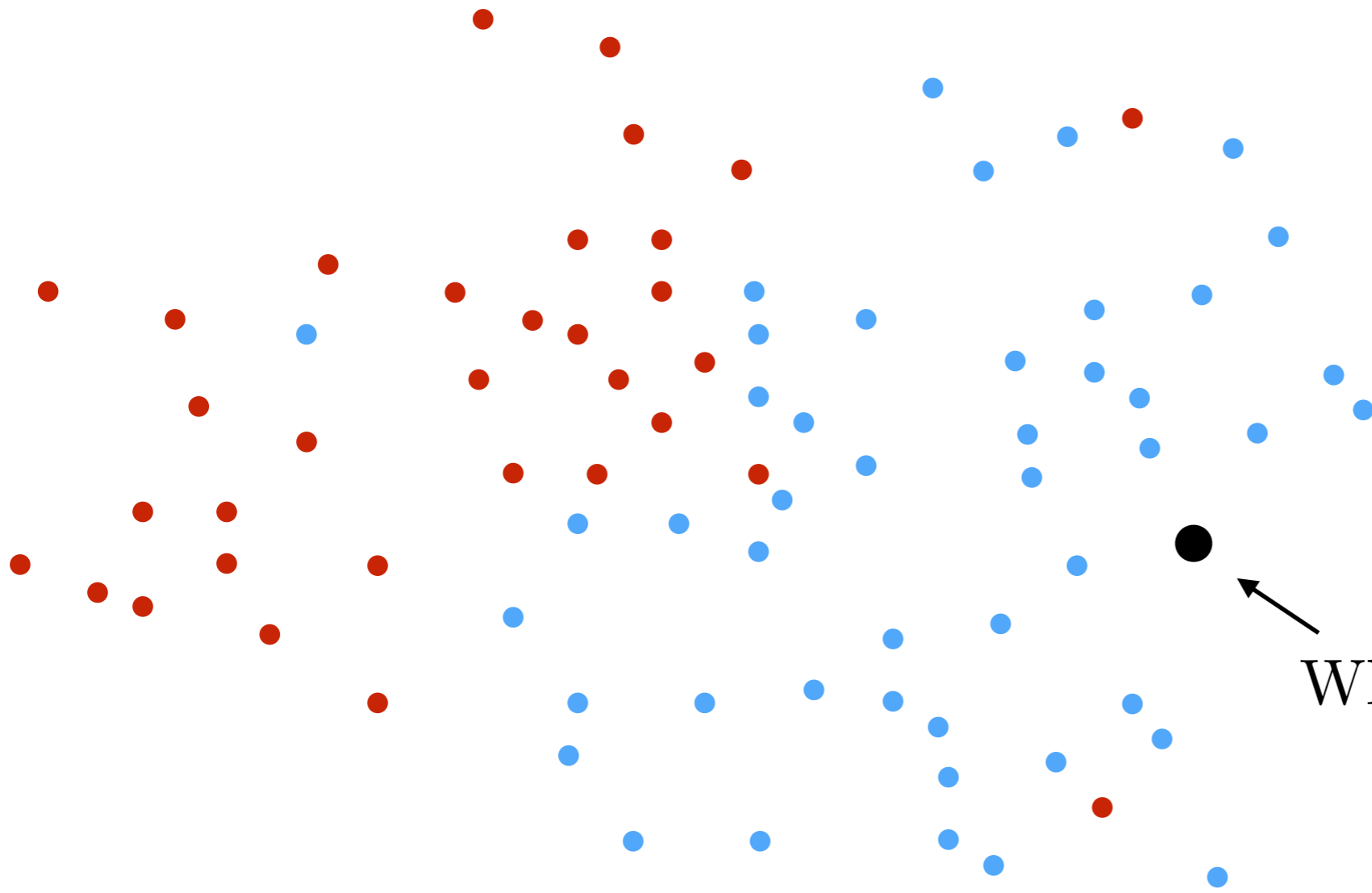


Interpretability in Deep Learning



Statistical learning results

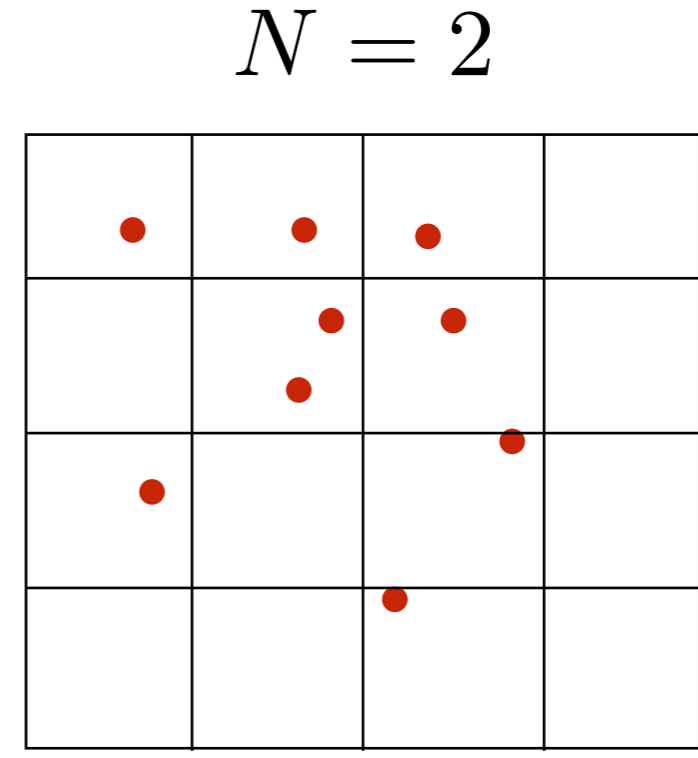
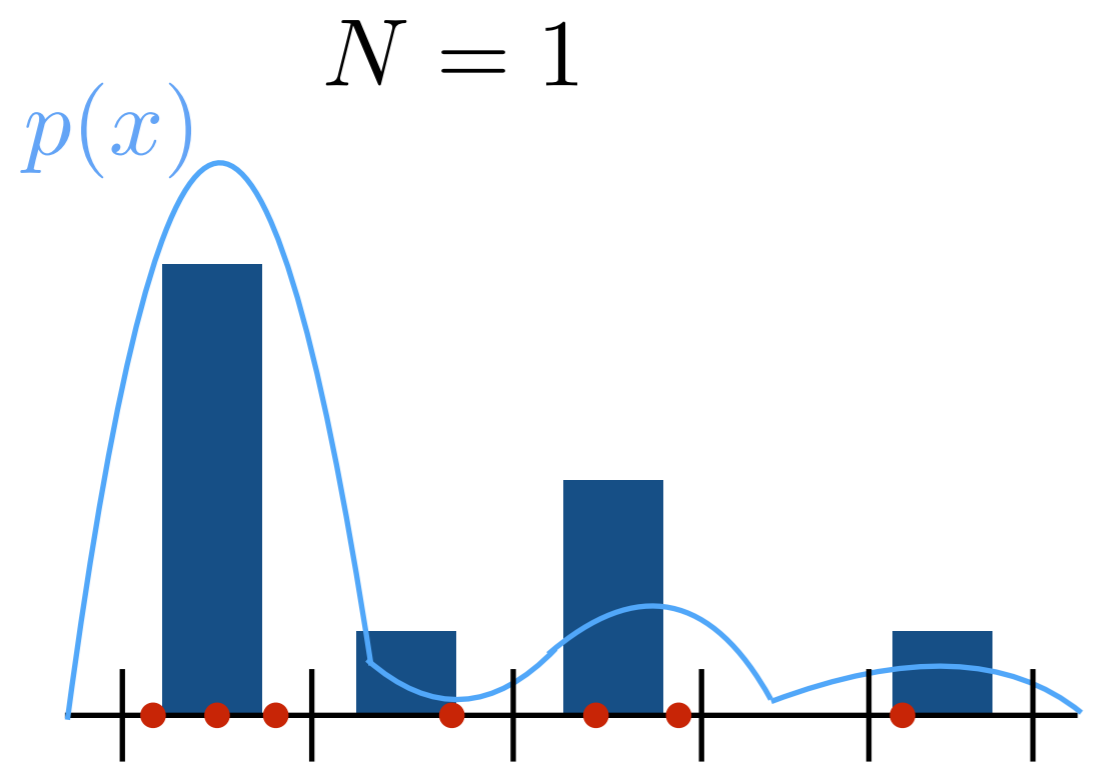
$$C_f = \int_{\mathbb{R}^D} \|\omega\|_1 |\hat{f}(\omega)| d\omega$$



Which color should be this circle?

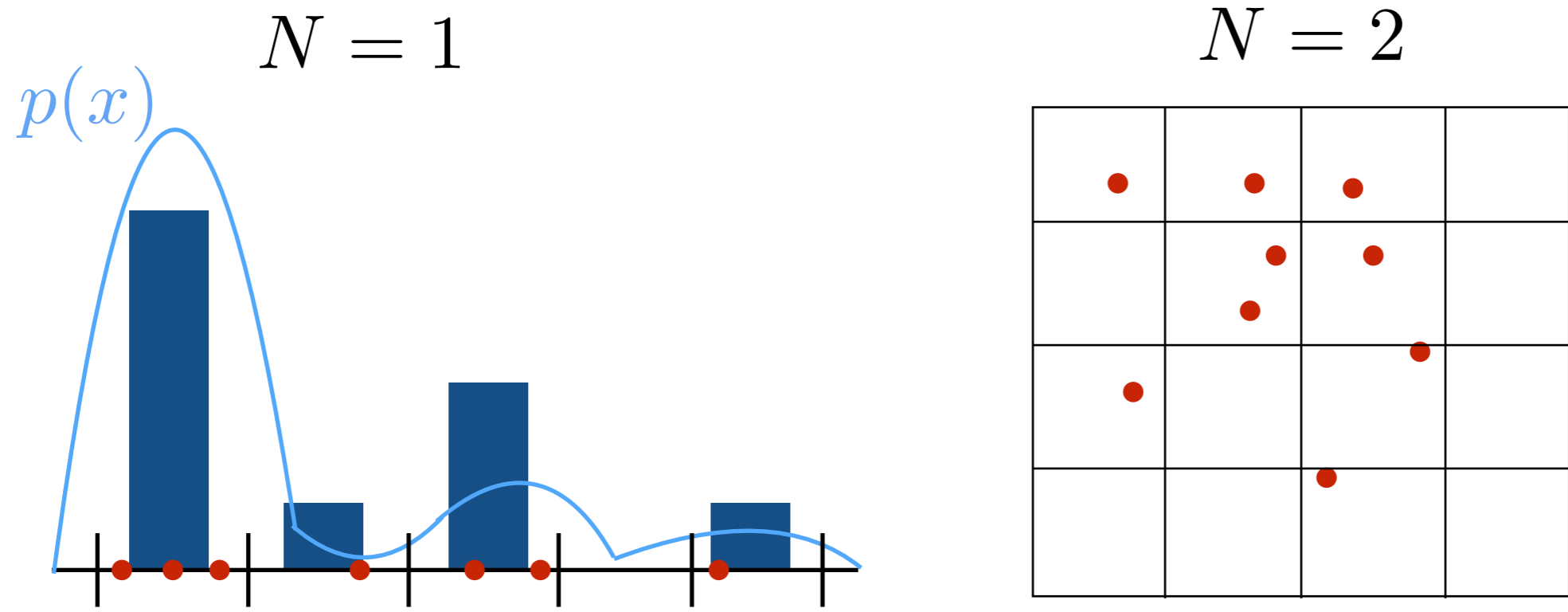
An example of supervised task: classification

- 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.

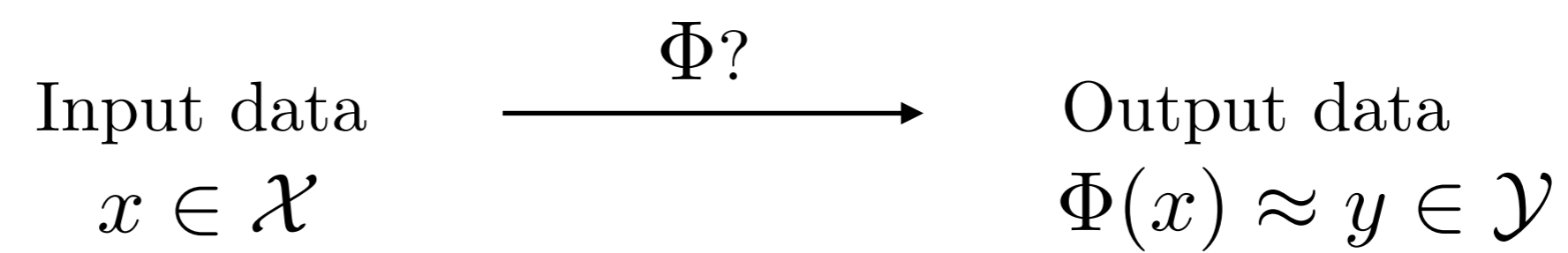
- Pdfs are difficult to estimate in high dimension.



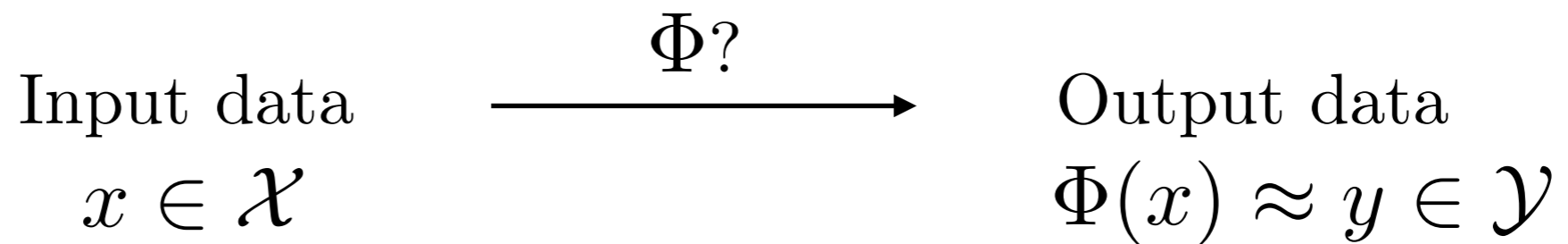
- For a fixed number of points and bin size, as N increases, the bins are likely to be empty.

Curse of dimensionality:
occurs in many machine learning problems

$\mathcal{X} = \mathbb{R}^2$ Samples space
 $\mathcal{Y} = \{\bullet, \bullet\}$ Labels



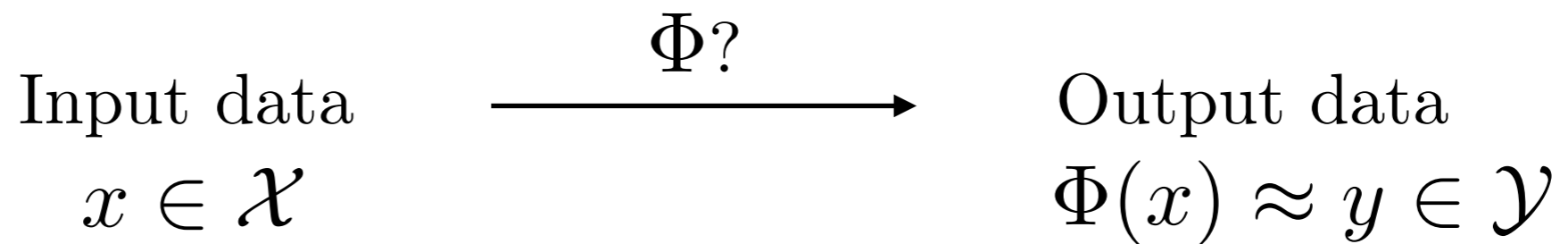
$\mathcal{X} = \mathbb{R}^2$ Samples space
 $\mathcal{Y} = \{\bullet, \bullet\}$ Labels



- 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.

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

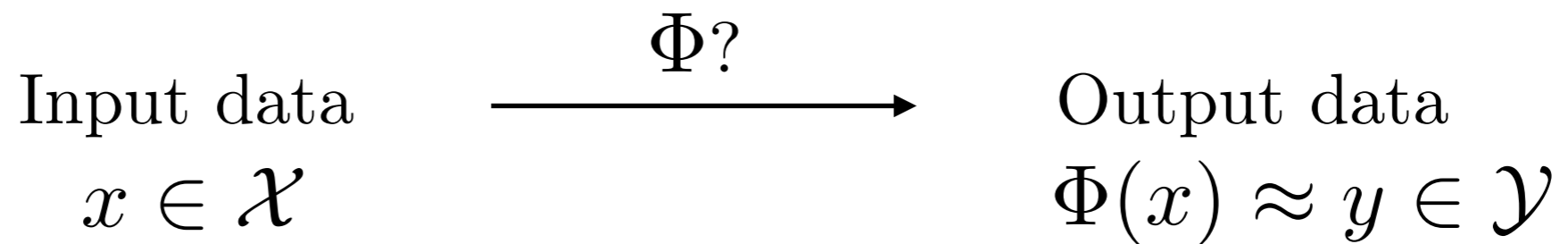
$\mathcal{Y} = \{\bullet, \bullet\}$ Labels



- 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,...

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

\mathcal{D} : dataset to construct Φ

- Supervised learning:

$$\mathcal{D} = \{(x, y)\}$$

- Unsupervised learning:

$$\mathcal{D} = \{(x)\}$$

- Semi-supervised learning:

$$\mathcal{D} = \{(x_1)\} \cup \{(x_2, y_2)\}$$

- Weakly-supervised learning:

$$\mathcal{D} = \{(x_1, y_1 + \epsilon_1)\} \cup \{(x_2)\}$$

- Self-supervised learning:

$$\mathcal{D} = \{(x(p), y(p))\}$$

often few gold data

- Multi-task, Transfer-learning:

$$\mathcal{D}_1 \rightarrow \mathcal{D}_2$$

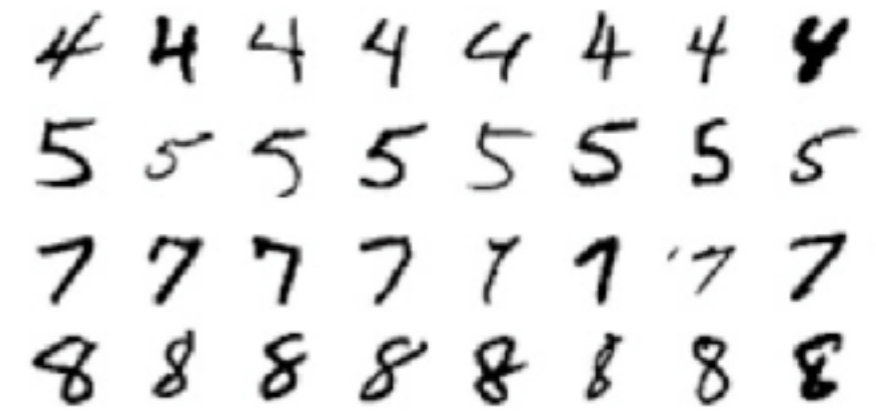
- How to address a supervised task:

- How to address a supervised task:
 1. Propose a model of your data.

- How to address a supervised task:

1. Propose a model of your data.

Ex.: MNIST (60k samples)



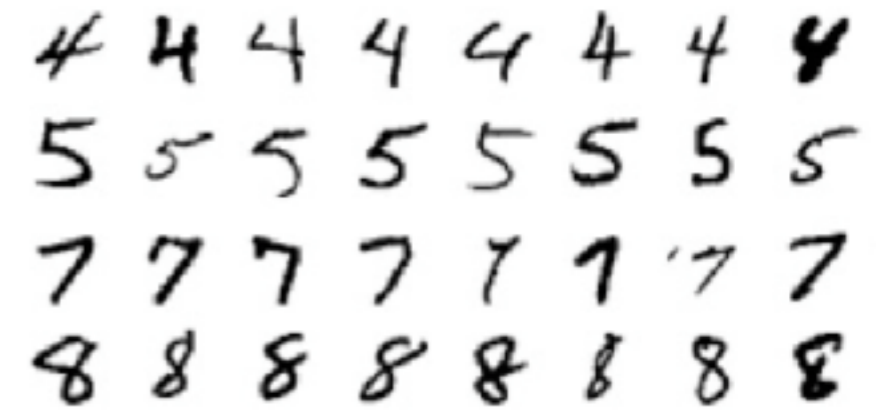
←
Small deformations
+ Translation

- How to address a supervised task:

1. Propose a model of your data.

Ex.: MNIST (60k samples)

2. Design a representation.



← Small deformations
+ Translation

- How to address a supervised task:

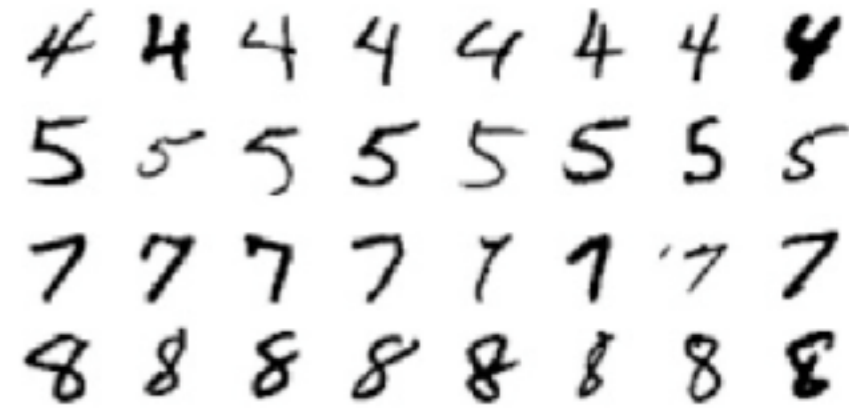
1. Propose a model of your data.

Ex.: MNIST (60k samples)

2. Design a representation.

Ex.: Scattering Transform.

Achieves translation invariance, linearises deformations.



←
Small deformations
+ Translation

- How to address a supervised task:

1. Propose a model of your data.

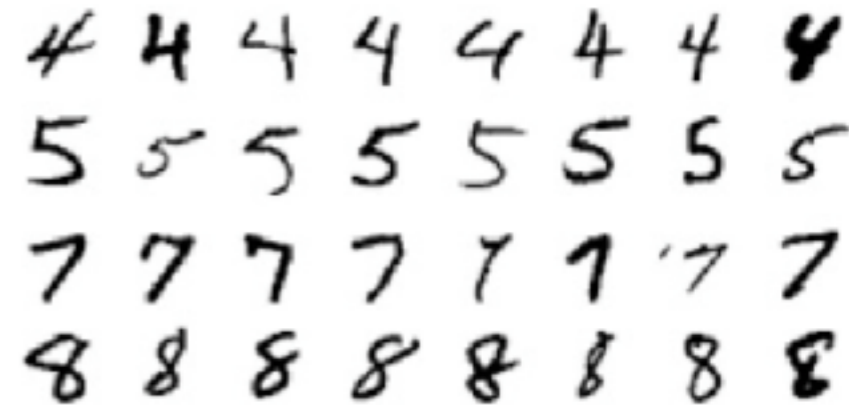
Ex.: MNIST (60k samples)

2. Design a representation.

Ex.: Scattering Transform.

Achieves translation invariance, linearises deformations.

3. Propose a (convex) classifier.

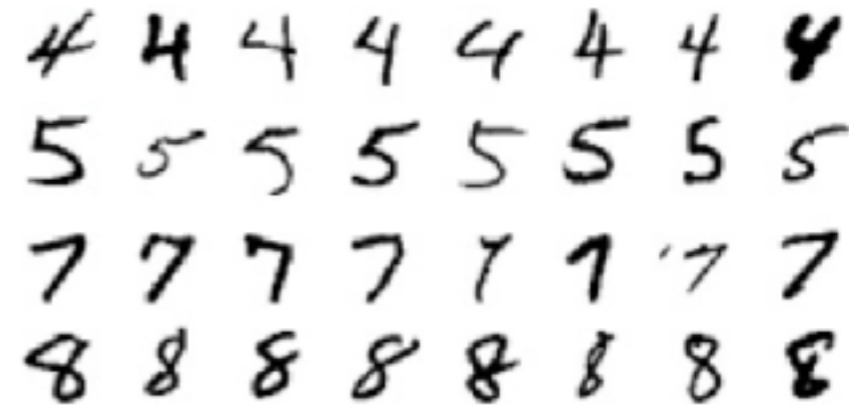


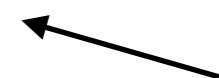
←
Small deformations
+ Translation

- How to address a supervised task:

- Propose a model of your data.

Ex.: MNIST (60k samples)




 Small deformations
+ Translation

- Design a representation.

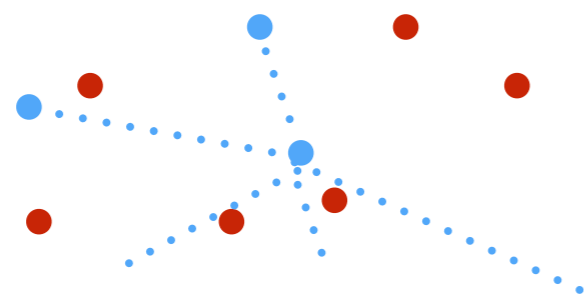
Ex.: Scattering Transform.

Achieves translation invariance, linearises deformations.

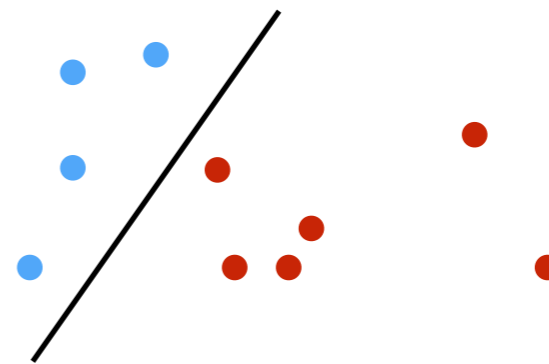
- Propose a (convex) classifier.

Ex.: Linear SVM.

... Displacement



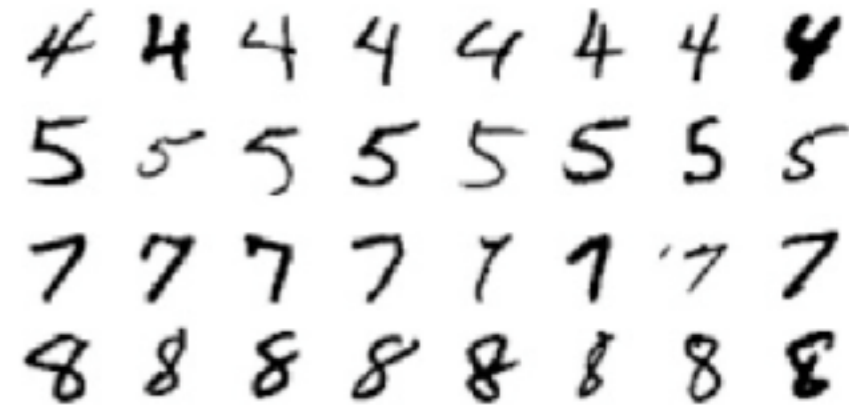
Φ
 + projection



- How to address a supervised task:

1. Propose a model of your data.

Ex.: MNIST (60k samples)



2. Design a representation.

Ex.: Scattering Transform.

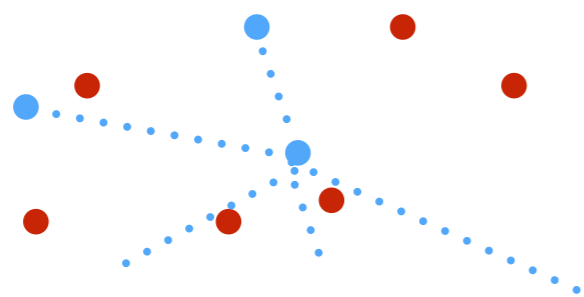
Achieves translation invariance, linearises deformations.

← Small deformations
+ Translation

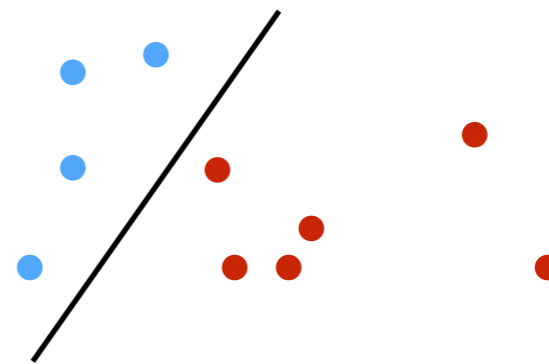
3. Propose a (convex) classifier.

Ex.: Linear SVM.

... Displacement



Φ
+ projection



4. Obtain reasonable performances.

In the following...

1. No model known on real images

1. No model known on real images
2. Limited *a priori*, except translation invariance

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

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

Ref.: image-net.org

- ImageNet 2012: (350GB)
 1 million training images, 1 000 classes
 400 000 test images
 Large coloured images of various sizes

Ref.: image-net.org



- 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)

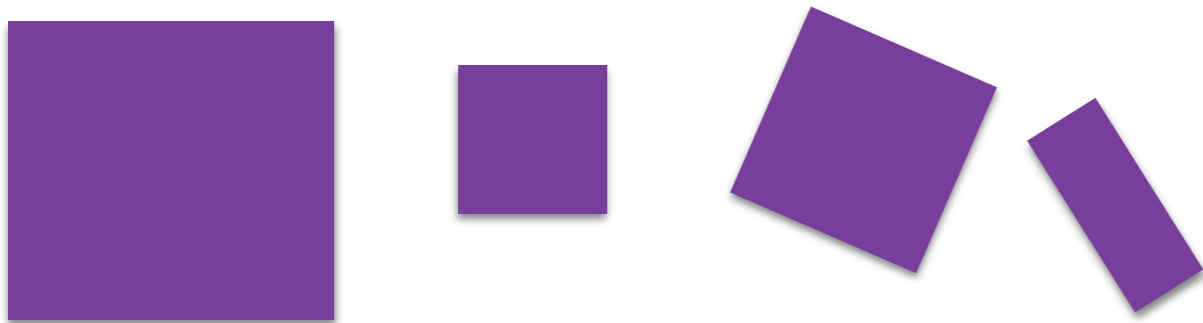
Ref.: image-net.org



Image variabilities

Geometric variability

Groups acting on images:
translation, rotation, scaling



Other sources : luminosity, occlusion,
small deformations

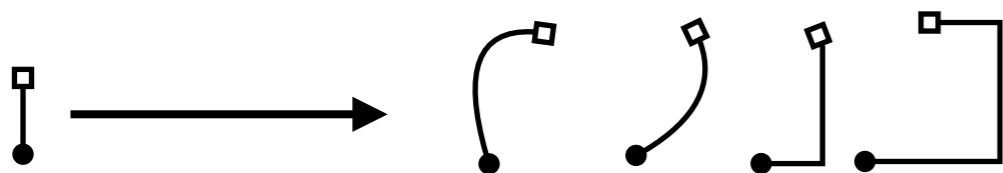
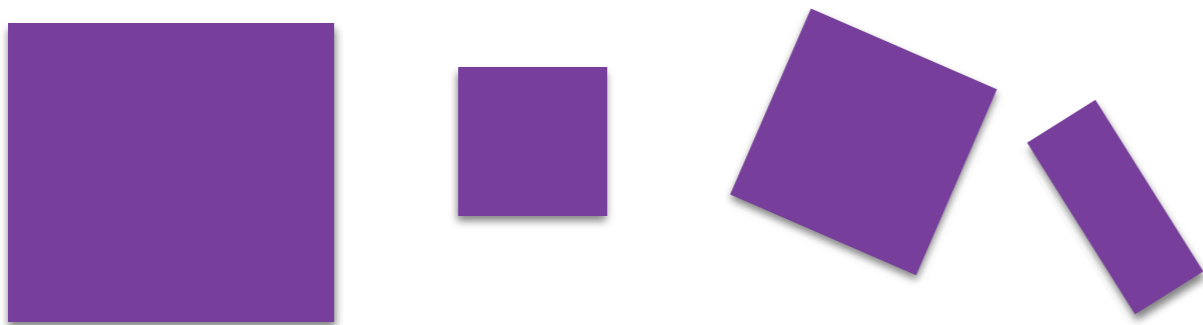


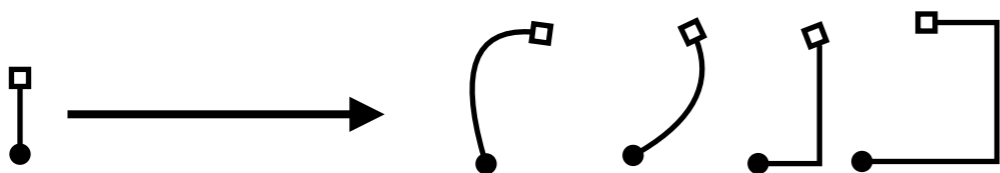
Image variabilities

Geometric variability

Groups acting on images:
translation, rotation, scaling



Other sources : luminosity, occlusion,
small deformations



Class variability

Intraclass variability

Not informative



Extraclass variability

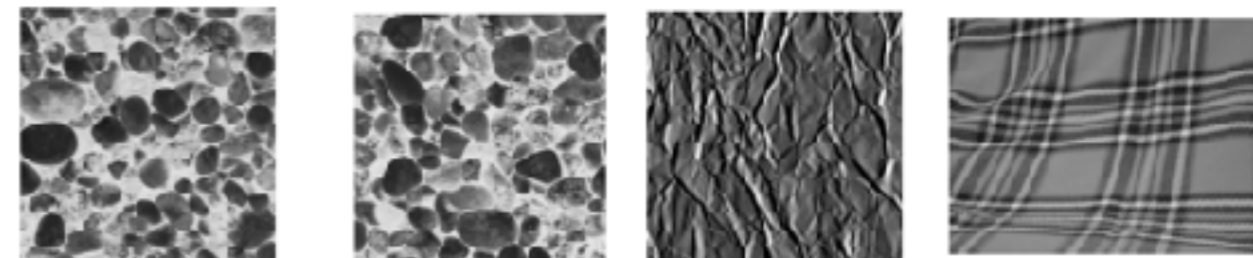
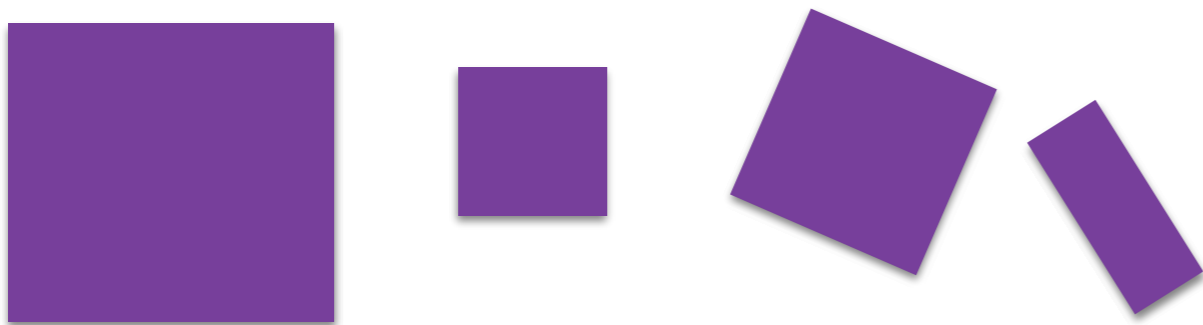


Image variabilities

Geometric variability

Groups acting on images:
translation, rotation, scaling



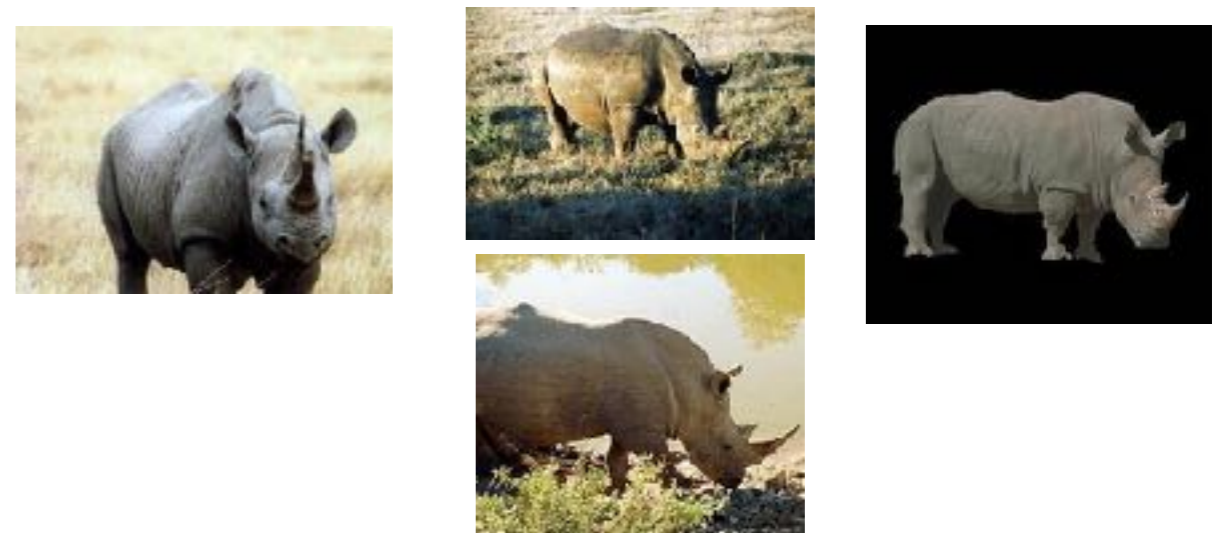
Other sources : luminosity, occlusion,
small deformations



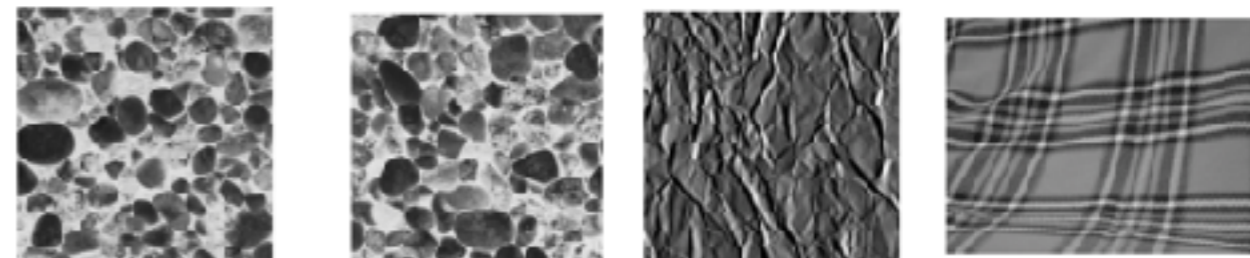
Class variability

Intraclass variability

Not informative



Extraclass variability



High variance: hard to reduce!

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

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

- **Stability** to noise

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

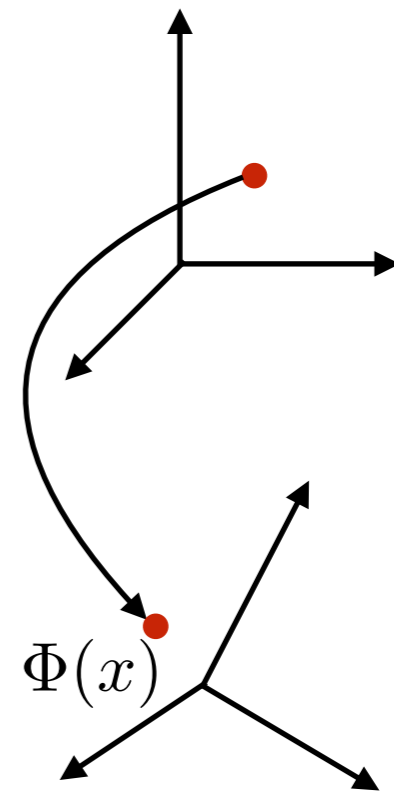
- **Reconstruction** properties

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

- **Linear separation** of the different classes

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

$$\forall i, \sigma(\Phi(X_i)) \ll 1$$



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

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

- **Stability** to noise

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

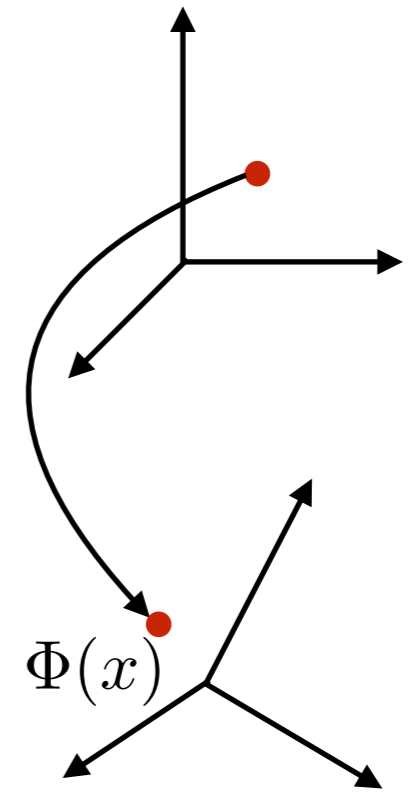
- **Reconstruction** properties

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

- **Linear separation** of the different classes

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

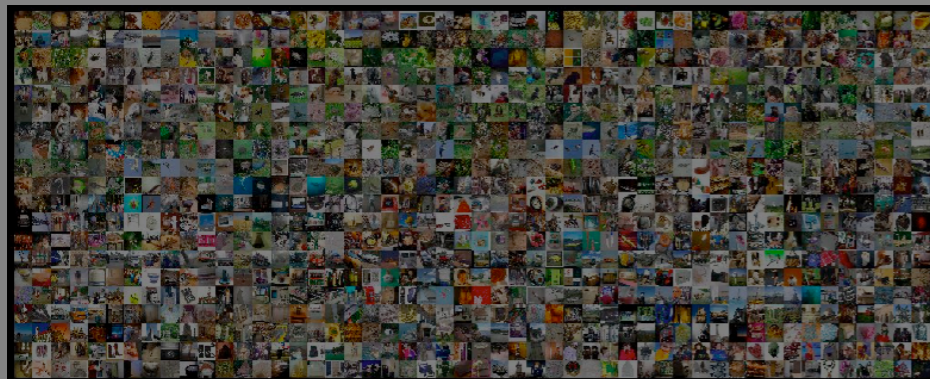
$$\forall i, \sigma(\Phi(X_i)) \ll 1 \quad \text{Can be difficult to handcraft..}$$



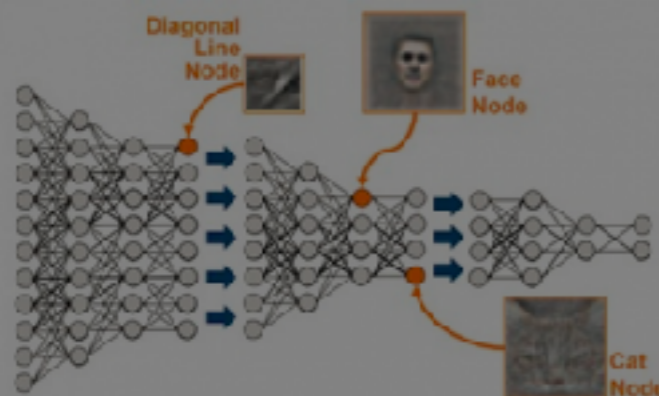
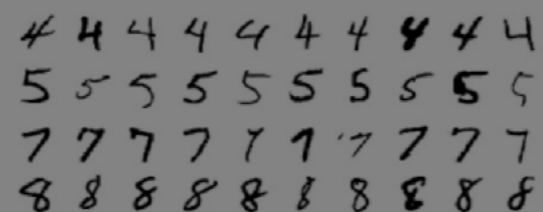
Is this solvable?

Years of research...

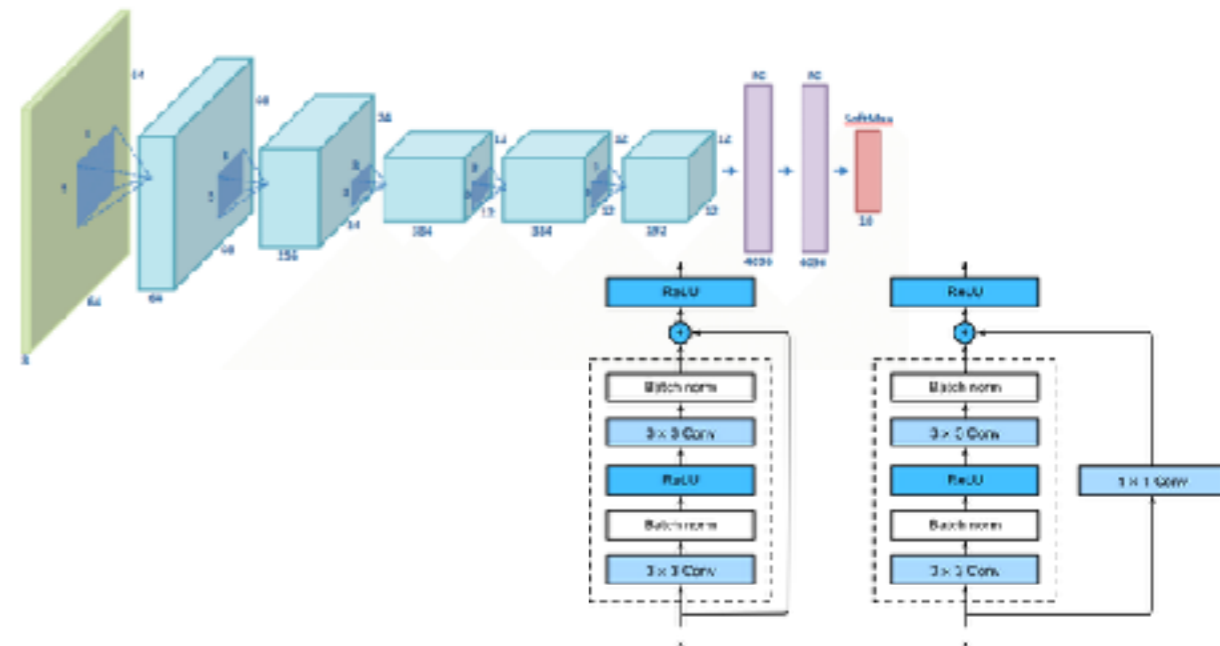
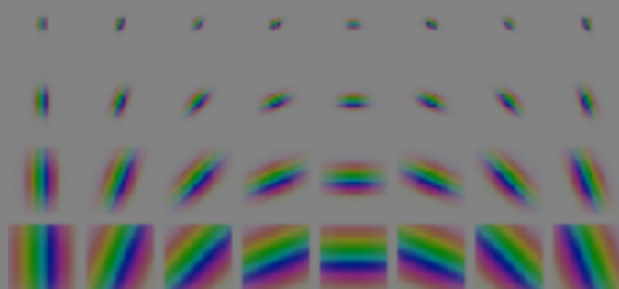




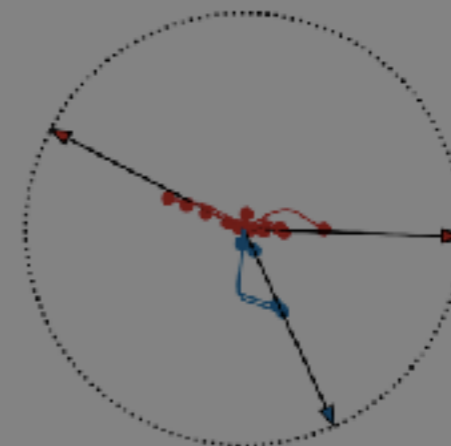
Introduction to image classification



Interpretability in Deep Learning



Fighting the curse of dimensionality with Deep Neural Networks



Statistical learning results

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

Solving high-dimensional tasks with deep learning



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



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

- *Solve* several high dimensional problems that seemed intractable. Impressive benchmarks.



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

- *Solve* several high dimensional problems that seemed intractable. Impressive benchmarks.
- Requires a *huge* amount of labeled data



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

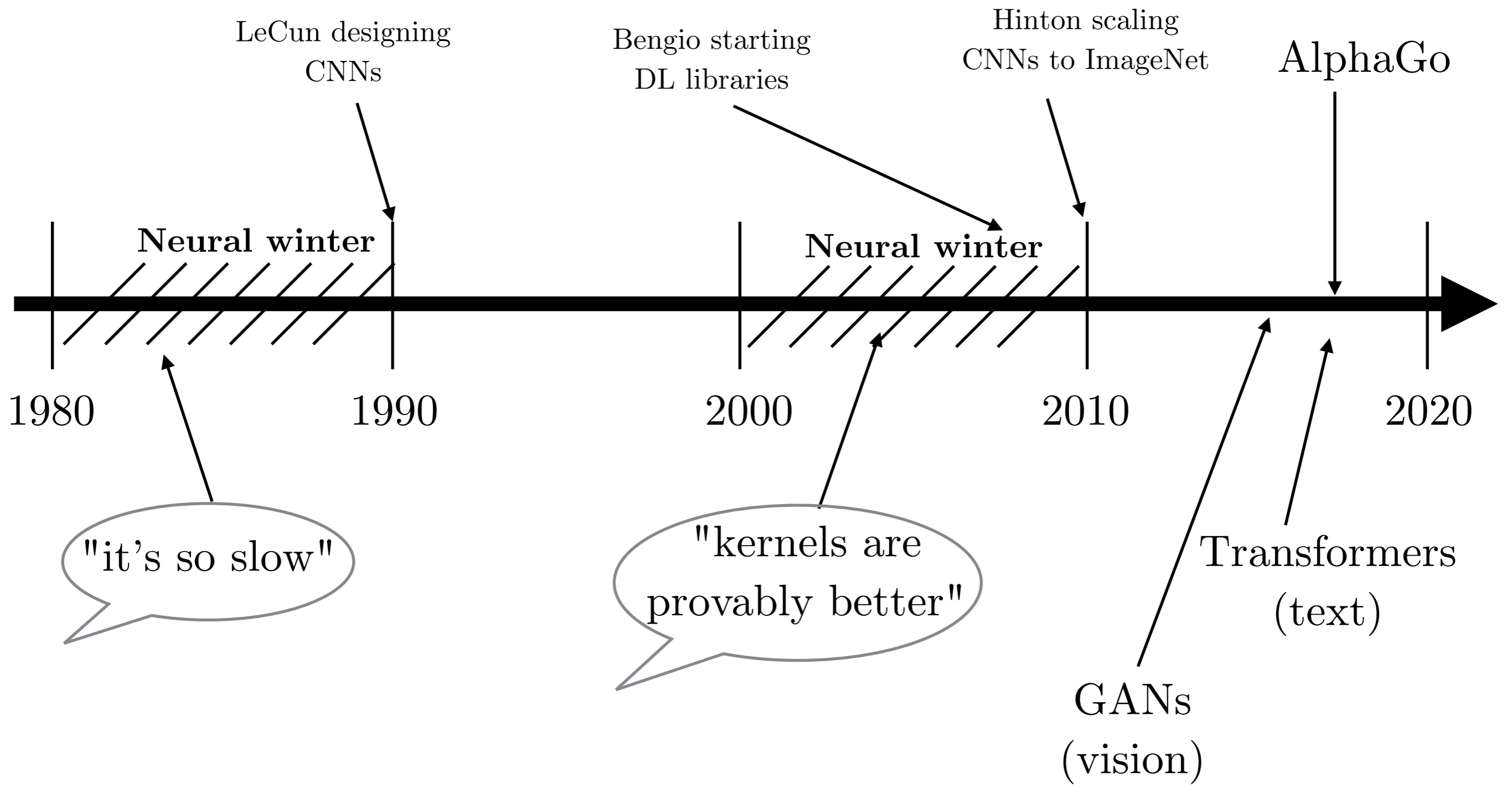
- *Solve* several high dimensional problems that seemed intractable. Impressive benchmarks.
- Requires a *huge* amount of labeled data
- *Generic* and *simple* to deploy (present in many final products) / requires a *large* expertise (highly demanded profiles)



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

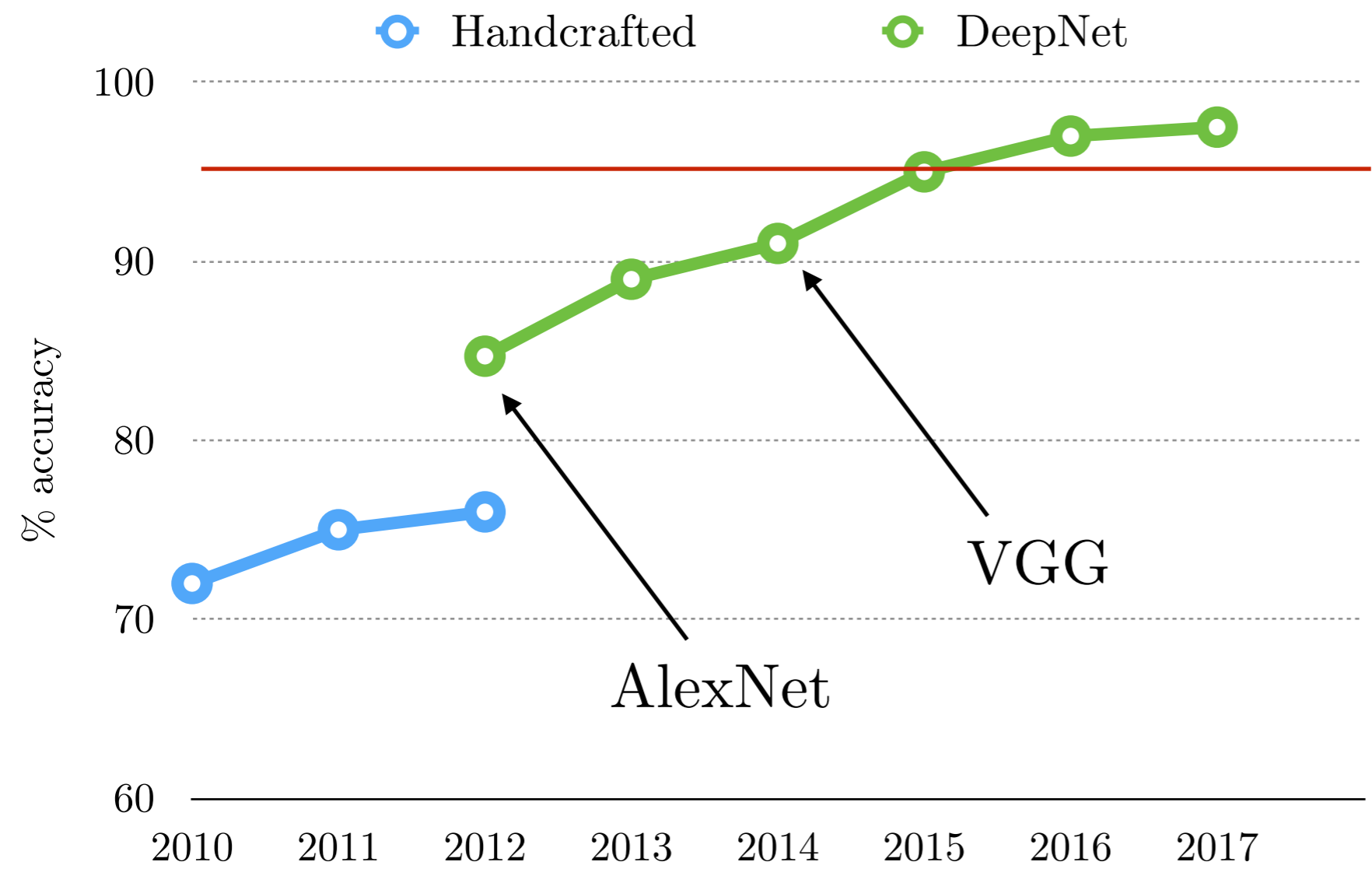
- *Solve* several high dimensional problems that seemed intractable. Impressive benchmarks.
- Requires a *huge* amount of labeled data
- *Generic* and *simple* to deploy (present in many final products) / requires a *large* expertise (highly demanded profiles)
- Handcrafted features are *not required*: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

A biased history of Deep Learning



- Accuracies!

• Accuracies!

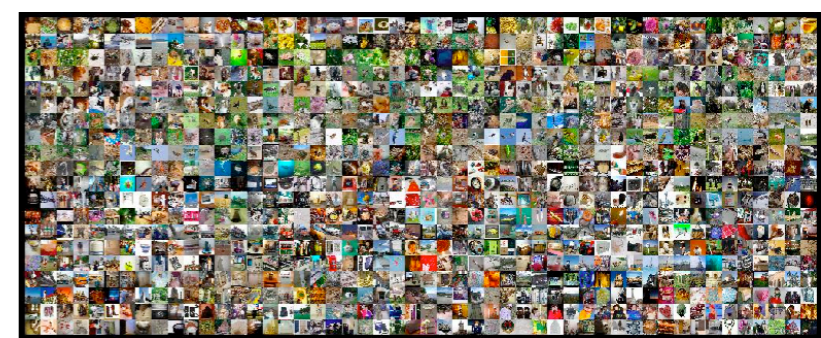
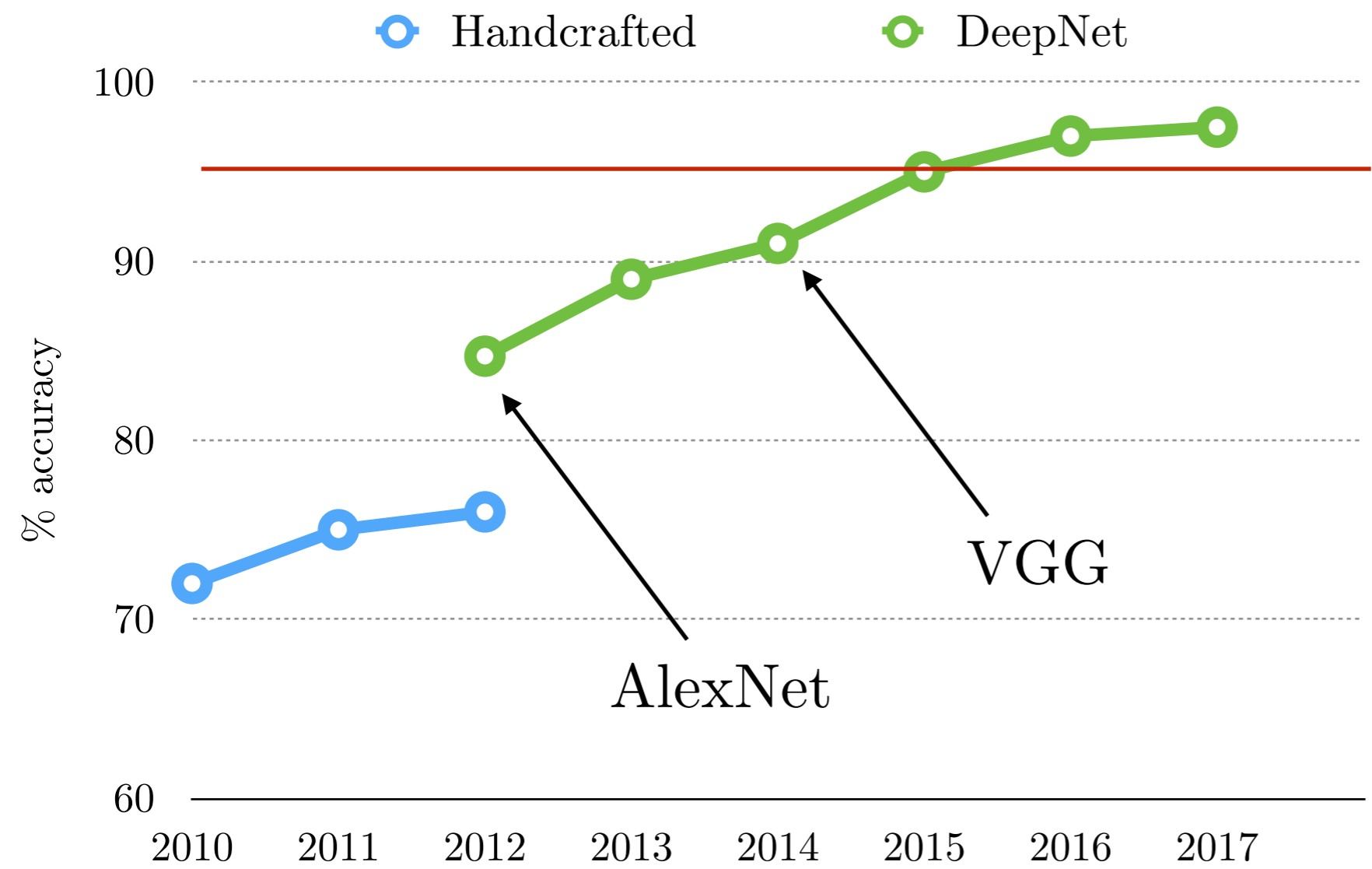


ImageNet:

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

top5 - ImageNet

• Accuracies!



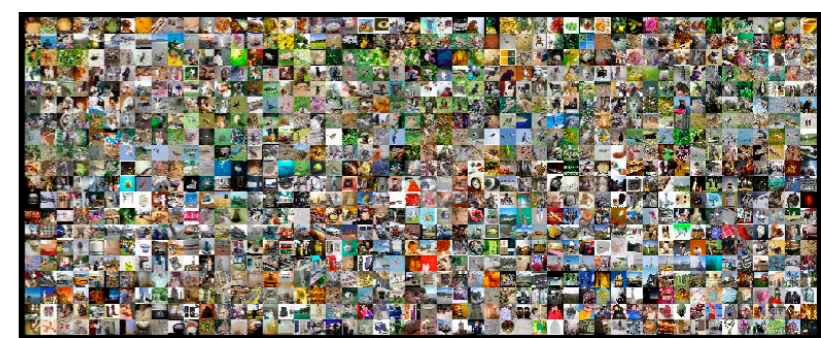
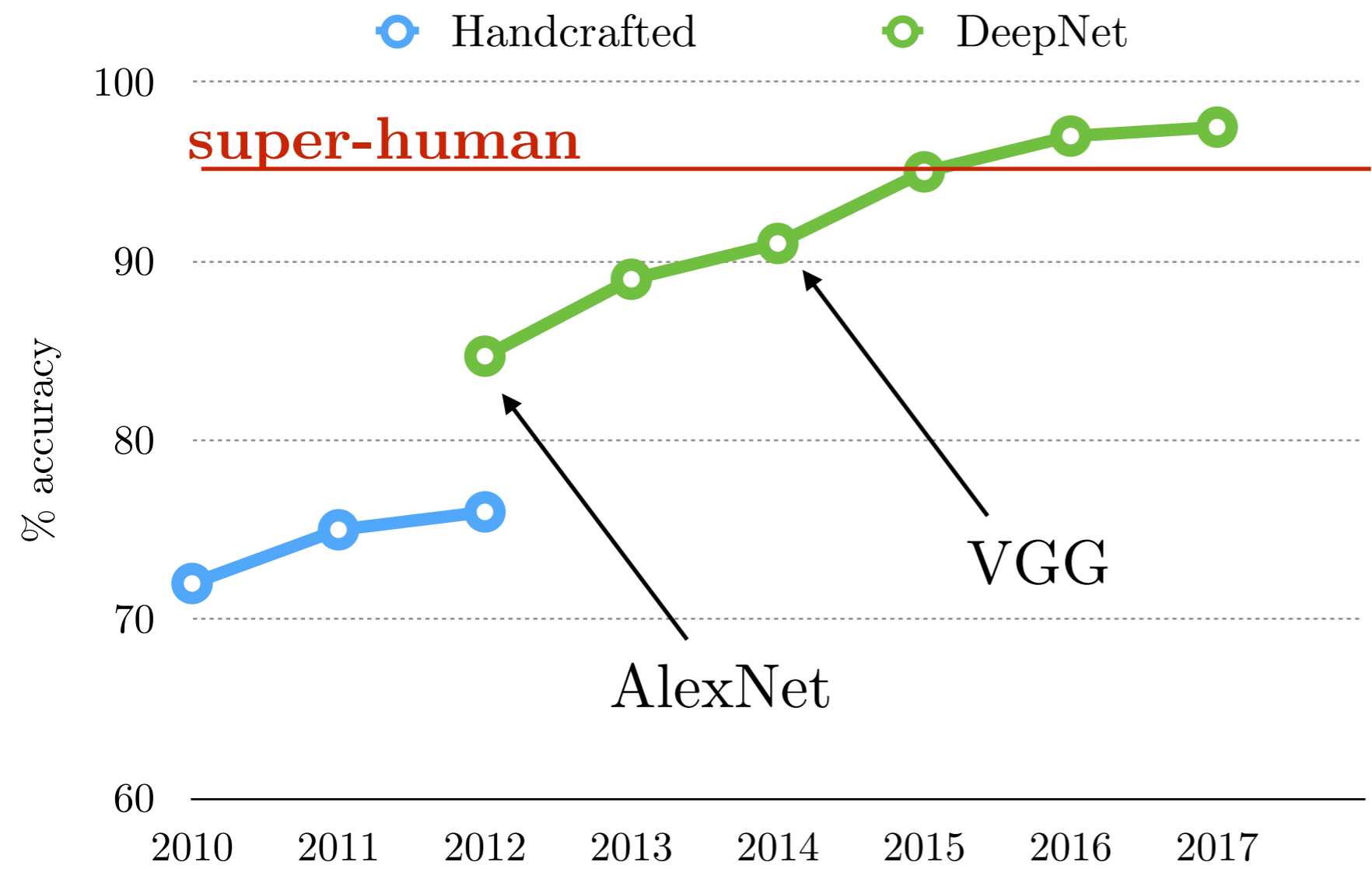
ImageNet:

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

top5 - ImageNet

Theory for good performances?

- Accuracies!

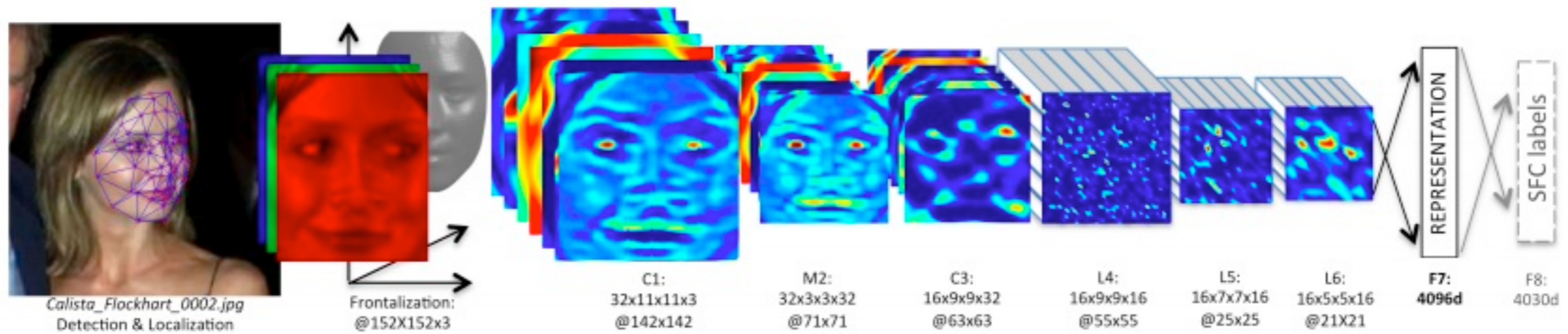


ImageNet:

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

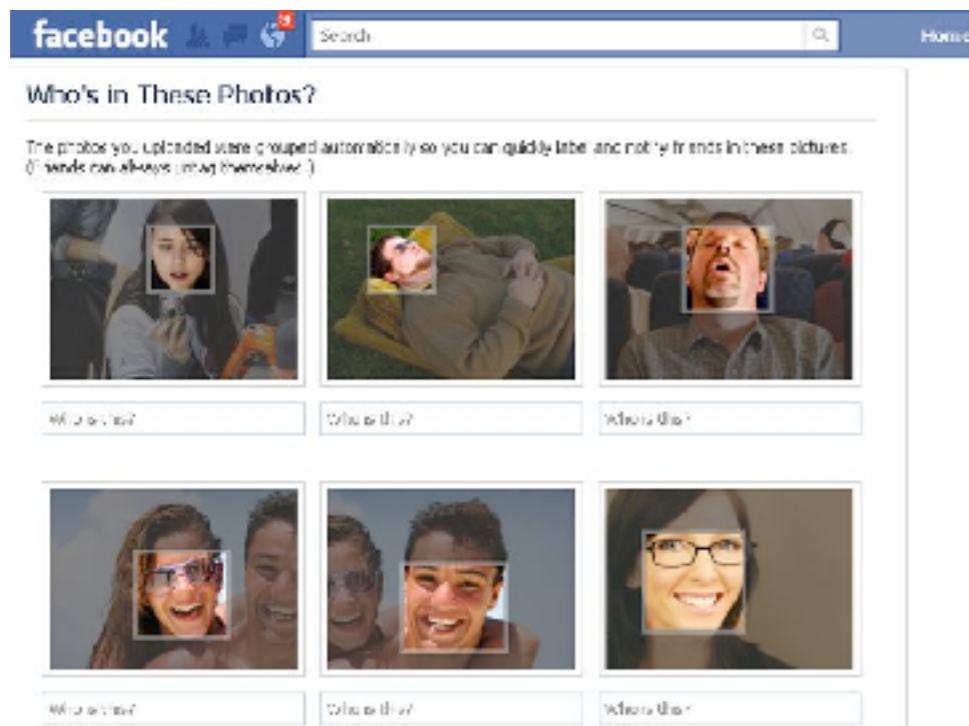
top5 - ImageNet

Theory for good performances?



et al.

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

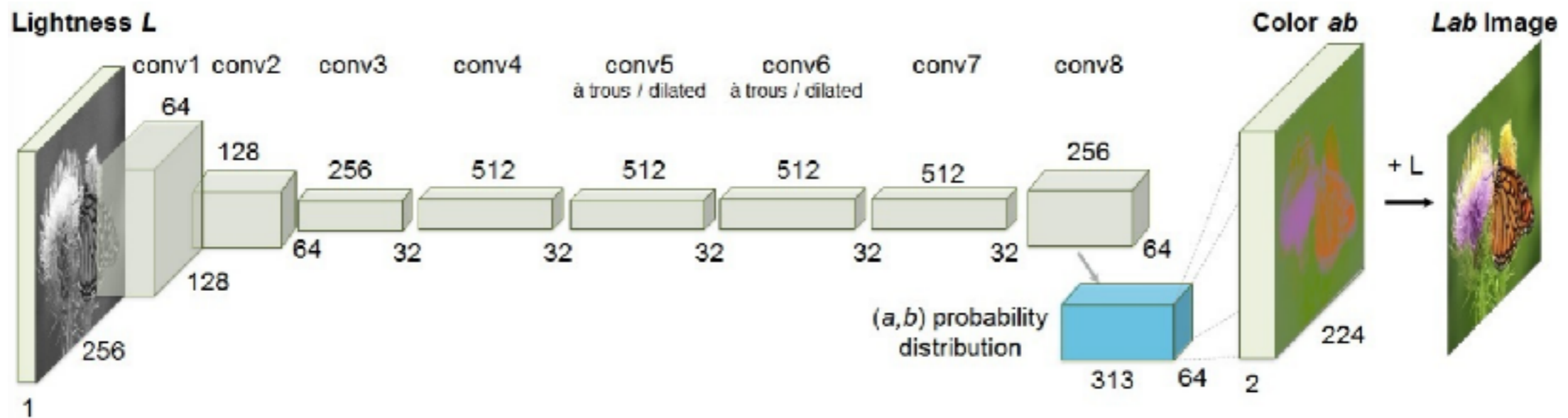


Ref.: DeepFace: Closing the Gap to Human-Level Performance in Face Verification
Taigman et al.





Colorful Image Colorization, Zhang et al.



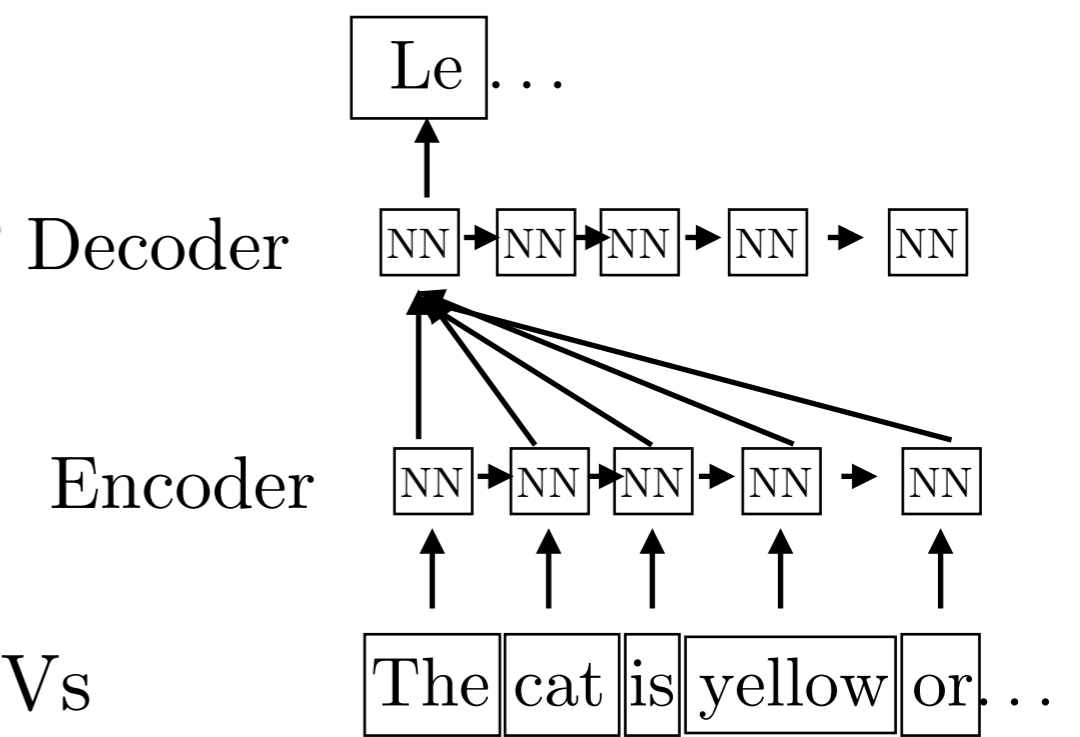
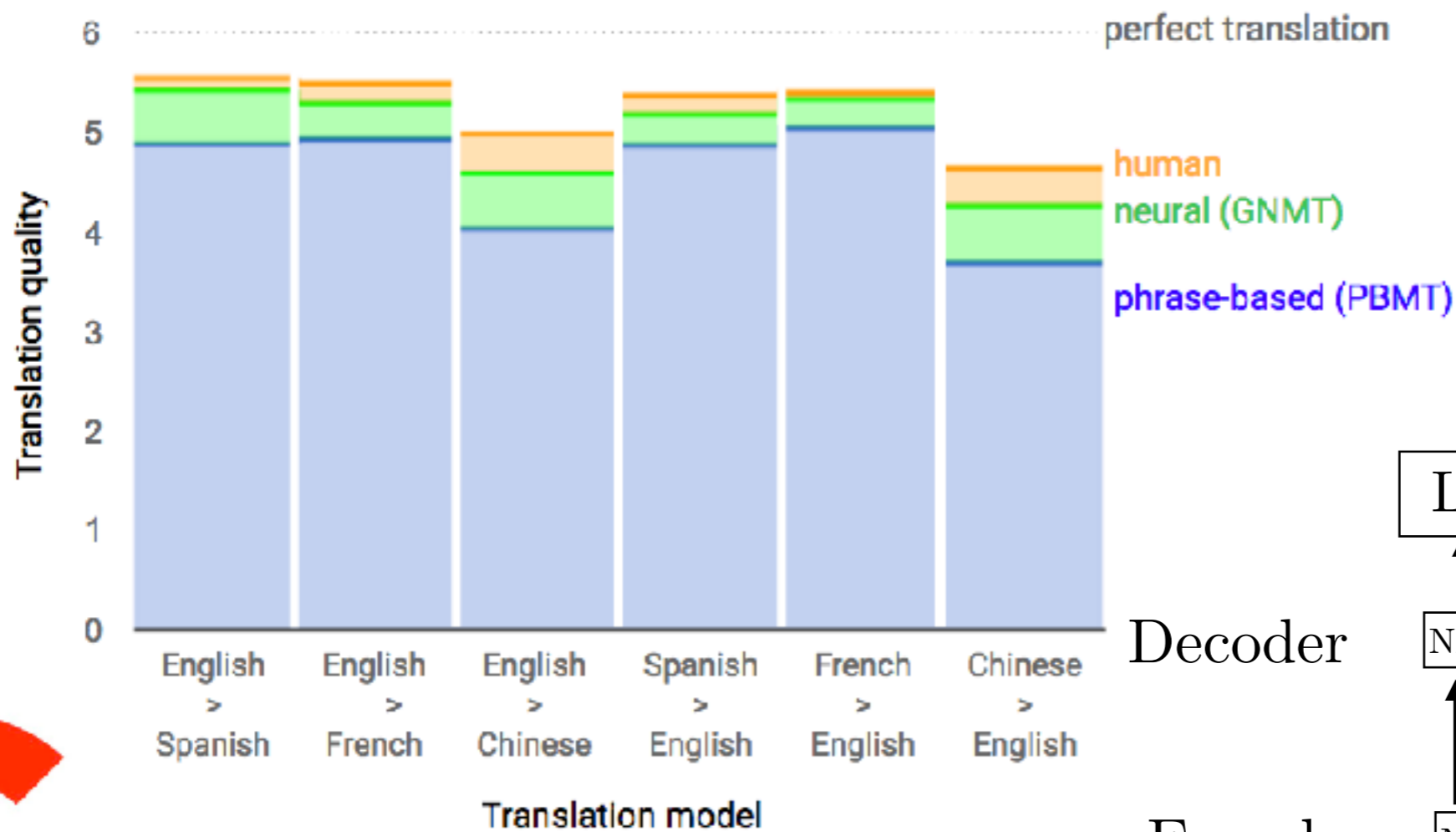
Coloring an image by hand takes several weeks



Spectacular results in face generation.

in text understanding/translations

- Translation (Google uses Recurrent Neural Networks):



Applications for HR: sorting CVs

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;
}

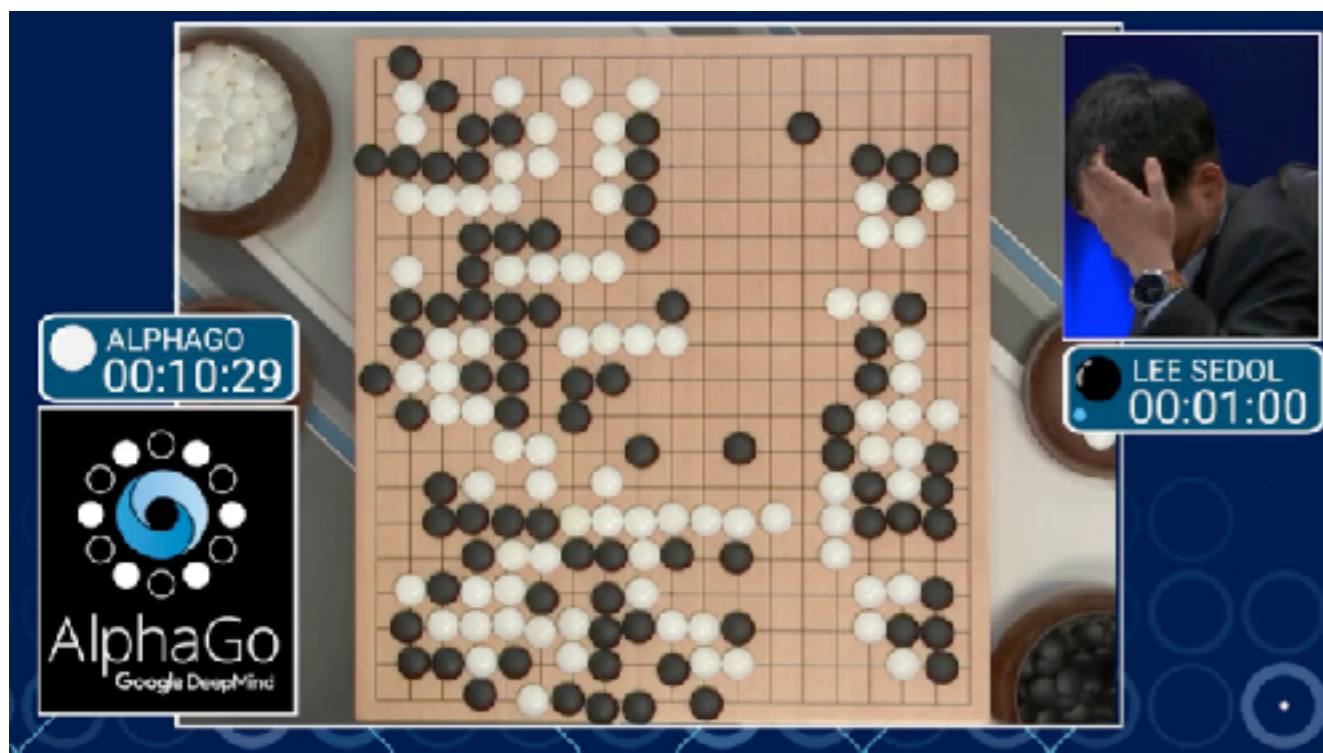
```

Real one?

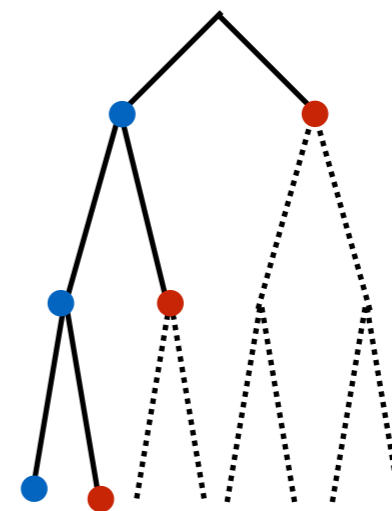
Outstanding results with 25 Game Strategy

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...



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

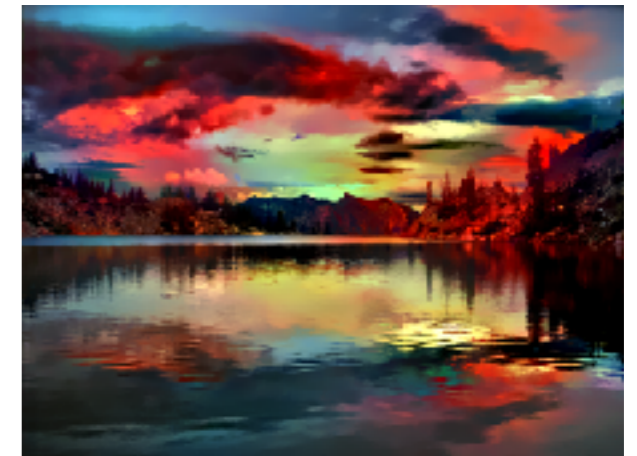
Output
 $\Phi \tilde{x}$



+



=



+



=



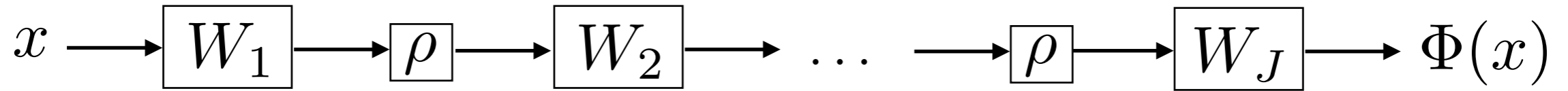
Ref.: Deep Photo Style Transfer, Luan et al.

Direct applications in Web design...

**A highly non-convex
and difficult optimization to
train a model**

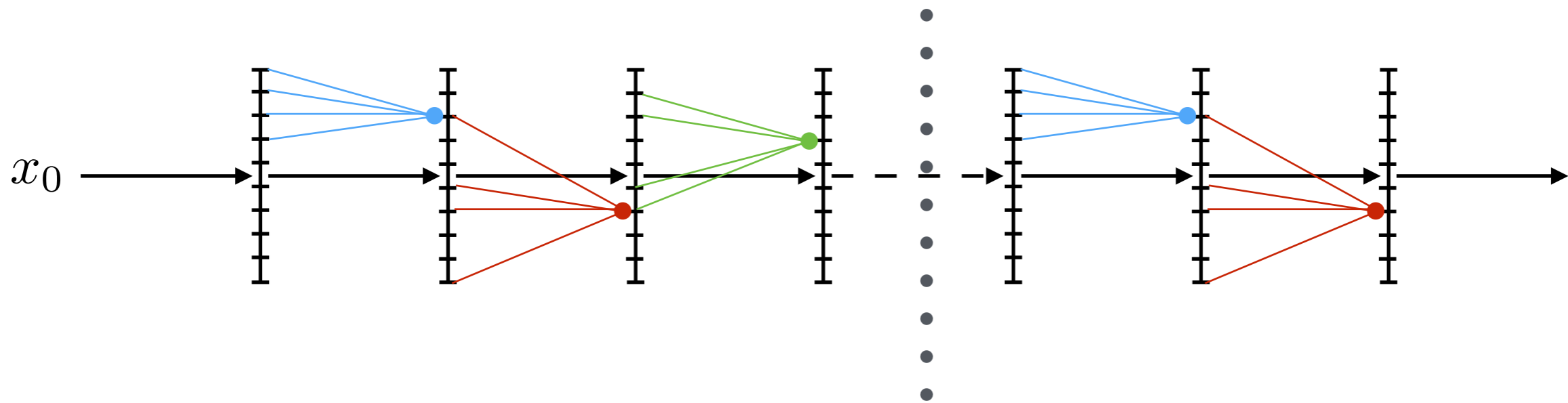
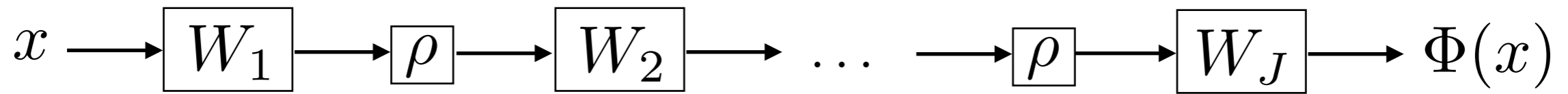
input signal

output signal



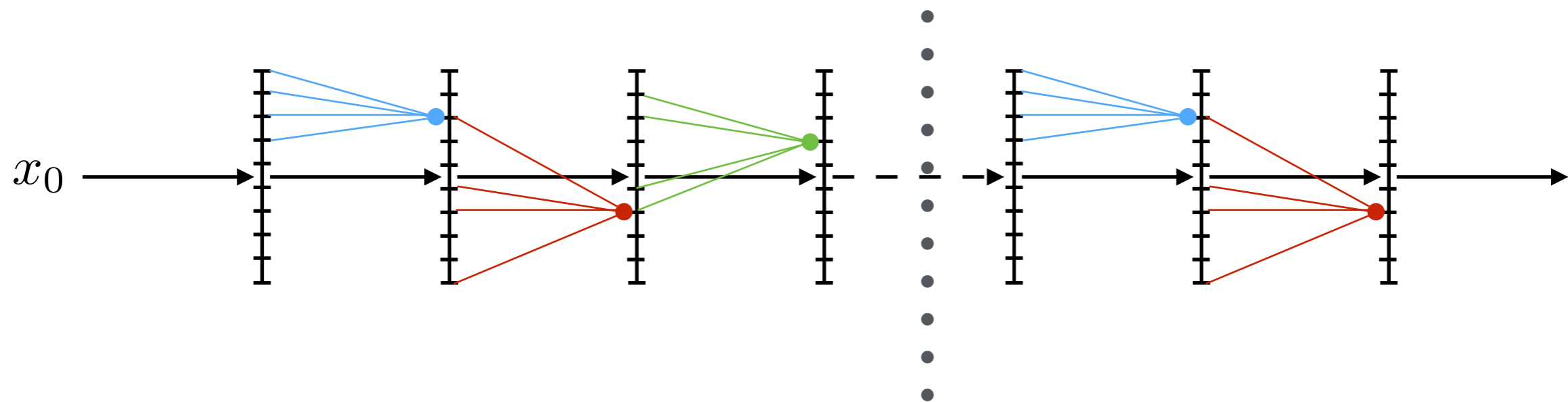
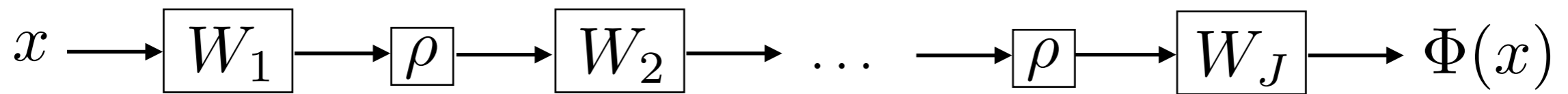
input signal

output signal



input signal

output signal

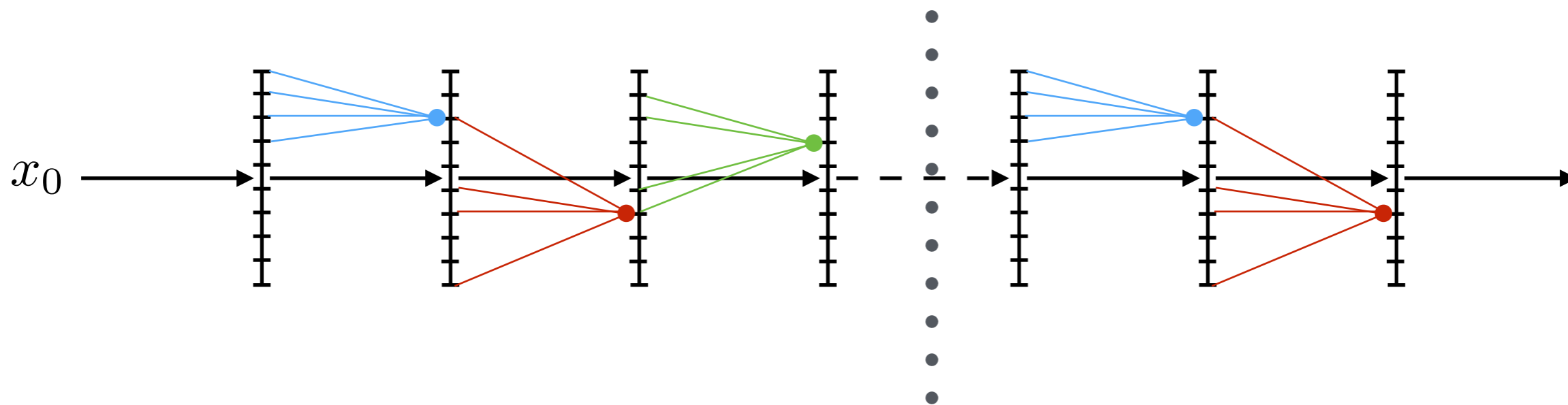
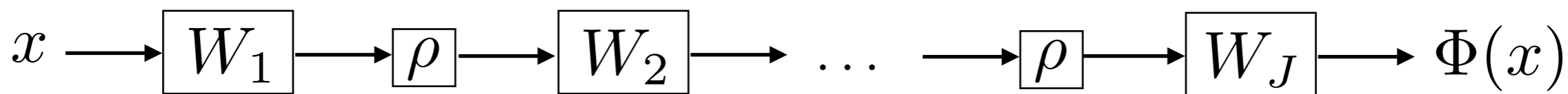


$$x_{j+1} = \rho W x_j \longrightarrow x_{j+1,k} = \rho \left(\sum_i W_{j,ki} x_{j,i} \right)$$

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

input signal

output signal



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

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

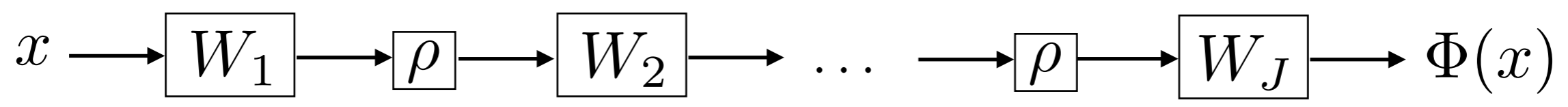
Note that $\Phi(x; W_1, \dots, W_J)$ is non-convex in x or each W_j

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

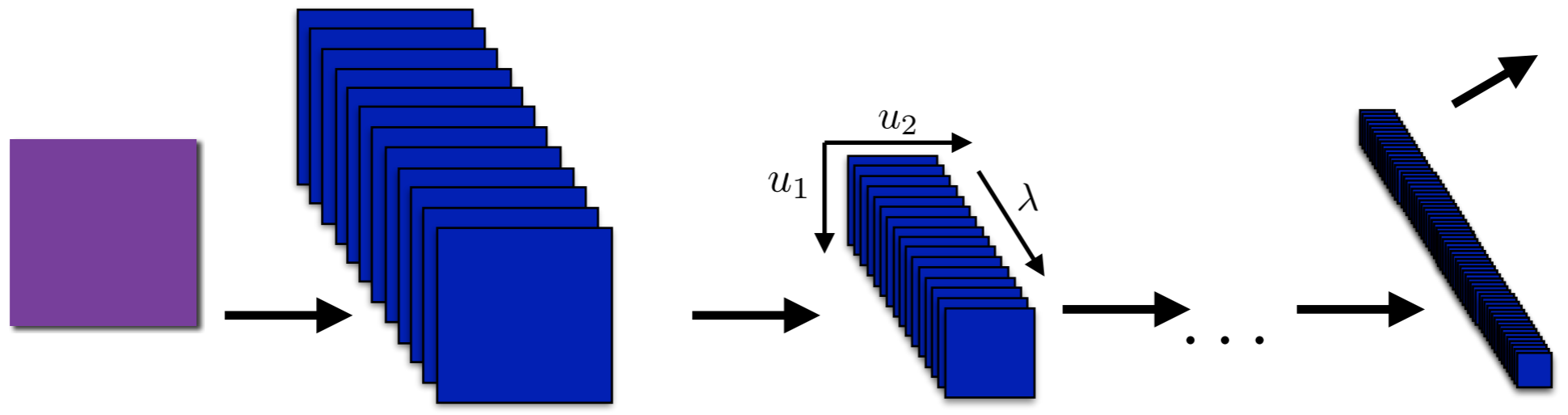
Networks

input signal

output signal



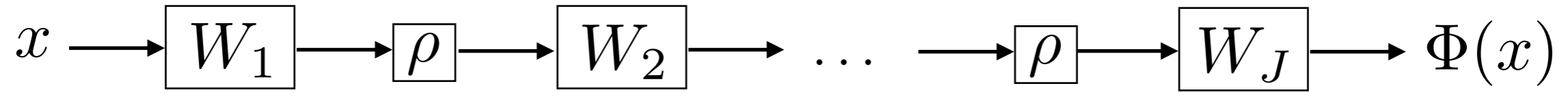
Schematic



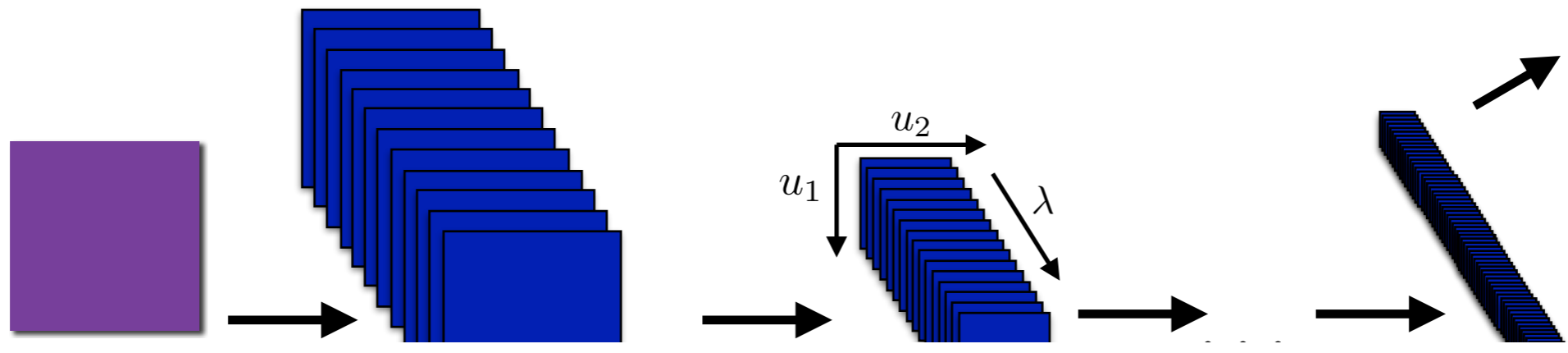
Networks

input signal

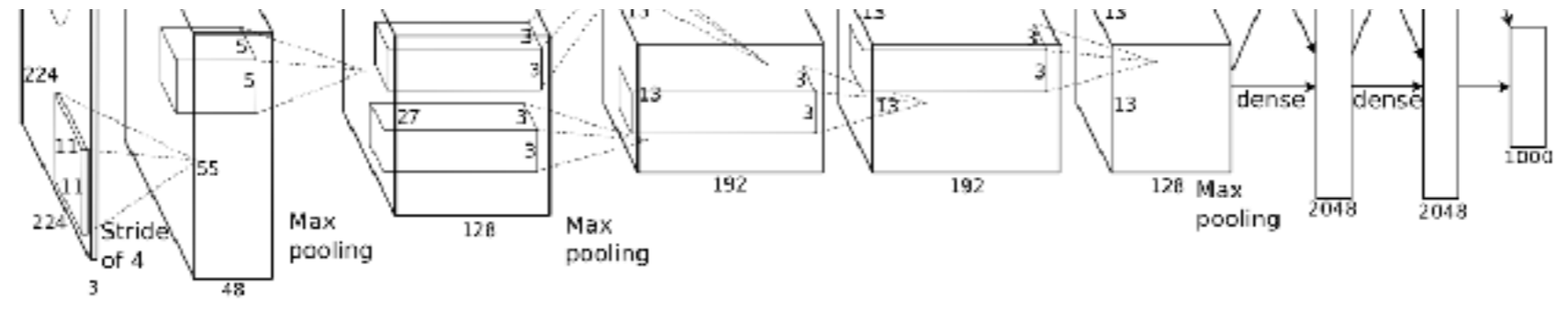
output signal



Schematic



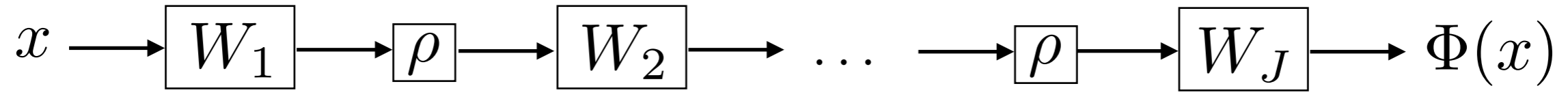
Engineering



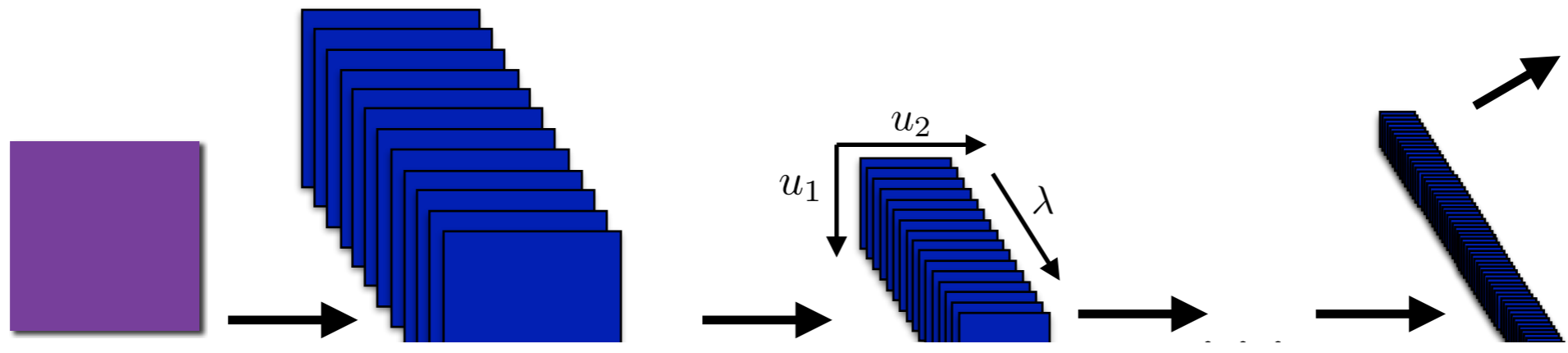
Networks

input signal

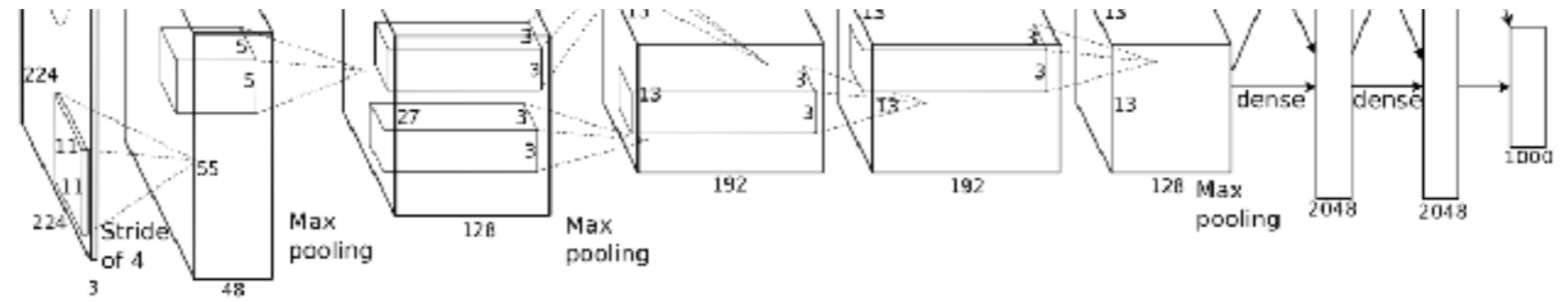
output signal



Schematic



Engineering



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

learned kernel

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

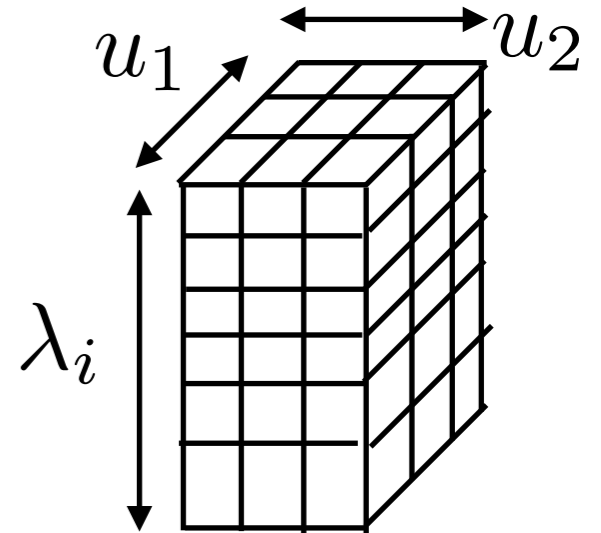
Sometimes some "pooling" are incorporated, mainly for speed purposes.

Again, this leads to a non convex loss.

Ref.: Signal Processing Tour, Mallat 1999

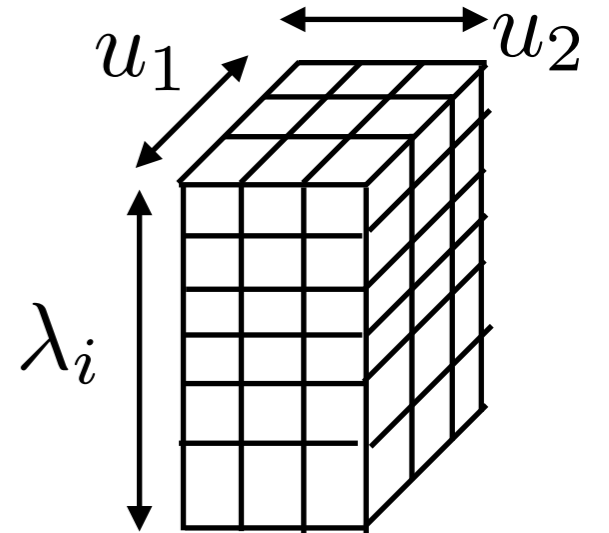
- Very often, the filters of a CNN have a small support (3x3) and are interlaced with downsampling.

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

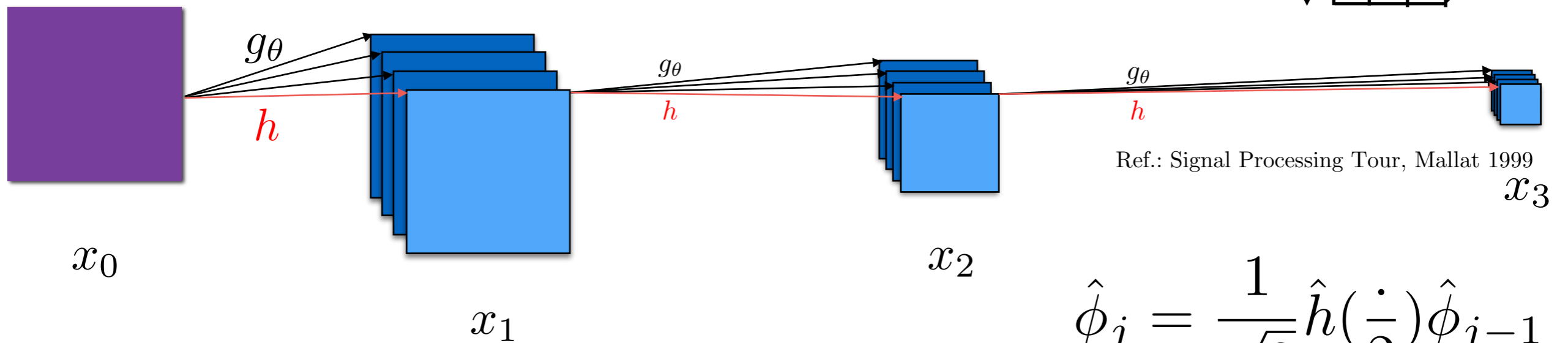


- Very often, the filters of a CNN have a small support (3x3) and are interlaced with downsampling.

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



- Similar to a Wavelet Transform.



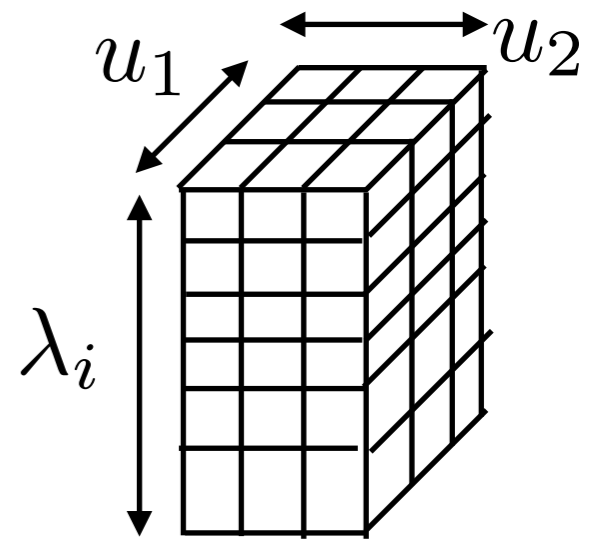
Ref.: Signal Processing Tour, Mallat 1999

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

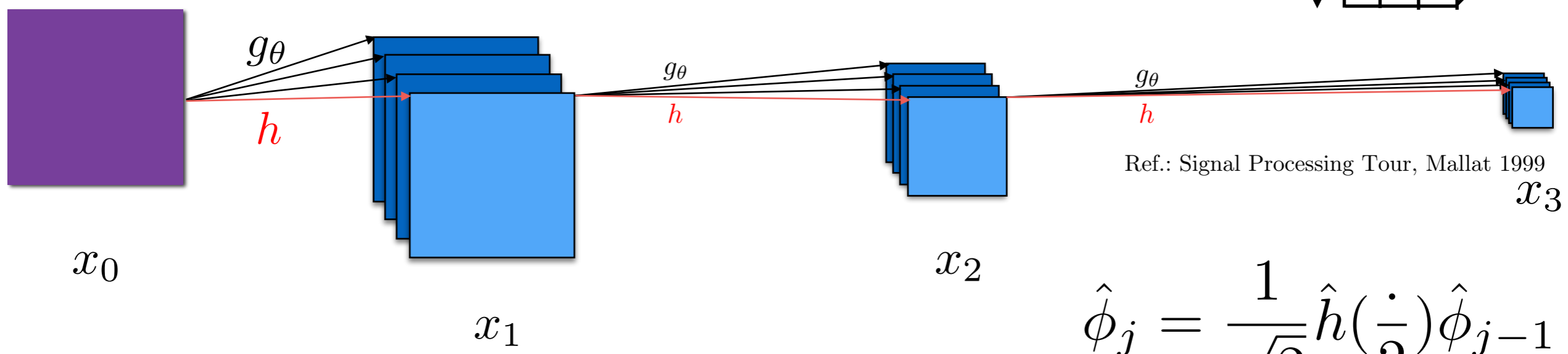
$$\hat{\psi}_{j,\theta} = \frac{1}{\sqrt{2}} \hat{g}_\theta\left(\frac{\cdot}{2}\right) \hat{\phi}_{j-1}$$

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

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



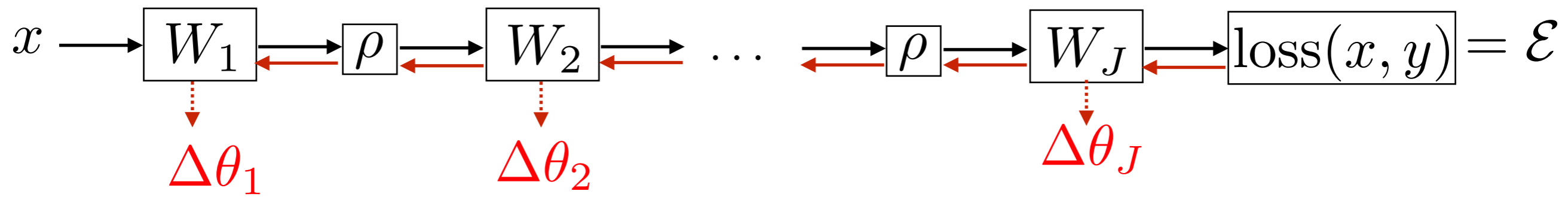
- Similar to a Wavelet Transform.



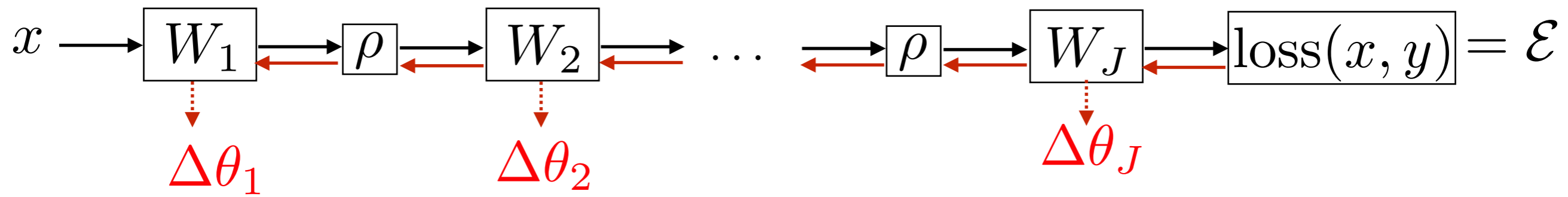
$$\hat{\phi}_j = \frac{1}{\sqrt{2}} \hat{h}\left(\frac{\cdot}{2}\right) \hat{\phi}_{j-1}$$

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

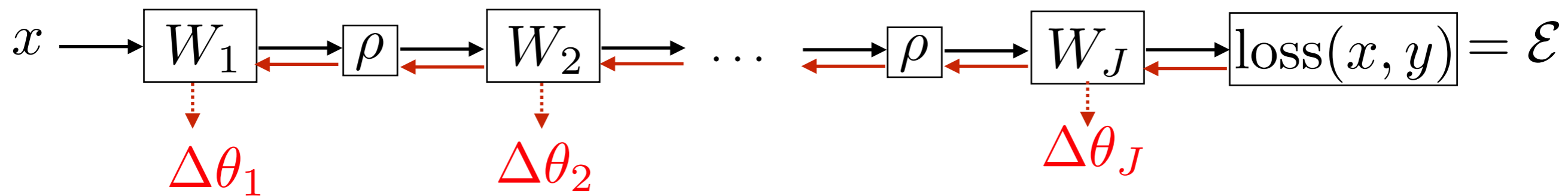
- Except the sum isn't separable.



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

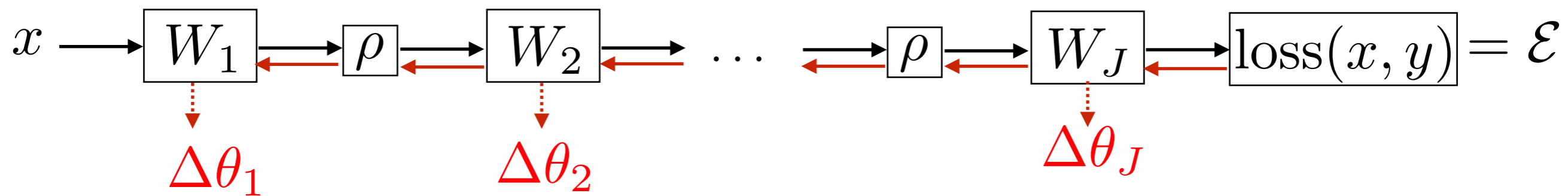


$$\leftarrow \nabla_{x_j}(\mathcal{E}) = \frac{\partial(\rho W_j)^T}{\partial x_j} \nabla_{x_{j+1}}(\mathcal{E}) \quad \downarrow \quad \nabla_{\theta_j}(\mathcal{E}) = \frac{\partial(\rho W_j)^T}{\partial \theta_j} \nabla_{x_{j+1}}(\mathcal{E})$$



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

- Intermediary representations objectives are not explicitly specified.



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

- Intermediary representations objectives are not explicitly specified.
- Difficult to distribute the model ... but GPUs!

Rem.: Yet, this paradigm has simplified lot of frameworks. ex.: pytorch on GPUs!

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

$$\theta^{t+1} = \theta^t - \alpha_t \sum_{i=1}^{\mathcal{B}} \nabla (\ell \circ \Phi)(X_i^t; \theta^t)$$

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

$$\theta^{t+1} = \theta^t - \alpha_t \sum_{i=1}^{\mathcal{B}} \nabla(\ell \circ \Phi)(X_i^t; \theta^t)$$

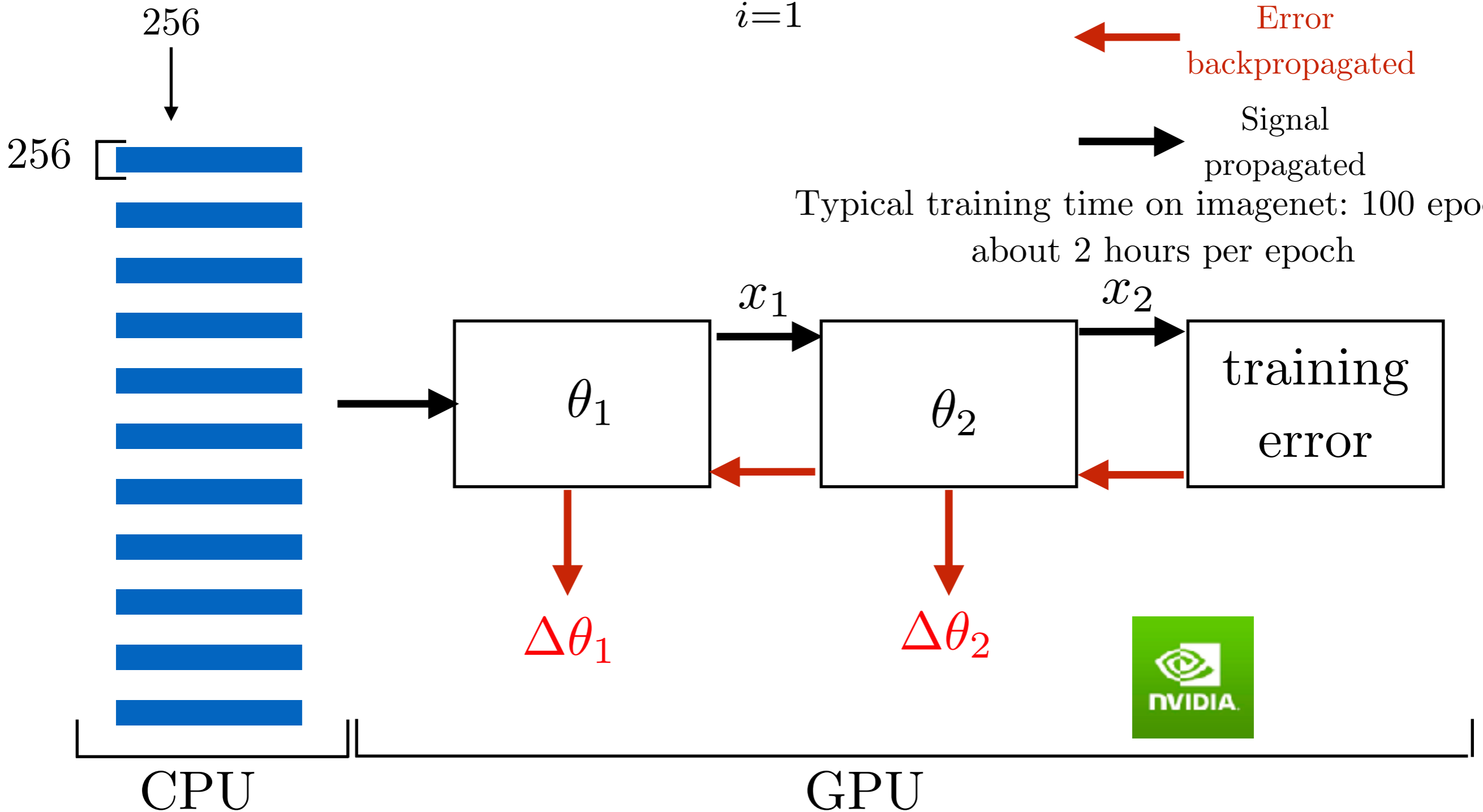
256

256

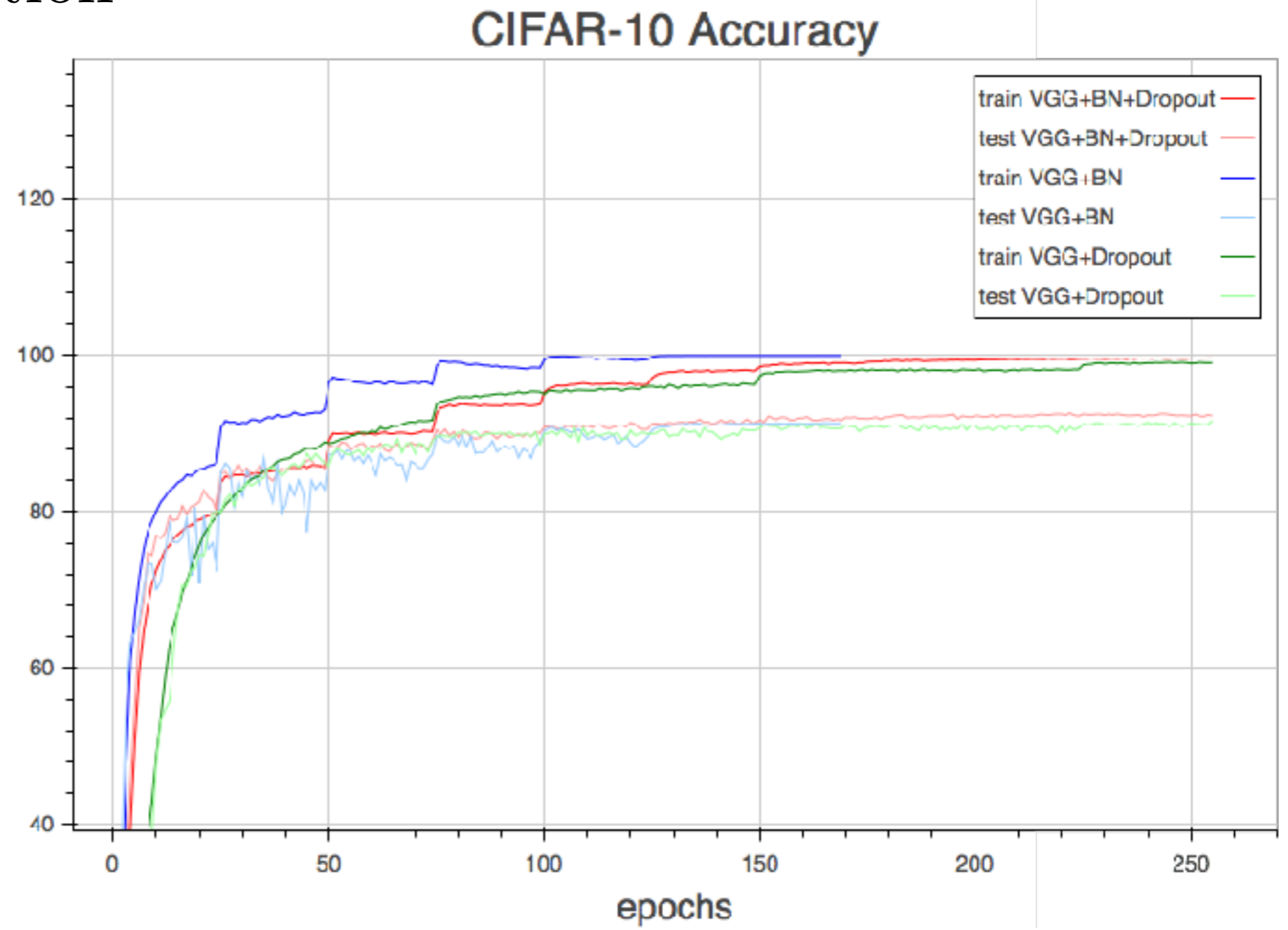
Error backpropagated

Signal propagated

Typical training time on imagenet: 100 epoch about 2 hours per epoch



- Batch-normalization
- Data augmentation
- Dropout
- Learning rate



A strength of DL

A strength of DL

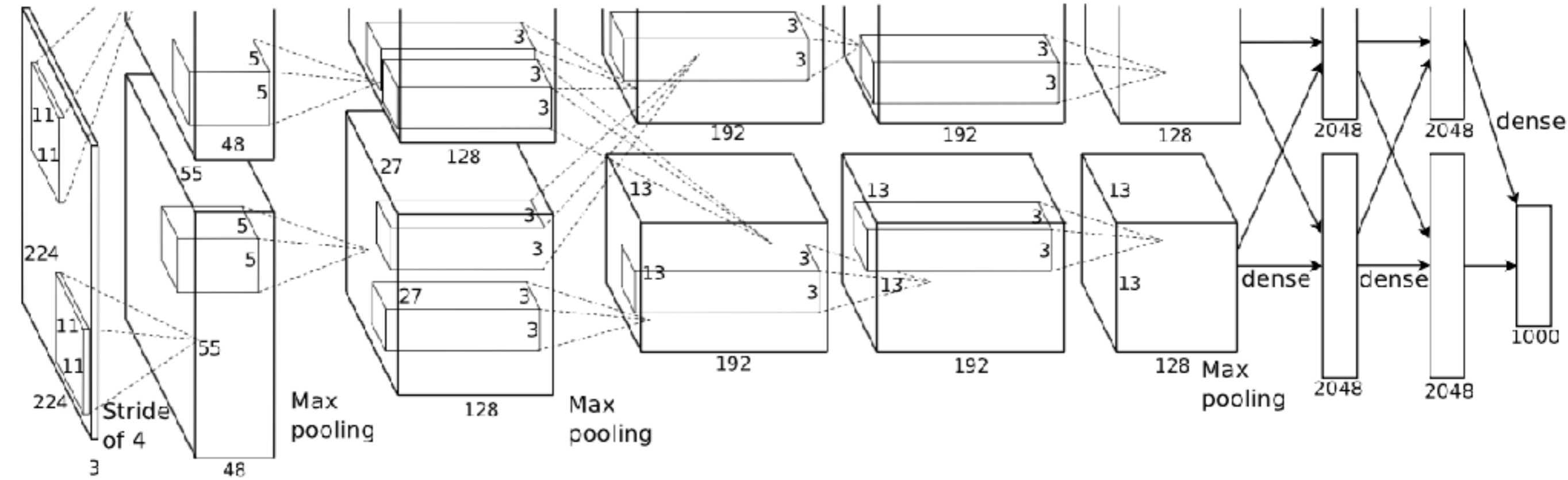
- Data? Computer power? Not only:

A strength of DL

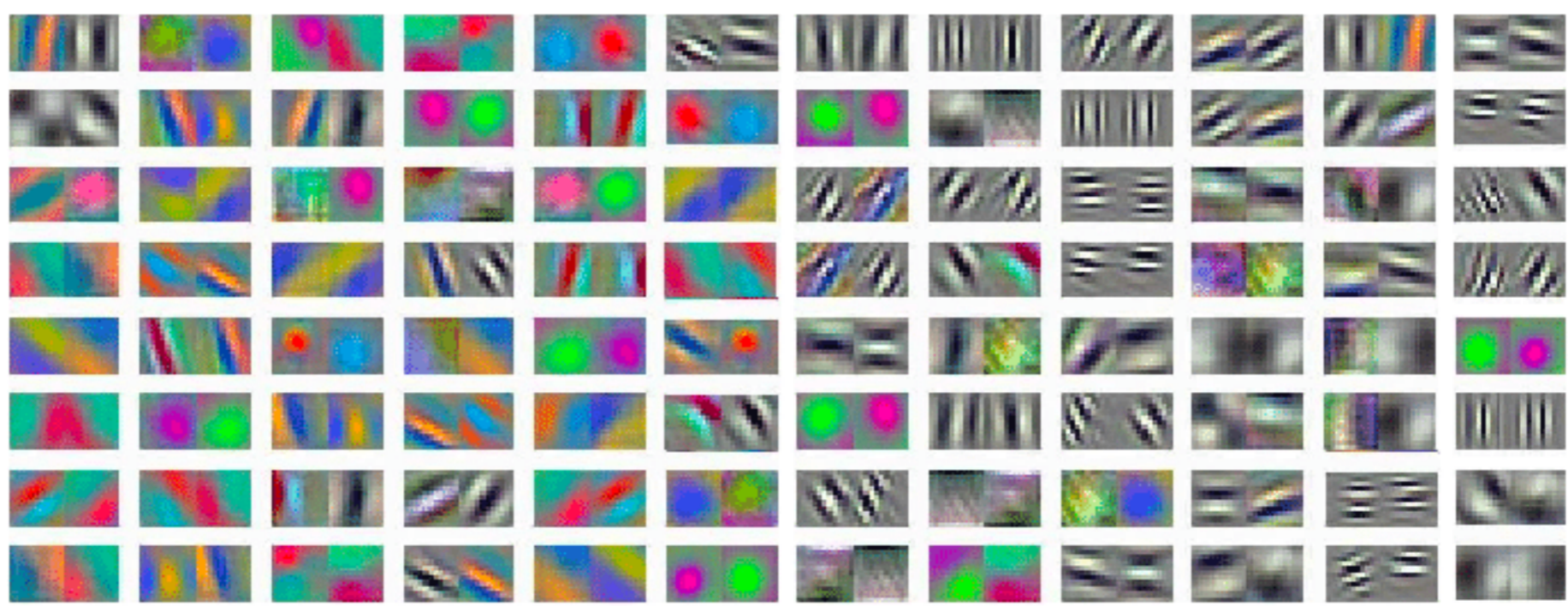
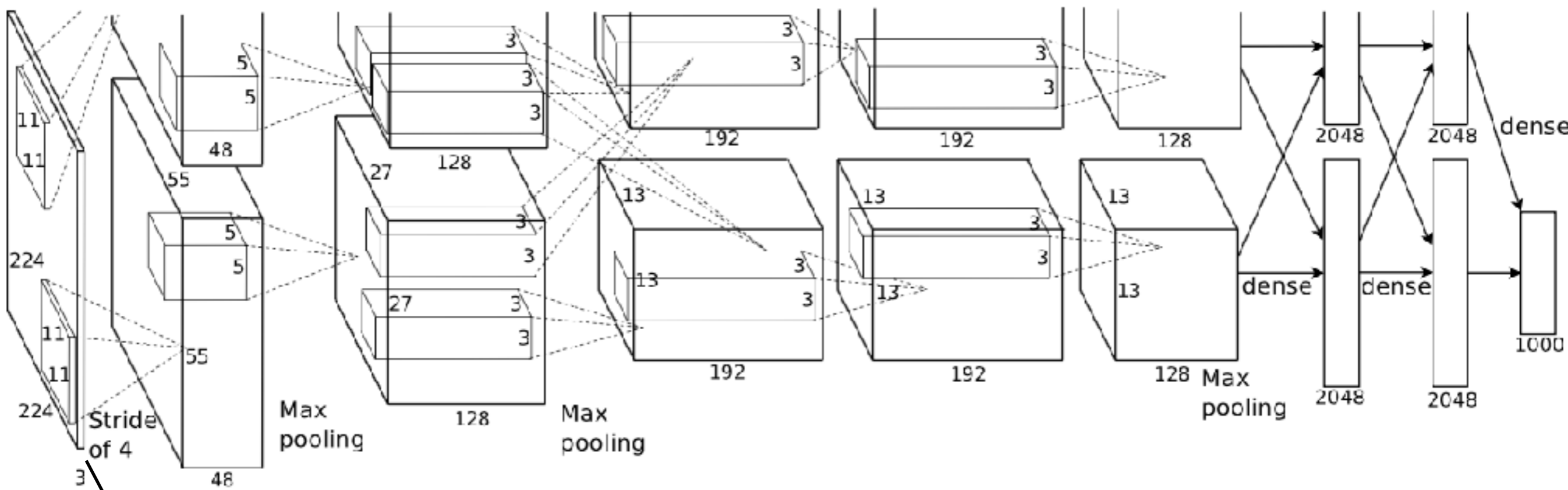
- Data? Computer power? Not only:
- **Flexibility&modularity**: quickly benchmarking non-linearity, layer dimension, losses, batch size, learning rate schedule...

A strength of DL

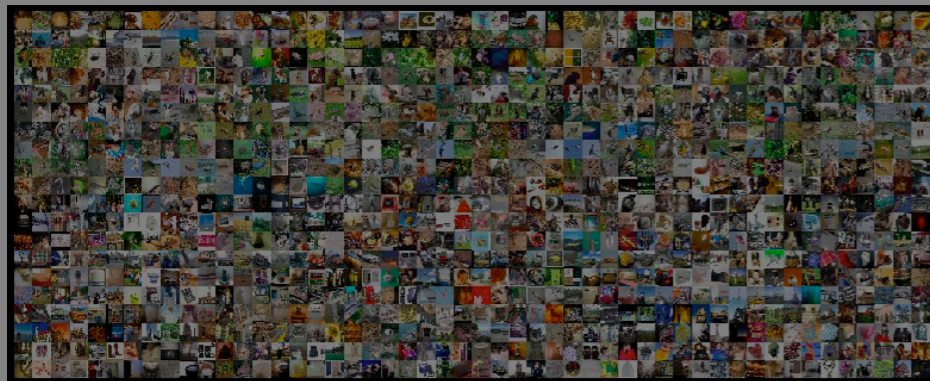
- Data? Computer power? Not only:
- **Flexibility&modularity:** quickly benchmarking non-linearity, layer dimension, losses, batch size, learning rate schedule...
- Is it overfitting? Clearly, yet the representations learned are empirically useful.



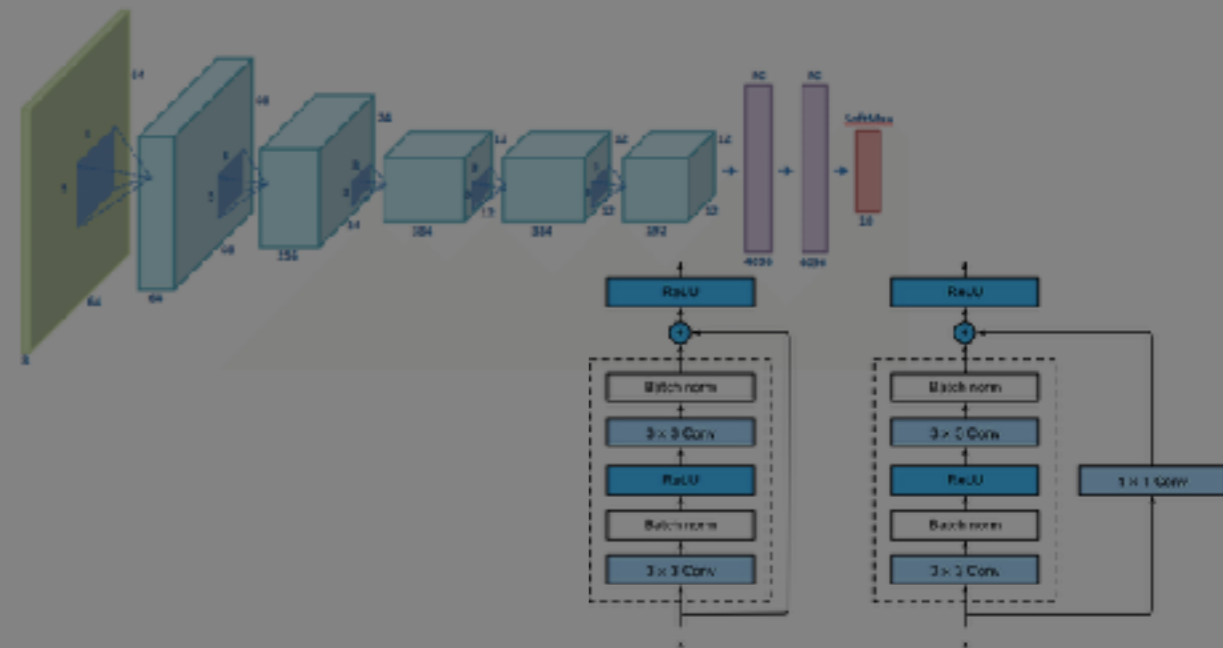
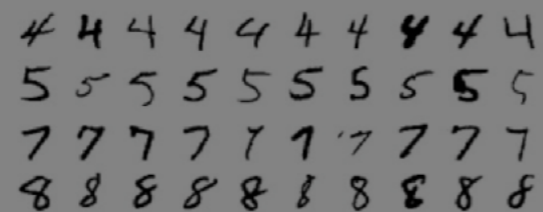
One detailed example: the AlexNet



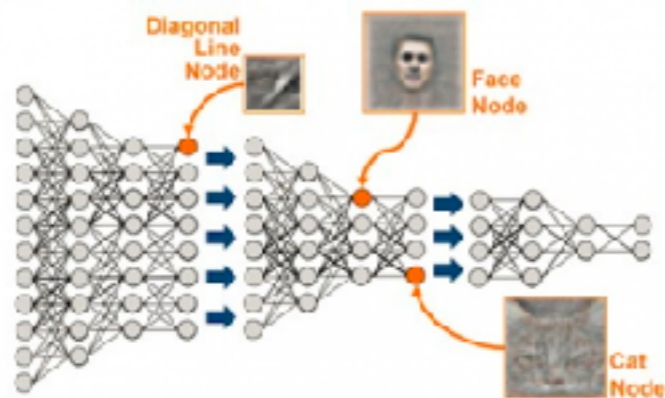
One detailed example: the AlexNet



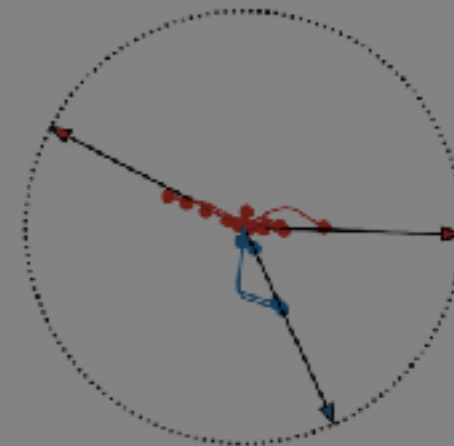
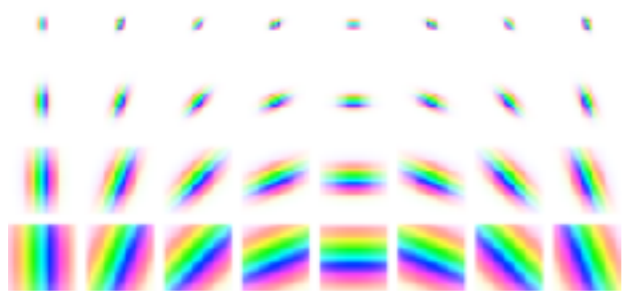
Introduction to image classification



Fighting the curse of dimensionality with Deep Neural Networks



Interpretability in Deep Learning

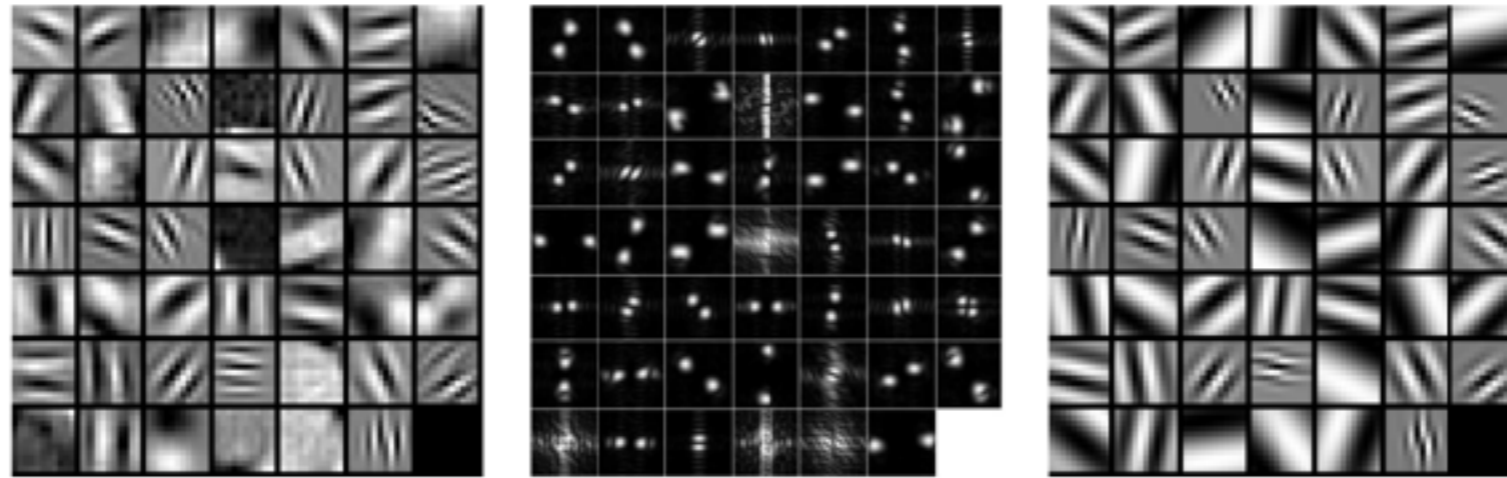


Statistical learning results

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

Under the hood of Deep Neural Networks

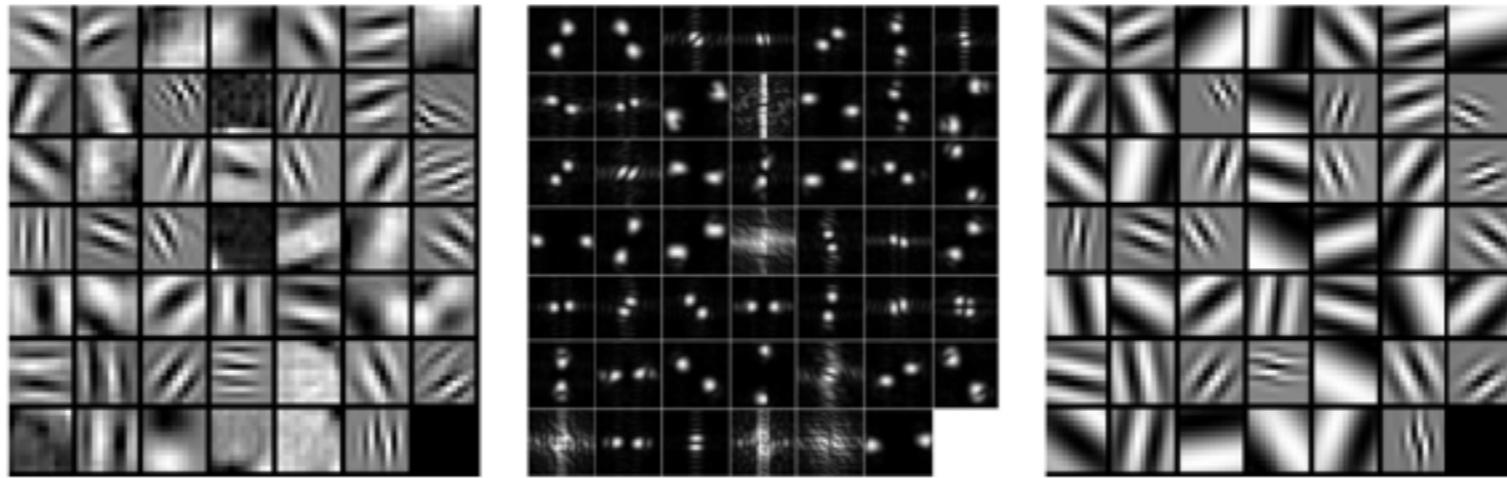
$$\psi_{C,D,\xi}(u) = C e^{-u^T D u} e^{i u^T \xi}$$



Ref.: I Waldspurger's phd

- Consider Gabor filters and fit the model.

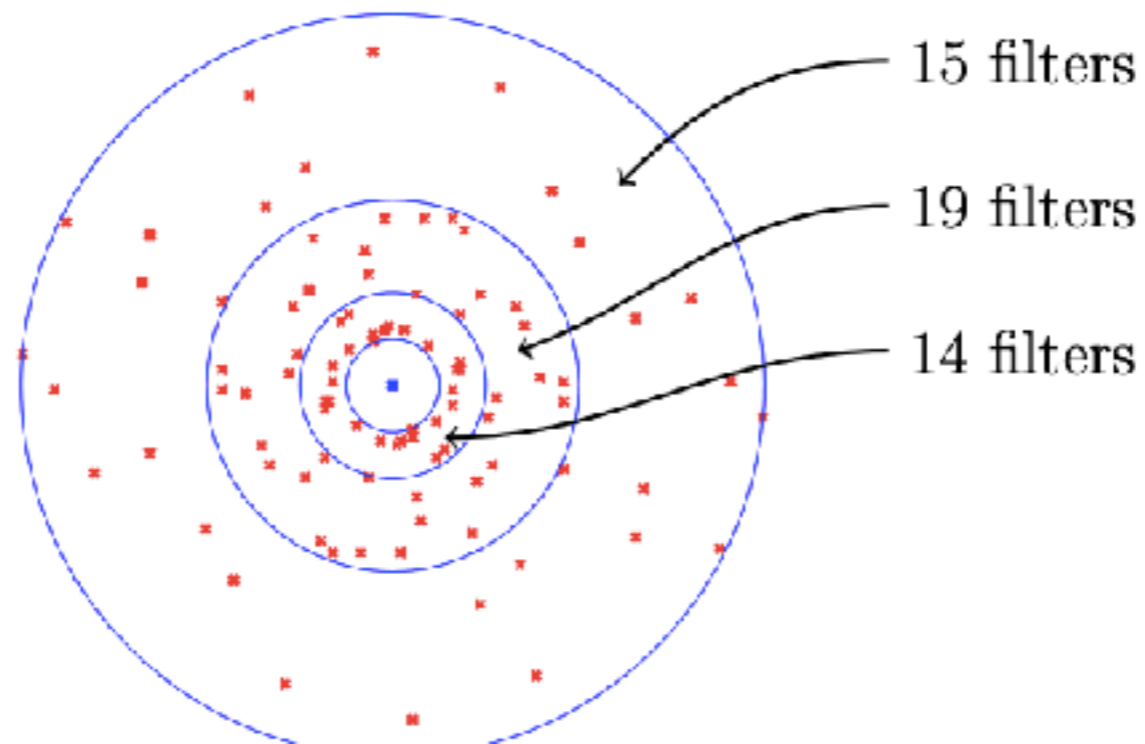
$$\psi_{C,D,\xi}(u) = C e^{-u^T D u} e^{i u^T \xi}$$



Ref.: I Waldspurger's phd

- Consider Gabor filters and fit the model.

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



Ref.: I Waldspurger's phd

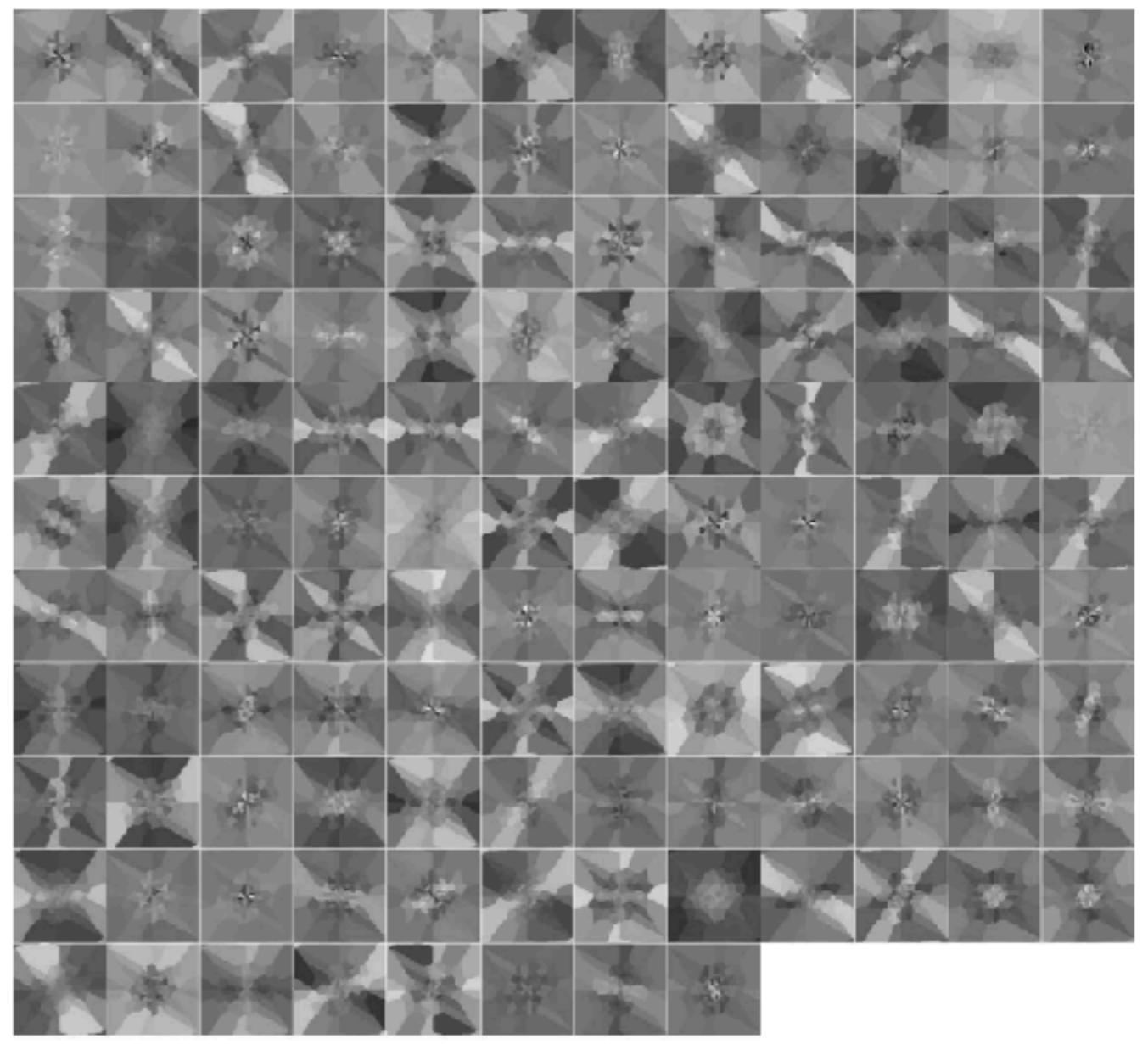
First layer:

$$\psi_\lambda(u)$$

Second layer:

$$\psi(u, \lambda) \approx \phi^1(u) \times \phi^2(\lambda)$$

Recombines along λ



Visualisation of ϕ^2 in the frequency plane

Ref.: I Waldspurger's phd

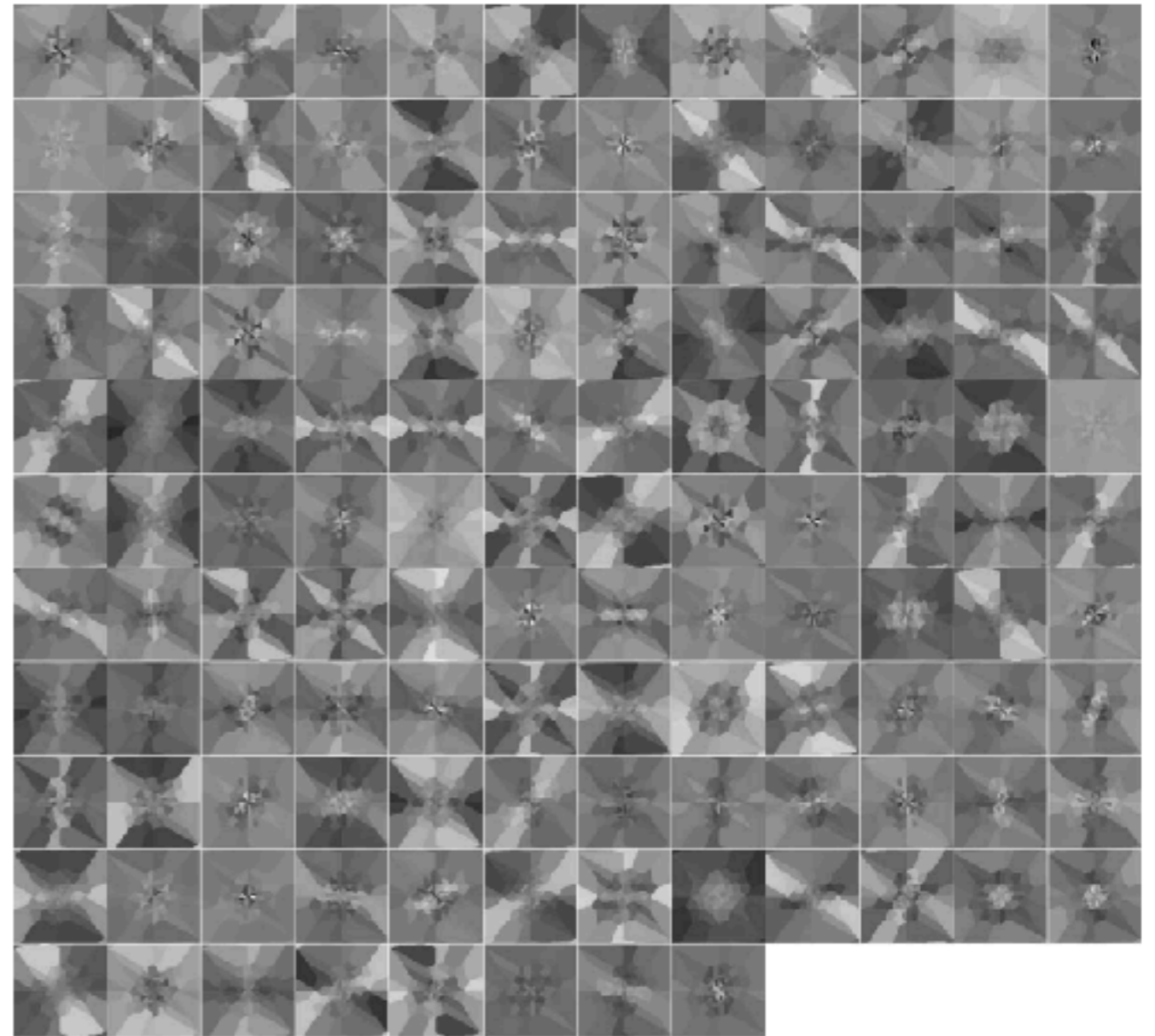
First layer:

$$\psi_\lambda(u)$$

Second layer:

$$\psi(u, \lambda) \approx \phi^1(u) \times \phi^2(\lambda)$$

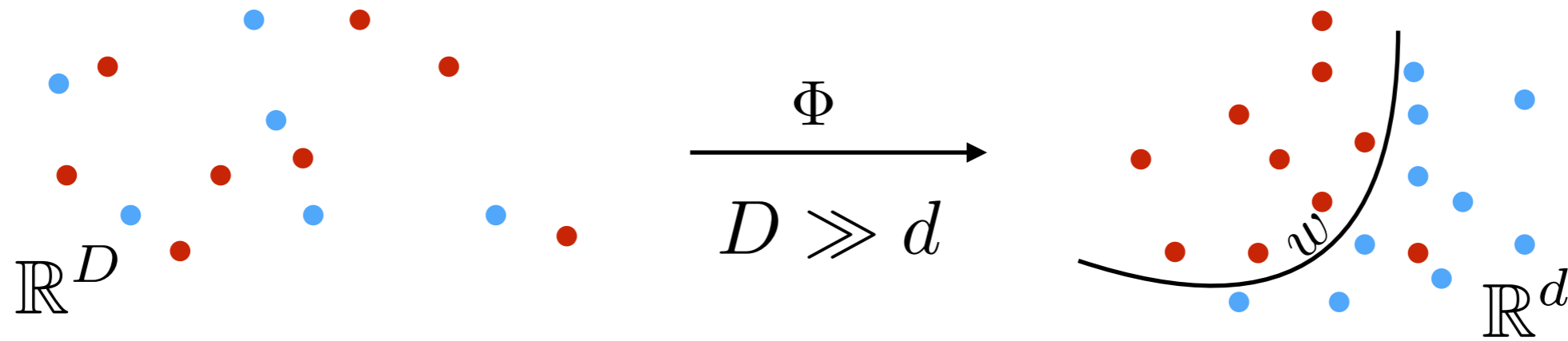
Recombines along λ



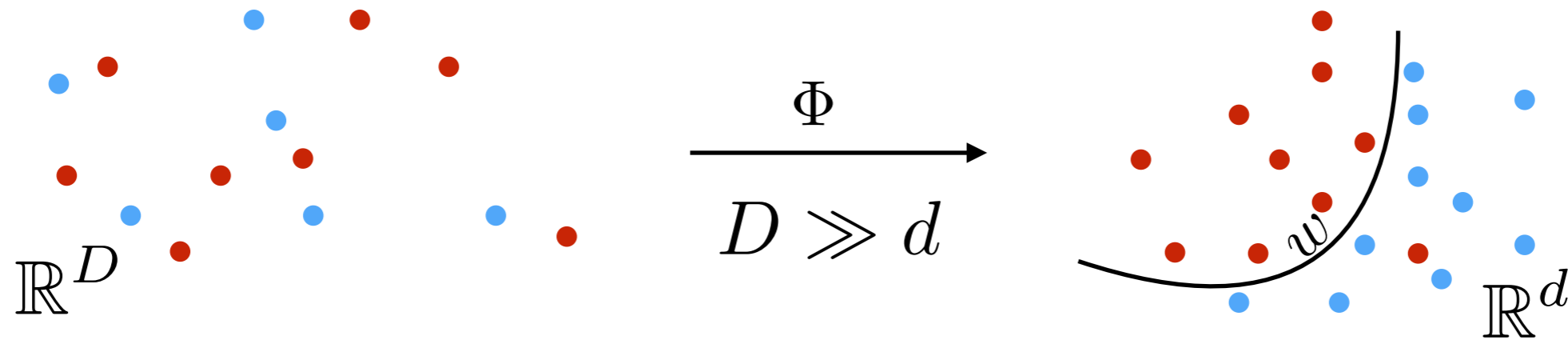
Visualisation of ϕ^2 in the frequency plane

Why was this possible?
 We were aware of the topology
 of the previous layer!

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



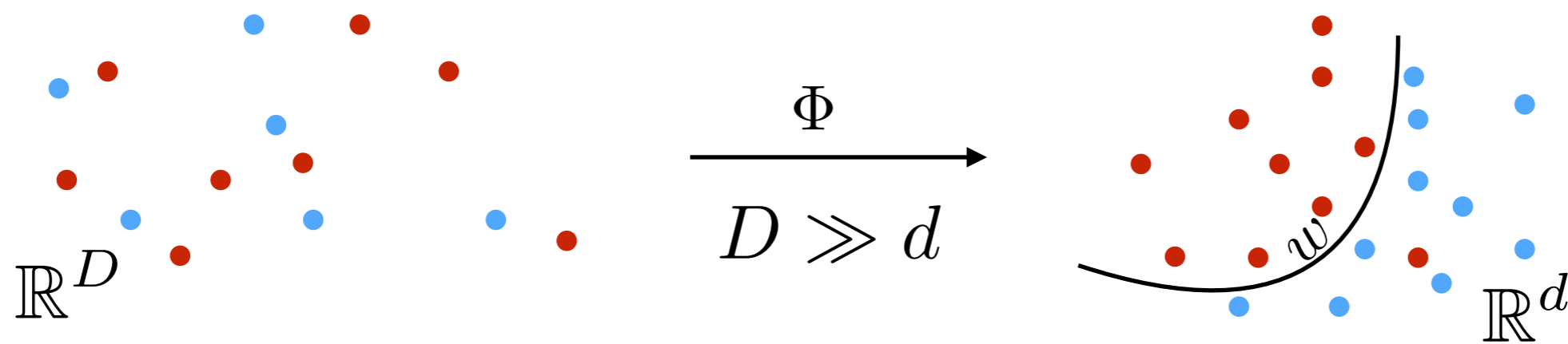
- Objective:** building a representation Φx of x such that a simple (say euclidean) classifier \hat{y} can estimate the label y :



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

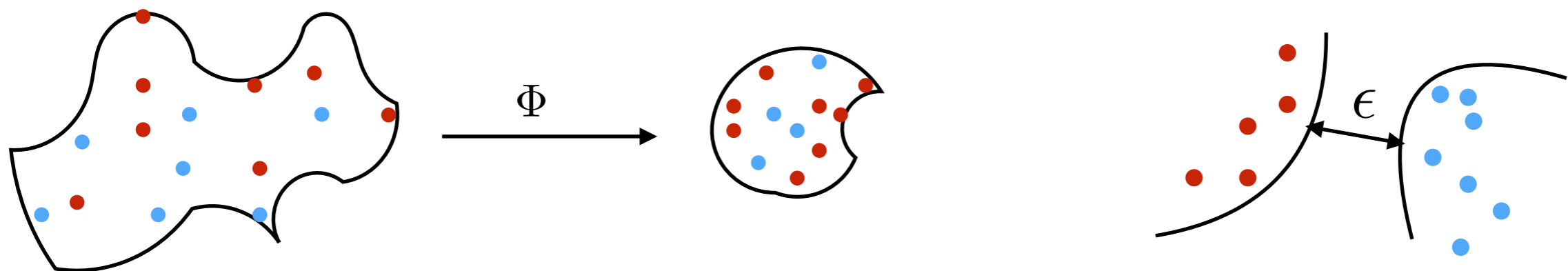
$$\|\Phi x - \Phi x'\| \lll 1 \Rightarrow \hat{y}(x) = \hat{y}(x')$$

- **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:



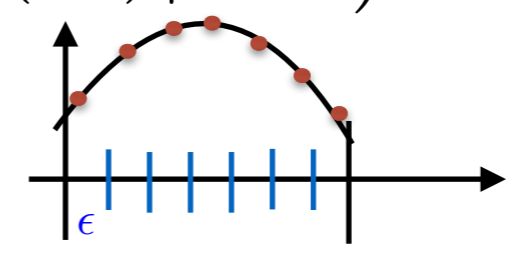
dimensional manifold hypothesis?

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 $N_\epsilon = \mathcal{O}(\epsilon^{-D})$



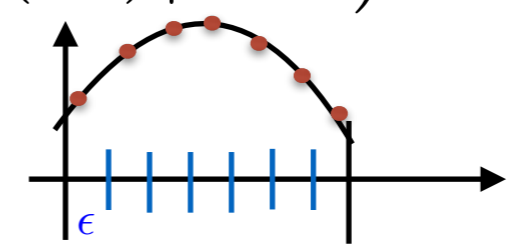
Model on the data: low

dimensional manifold hypothesis?

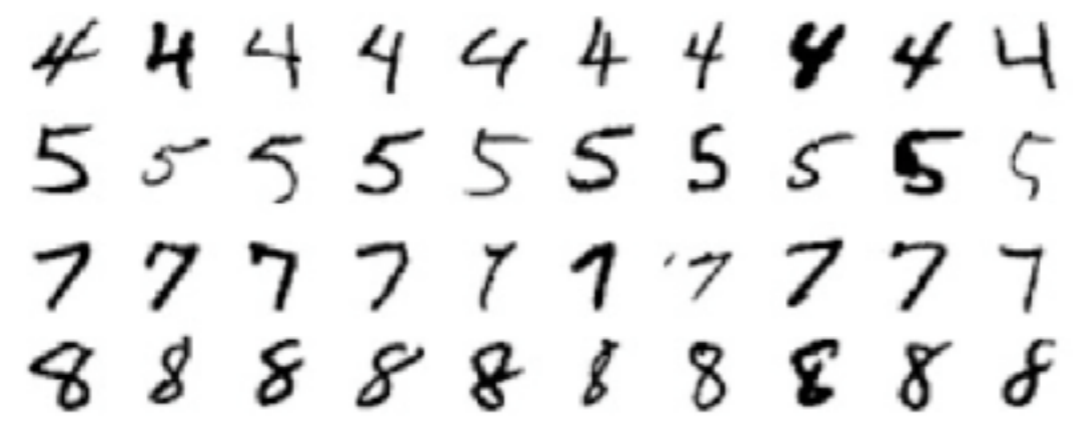
- Low dimensional manifold: dimension up to 6. Not higher:

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 $N_\epsilon = \mathcal{O}(\epsilon^{-D})$



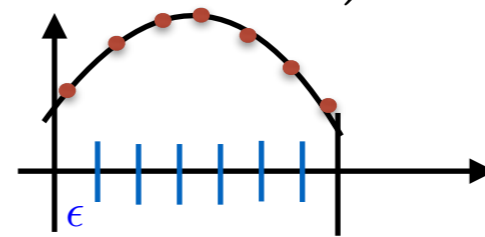
- Can be true for MNIST...



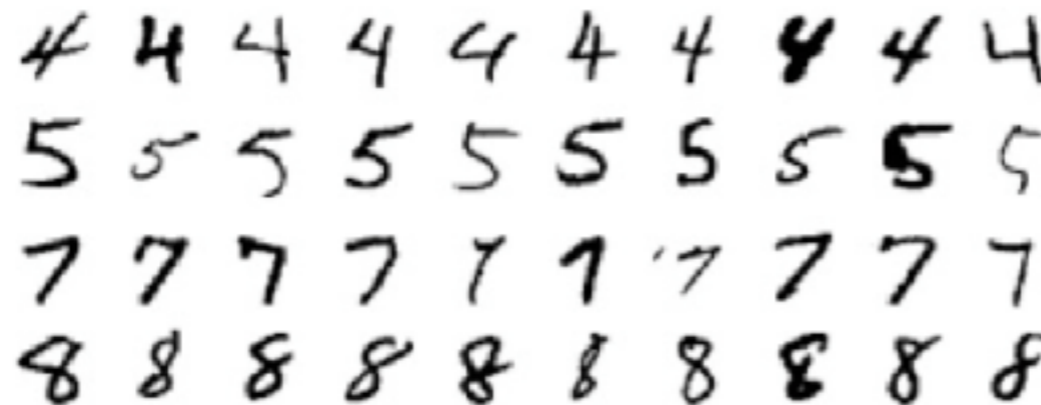
dimensional manifold hypothesis?

- 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...



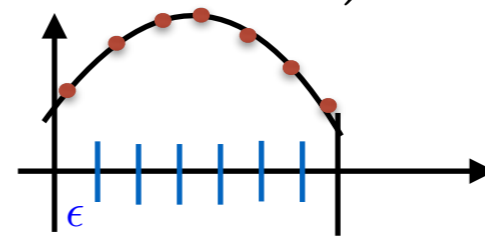
All variabilities are known

Small "limited" deformations + Translation

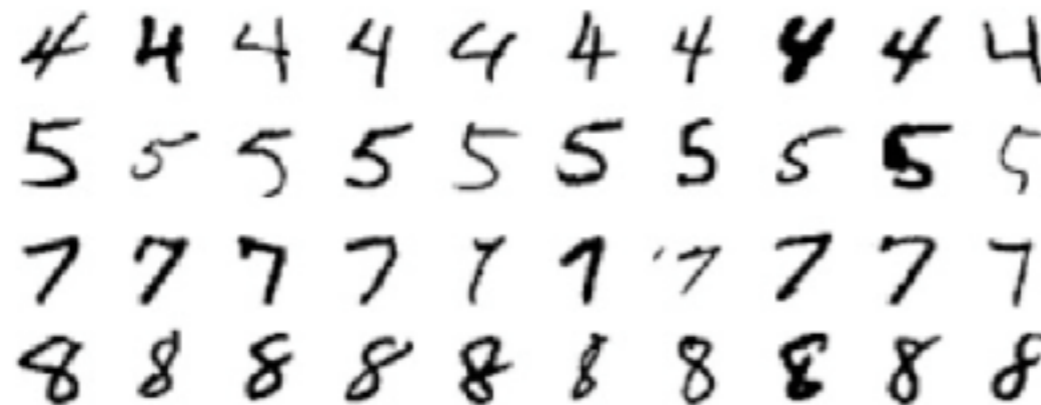
dimensional manifold hypothesis?

- 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?

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



(c) Object color



(d) Background color



(a) Lighting



(b) Scale

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



(c) Object color



(d) Background color



(a) Lighting

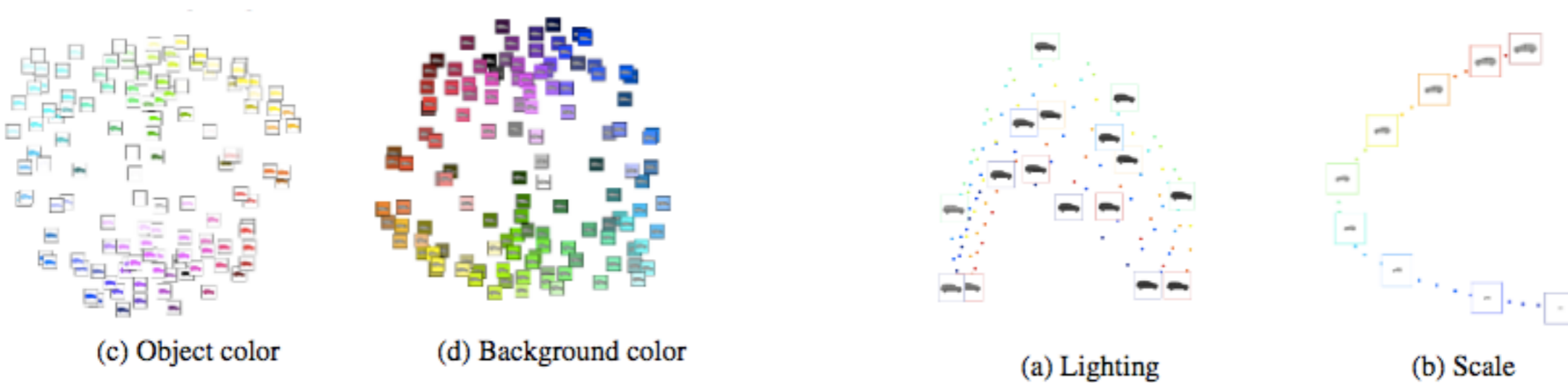


(b) Scale

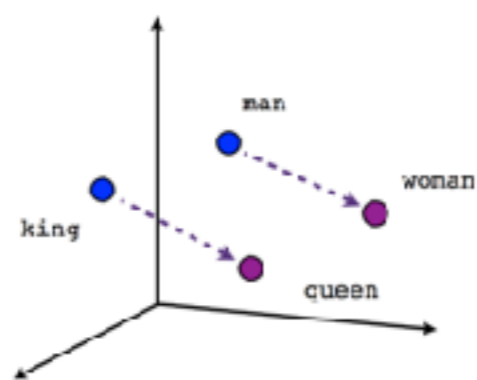
Flattening the space: progressive manifold?

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

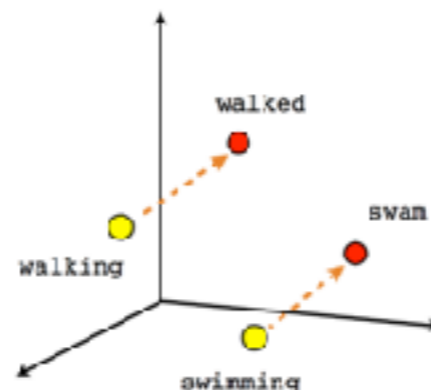
Ref.: Understanding deep features with computer-generated imagery, M Aubry, B Russel



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



Male-Female



Verb tense

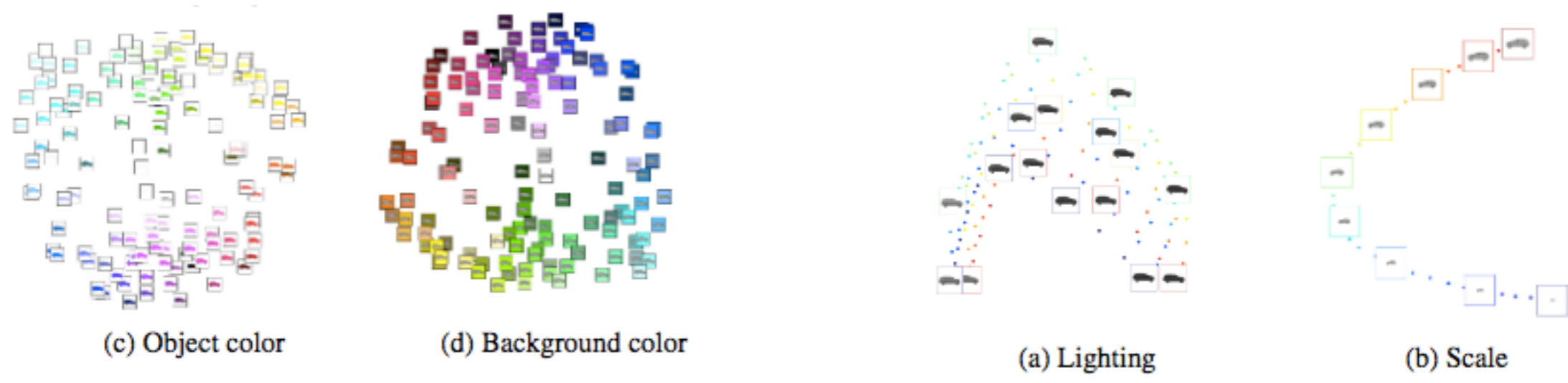


Country-Capital

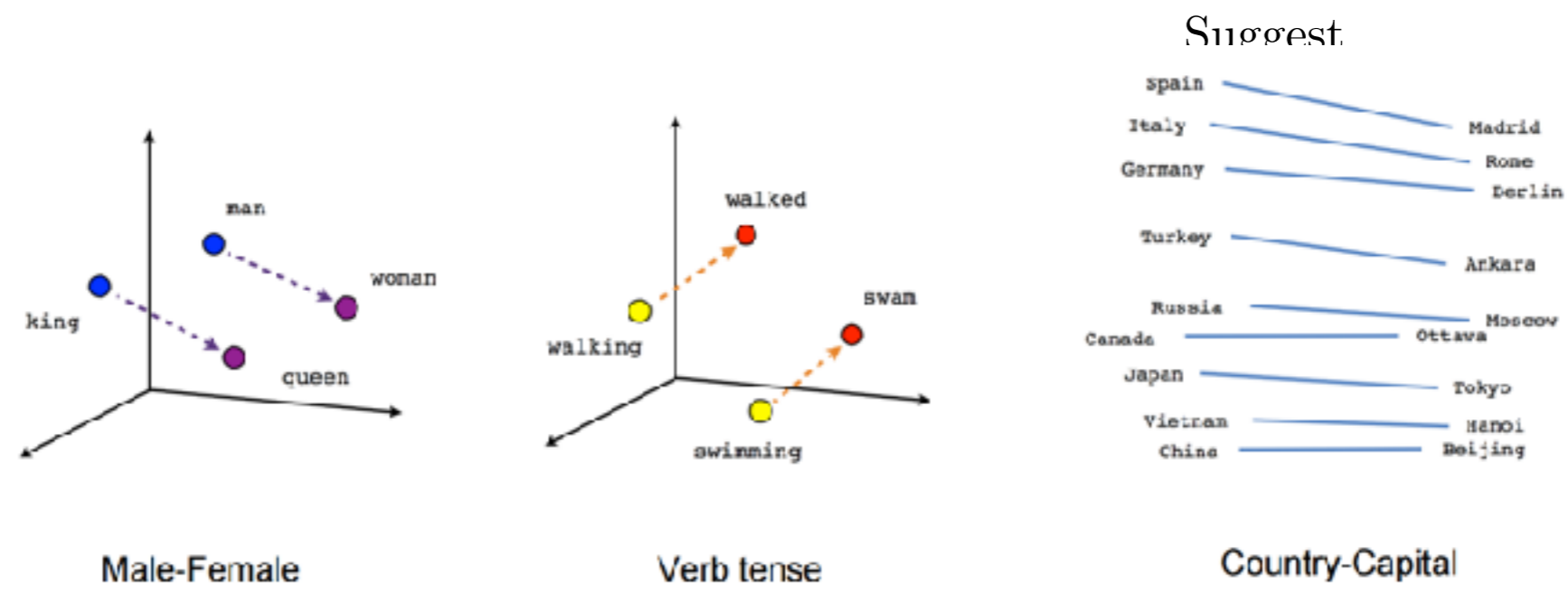
Flattening the space: progressive manifold?

- 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?



Difficult to find evidences of such phenomena

Concept of neuron?

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

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.

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.

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.



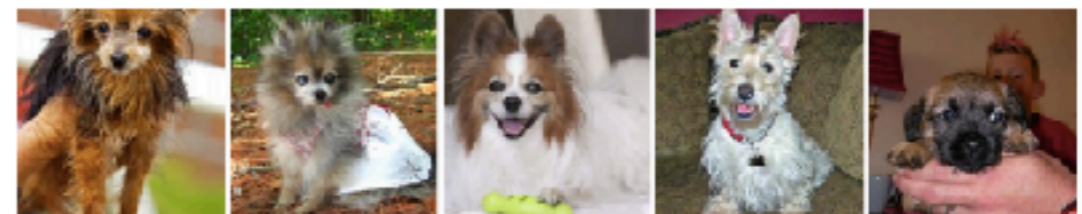
(a) Direction sensitive to white, spread flowers.



(b) Direction sensitive to white dogs.



(c) Direction sensitive to spread shapes.

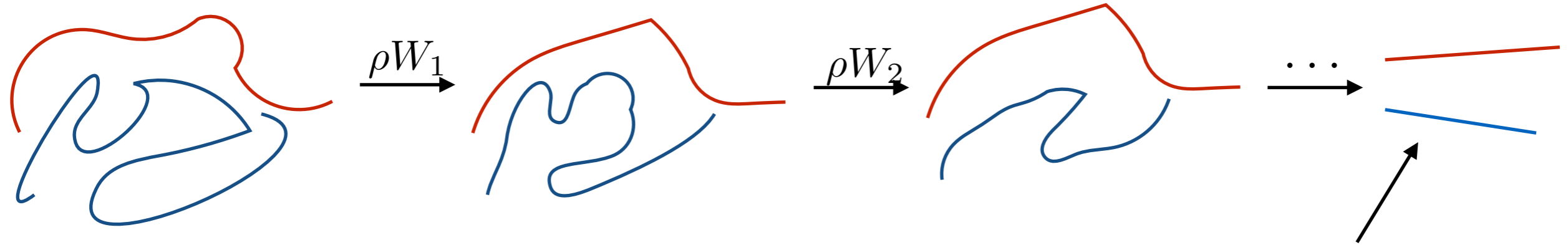


(d) Direction sensitive to dogs with brown heads.

Mechanism proposal: Flattening the level sets

— class 1
— class 2

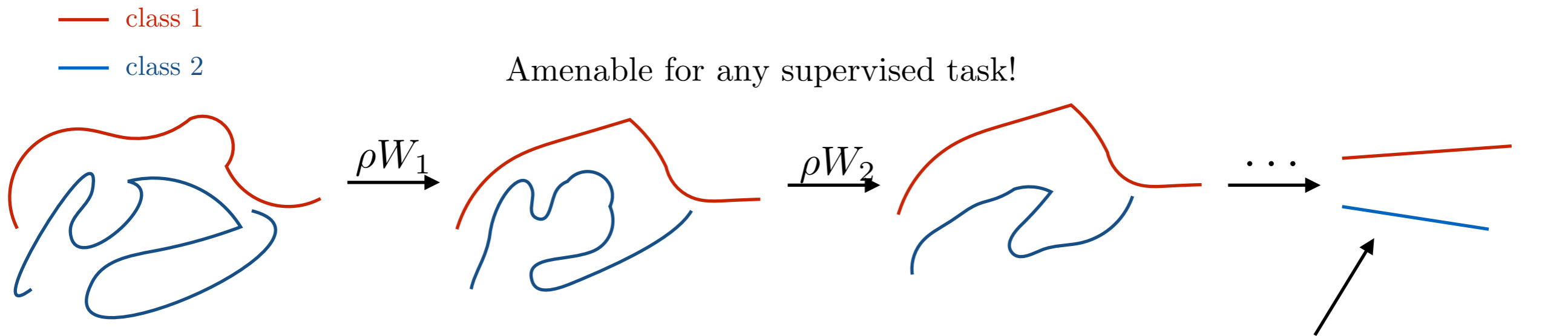
Amenable for any supervised task!



Ref.: Understanding Deep Convolutional Networks, Mallat, 2016

Linear invariant can be computed!

Mechanism proposal: Flattening the level sets



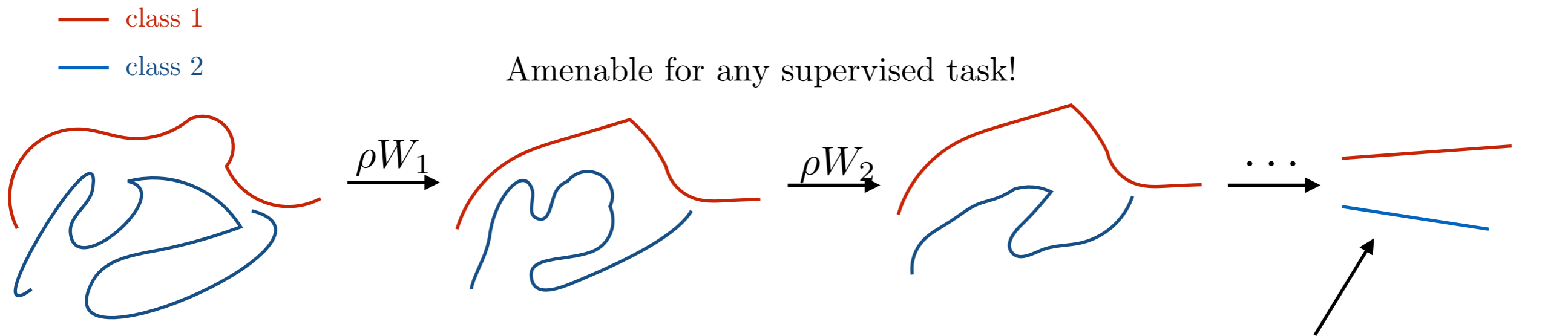
Ref.: Understanding Deep Convolutional Networks, Mallat, 2016

Linear invariant can be computed!

- How to linearize? Ex.: Gâteaux differentiability

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

Mechanism proposal: Flattening the level sets



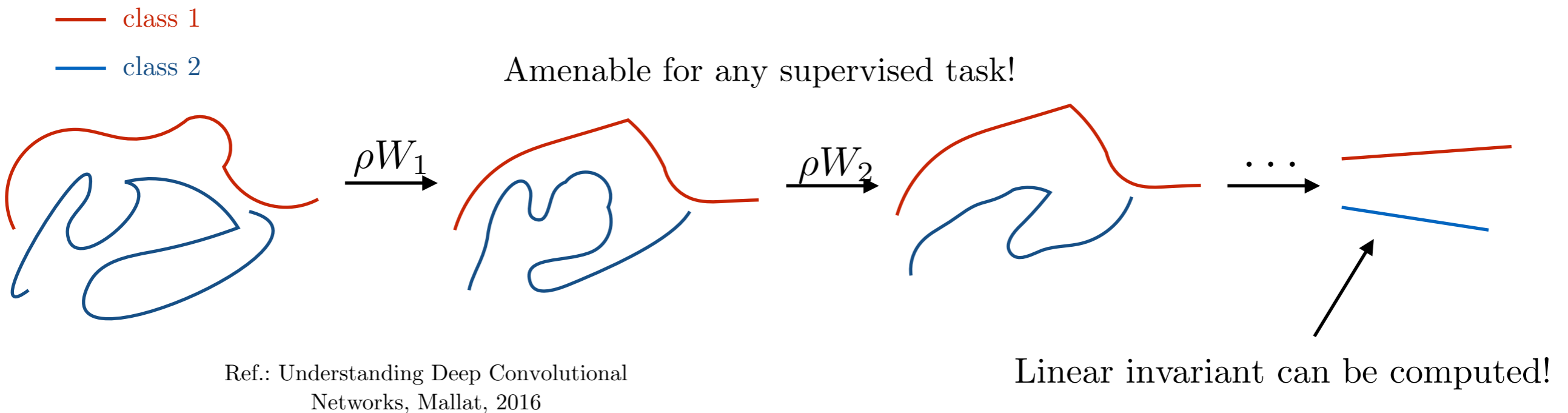
Ref.: Understanding Deep Convolutional Networks, Mallat, 2016

Linear invariant can be computed!

- How to linearize? Ex.: Gâteaux differentiability

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

Mechanism proposal: Flattening the level sets



- How to linearize? Ex.: Gâteaux differentiability

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

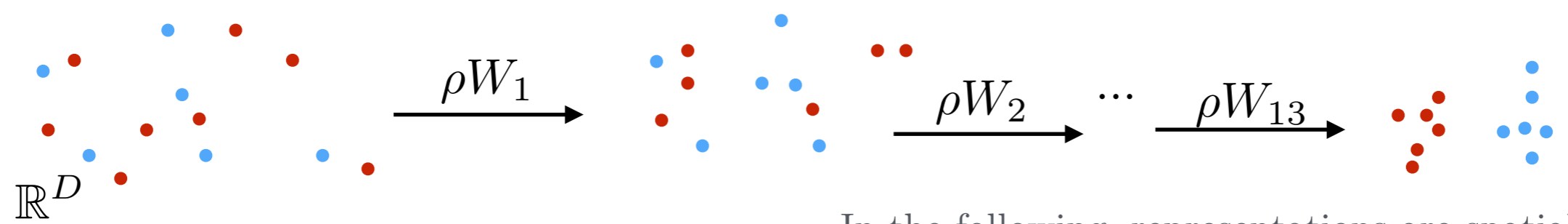
- However, exhibiting \mathcal{T} can be difficult. (*curse of dimensionality*)

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

Empirical observation: Progressive separability

Empirical observation: Progressive separability

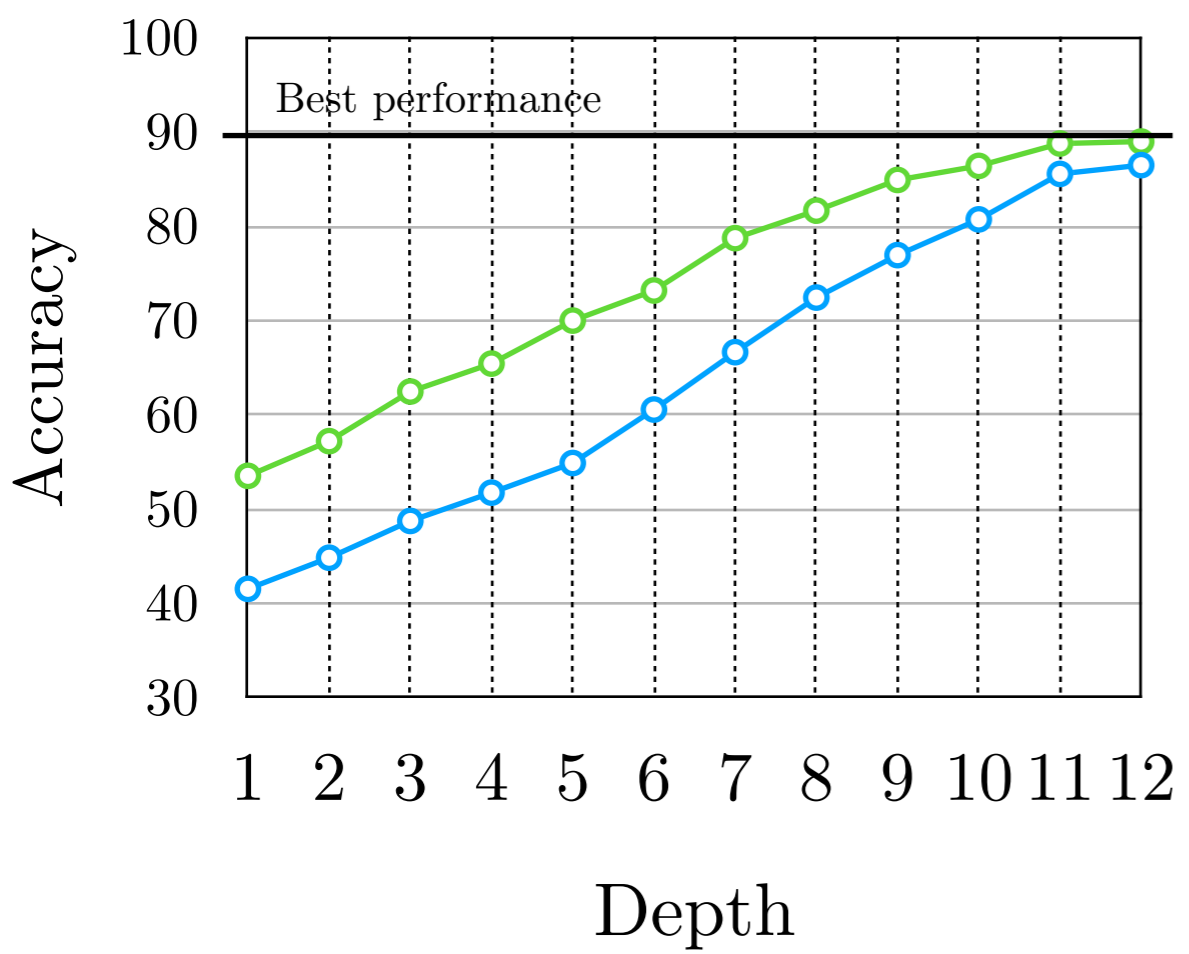
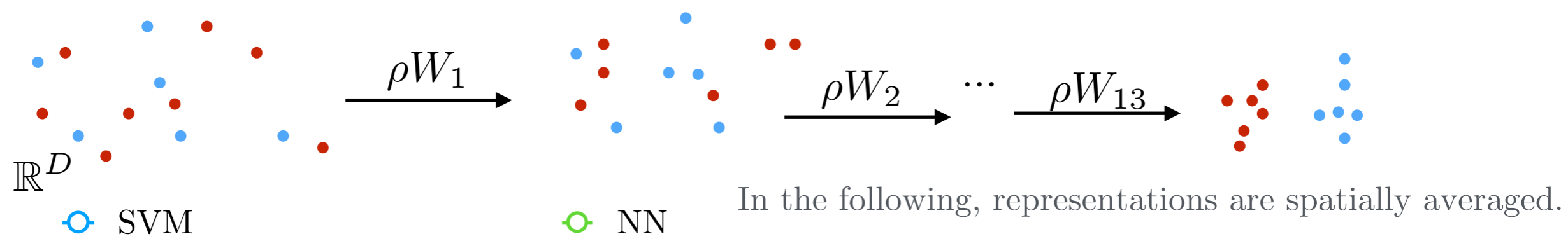
- Typical CNN exhibits a progressive contraction & separation, w.r.t. the depth:



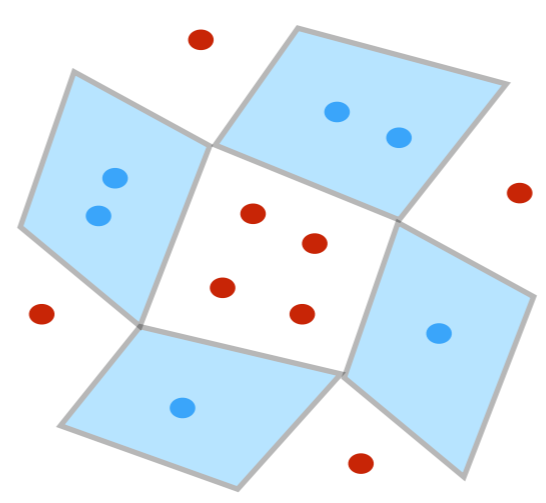
In the following, representations are spatially averaged.

Empirical observation: Progressive separability

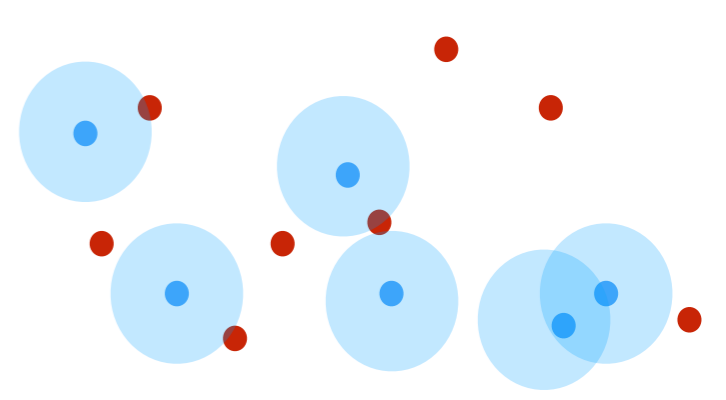
- Typical CNN exhibits a progressive contraction & separation, w.r.t. the depth:



Nearest Neighbor (NN)



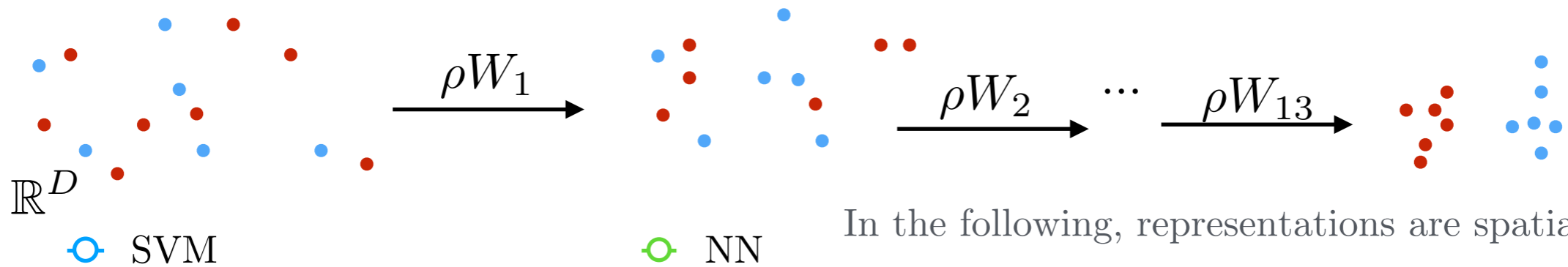
Gaussian SVM



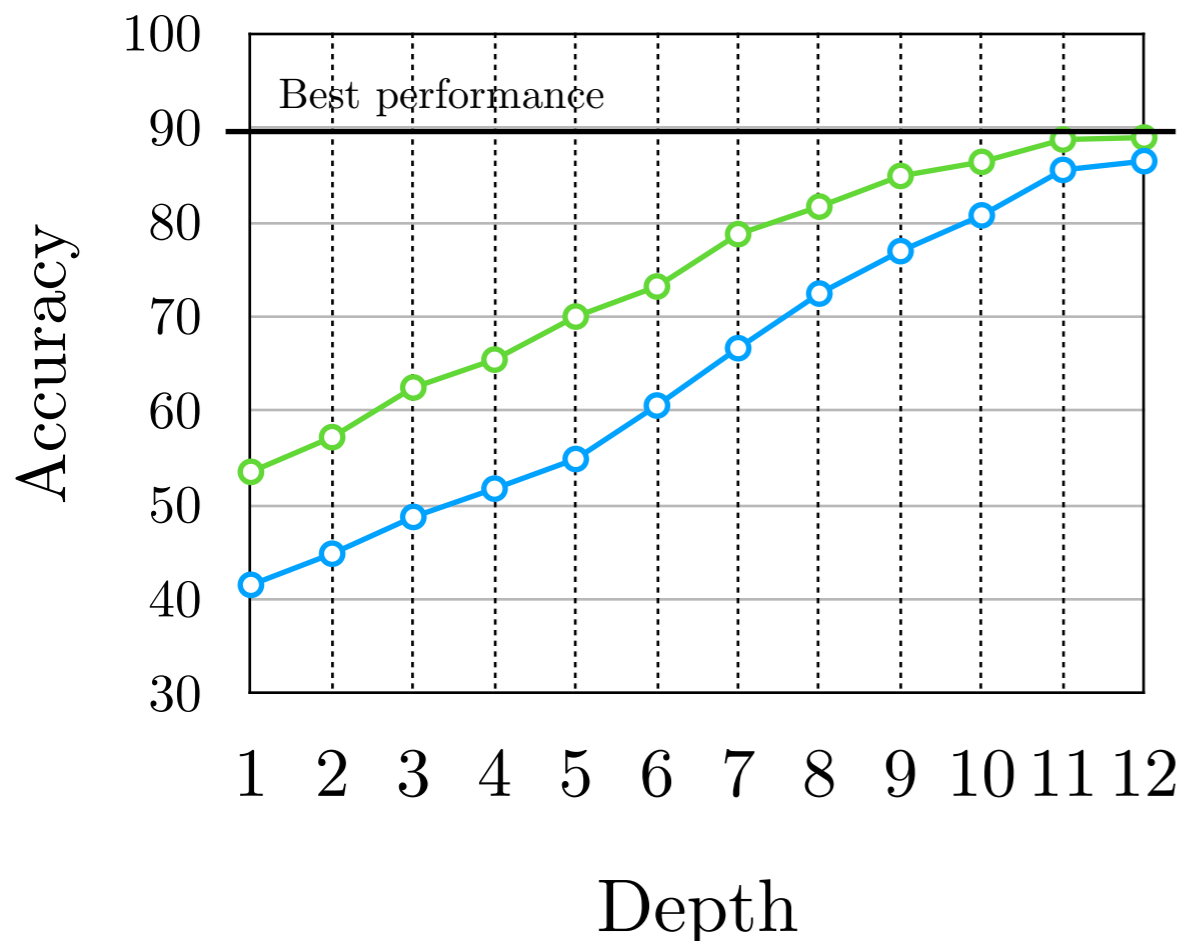
Localised classifiers

Empirical observation: Progressive separability

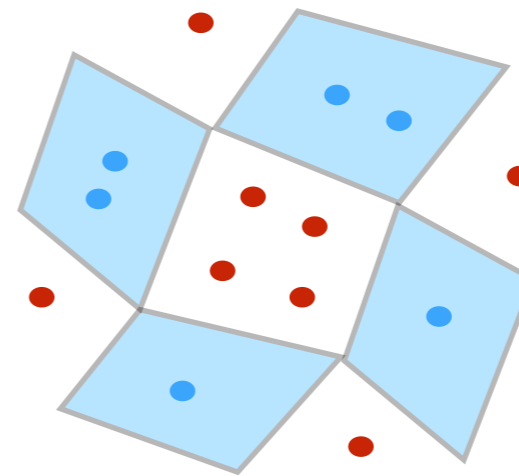
- Typical CNN exhibits a progressive contraction & separation, w.r.t. the depth:



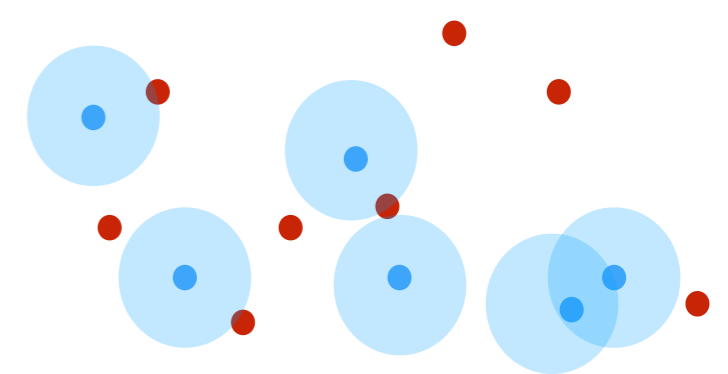
In the following, representations are spatially averaged.



Nearest Neighbor (NN)



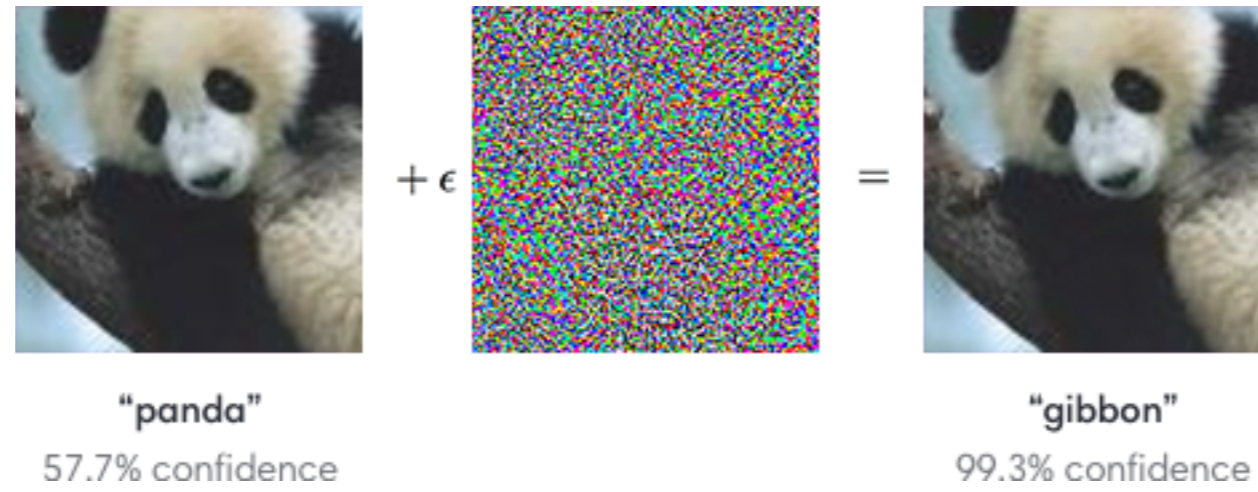
Gaussian SVM



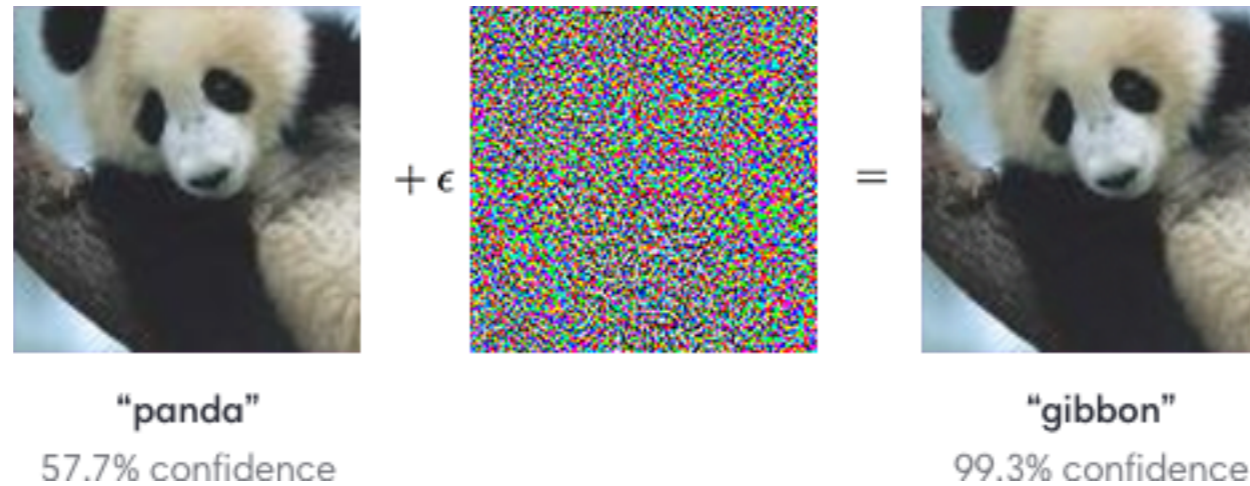
Localised classifiers

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

- **How can we explain it?**



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



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

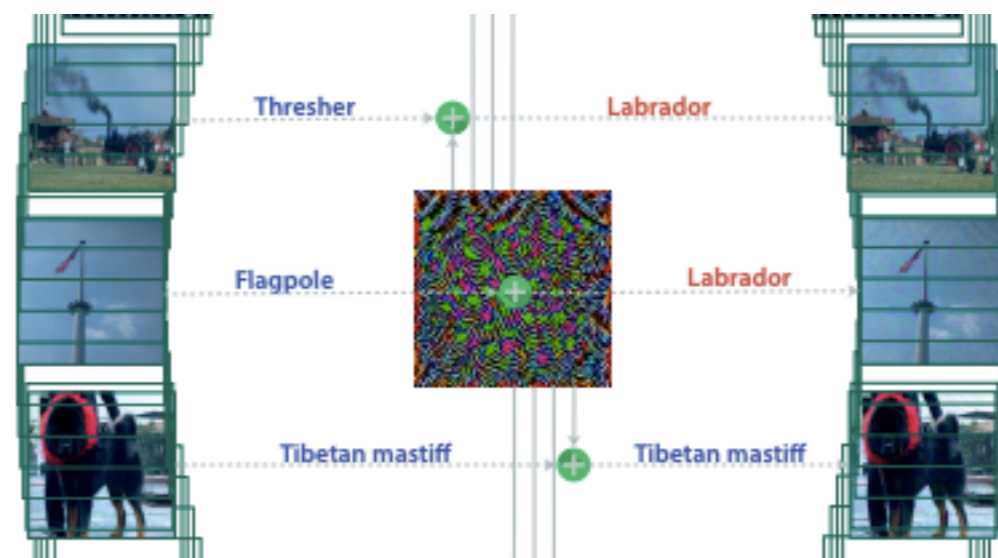
Ref.: Lipschitz Regularity of deep neural networks, Scaman and Virmaux

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

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.

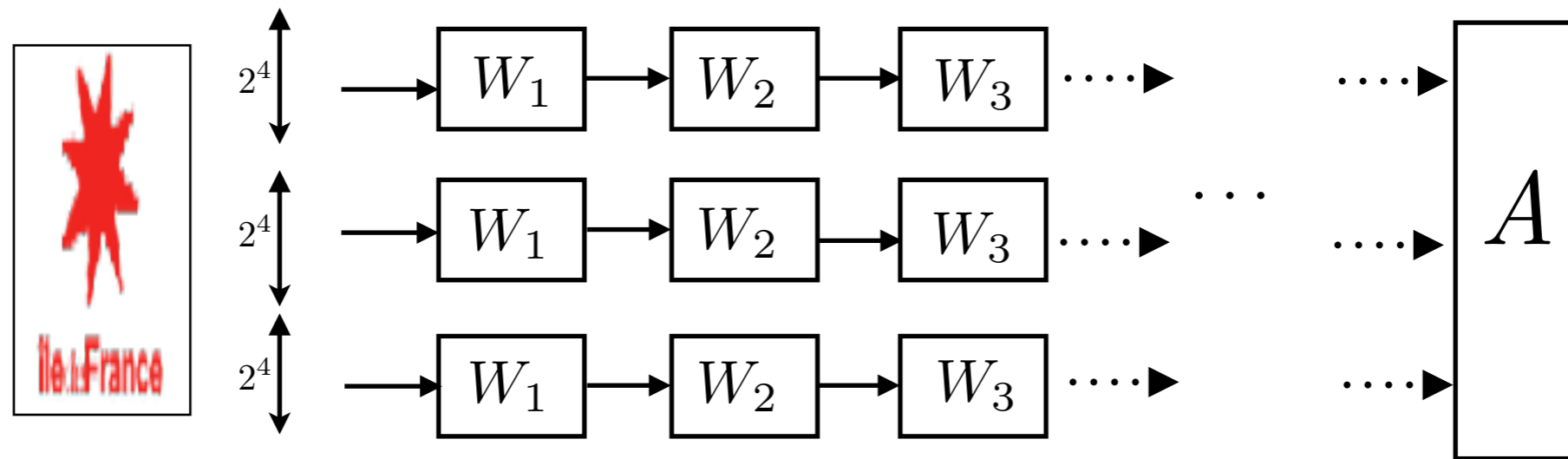


Surprising BagNet

Spatial distribution

"BagNet"

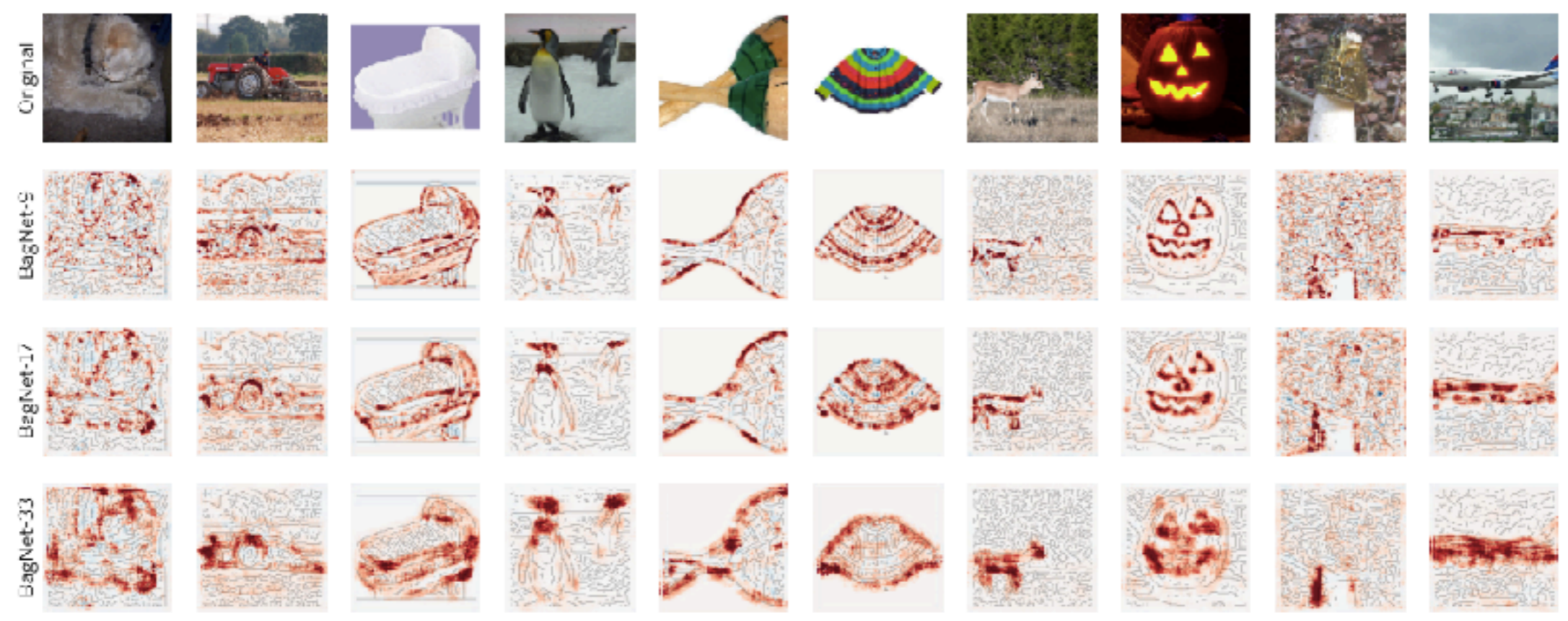
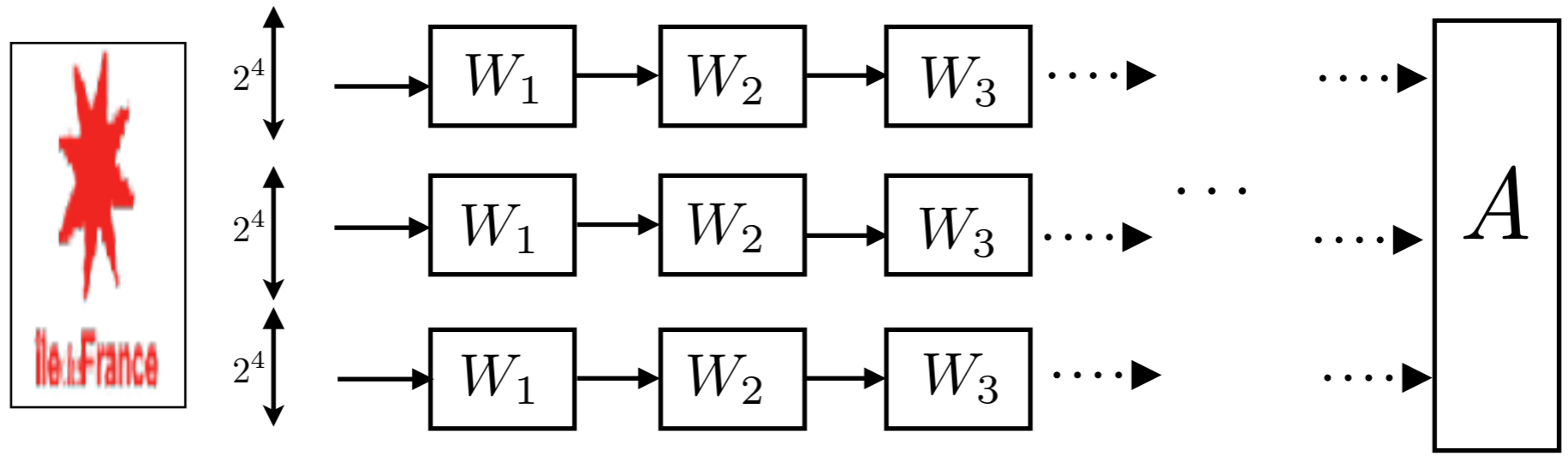
Ref.: APPROXIMATING CNNs WITH BAG-OF-LOCALFEATURES MODELS WORKS SURPRISINGLY WELL ON IMAGENET



Surprising BagNet Spatial distribution

"BagNet"

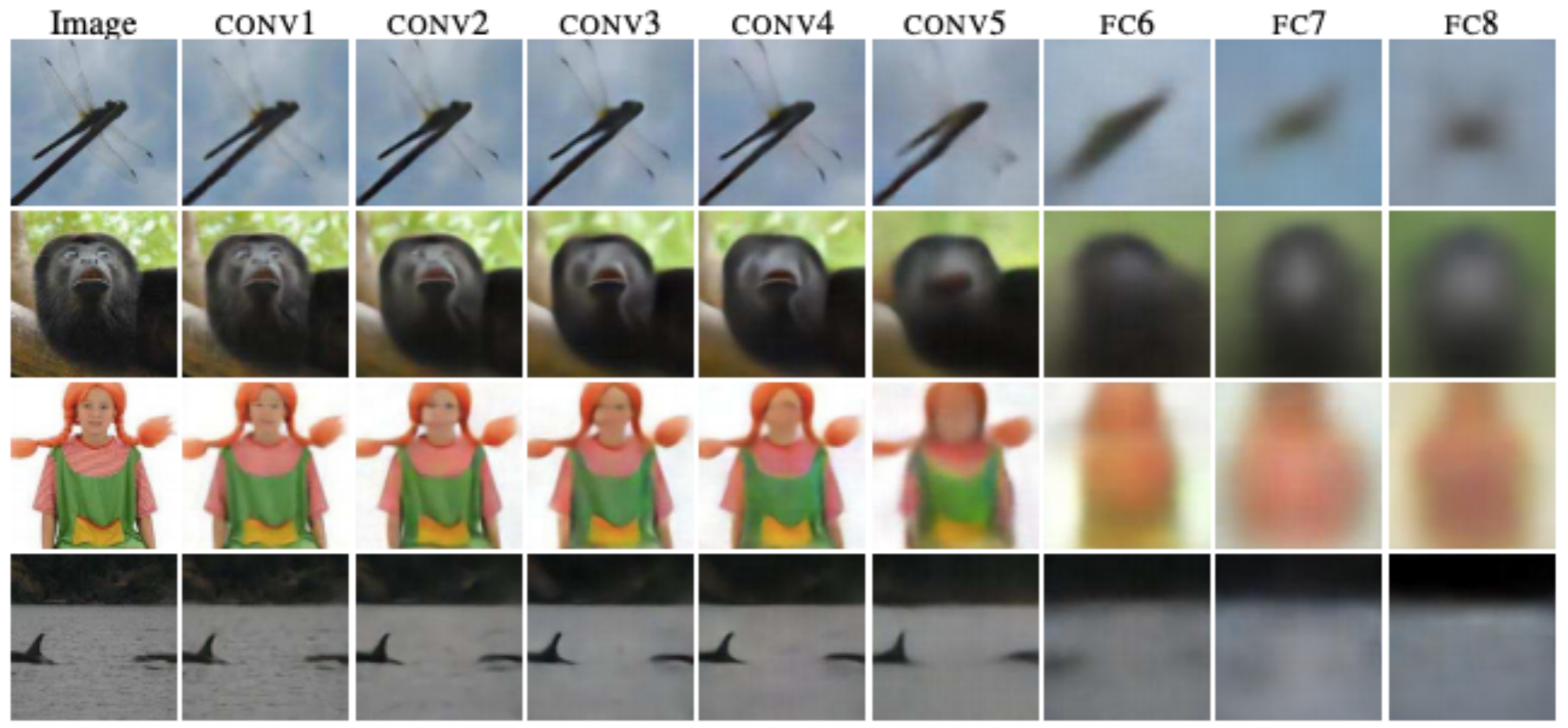
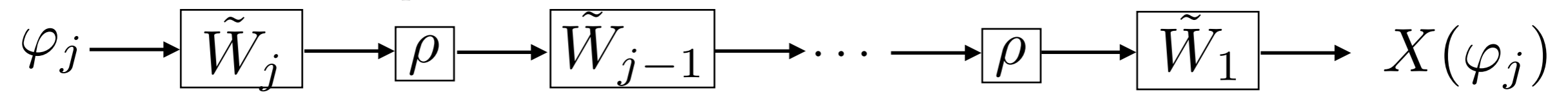
Ref.: APPROXIMATING CNNs WITH BAG-OF-LOCALFEATURES MODELS WORKS SURPRISINGLY WELL ON IMAGENET



Reconstruction from a given layer?



Learn the operators!



(Information bottleneck)

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

Measures the dependancy between variables

(Information bottleneck)

- Reducing the information sounds relevant:

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

Measures the dependancy between variables

(Information bottleneck)

- Reducing the information sounds relevant:

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

Measures the dependancy between variables

(Information bottleneck)

- Reducing the information sounds relevant:

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

Measures the dependancy between variables

(Information bottleneck)

- Reducing the information sounds relevant:

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

Measures the dependancy between variables

(Information bottleneck)

- Reducing the information sounds relevant:

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

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

(Information bottleneck)

- Reducing the information sounds relevant:

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

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

(Information bottleneck)

- Reducing the information sounds relevant:

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

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:

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

(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

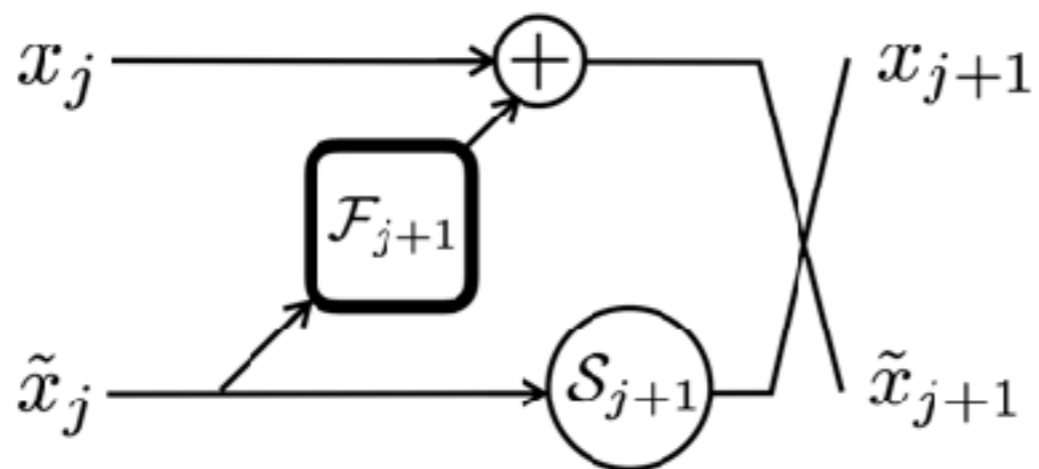
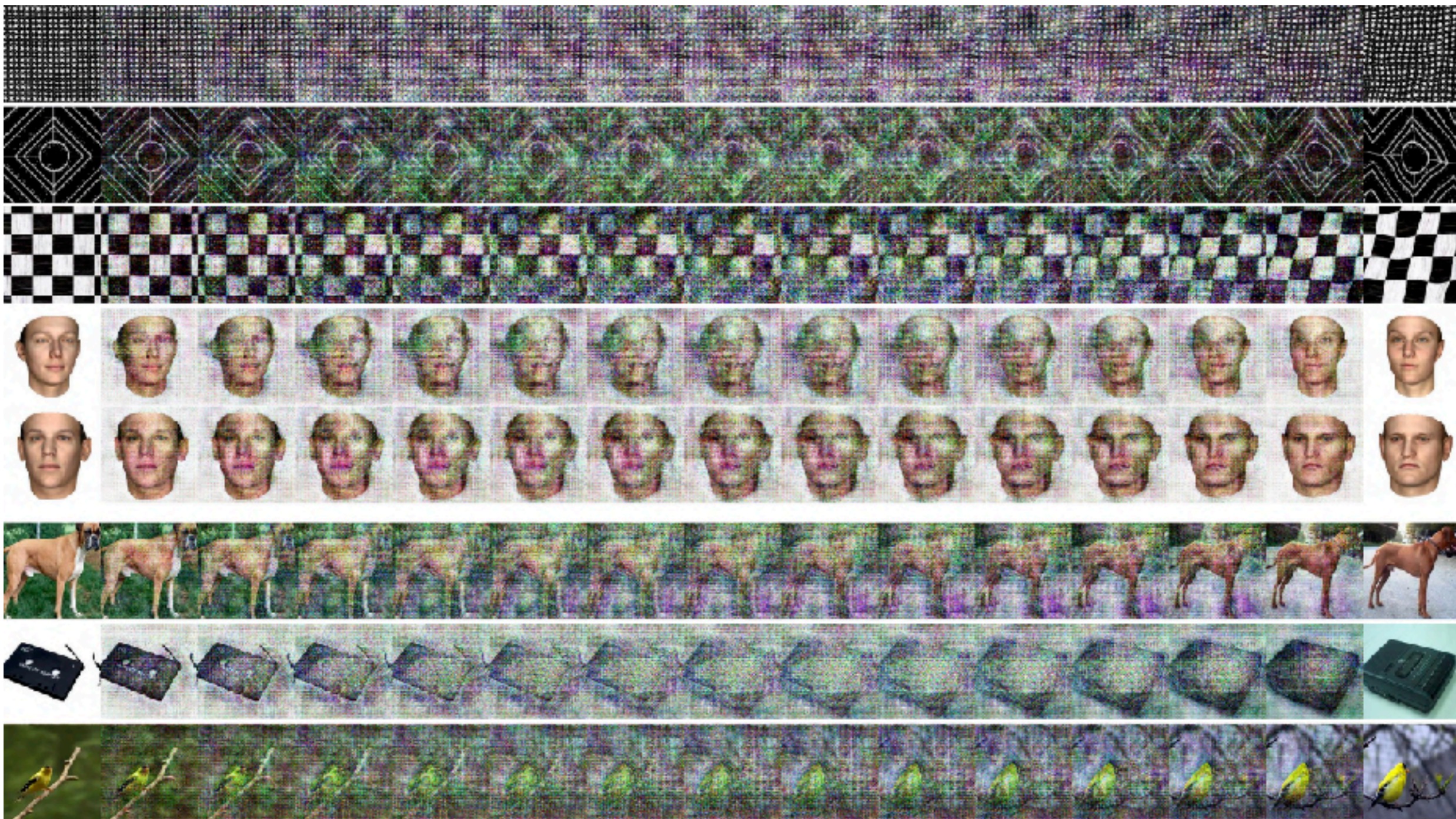
$$I(X; Y) \geq I(\Phi_1 X; Y) \geq \dots \geq I(\Phi_J X; Y)$$

... but "reveal" Y

They propose to introduce:

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

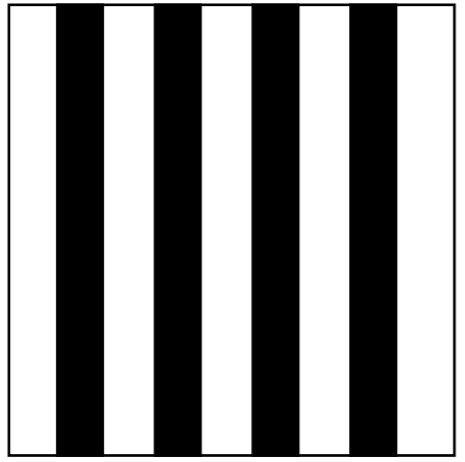
- But one can easily build invertible CNNs...



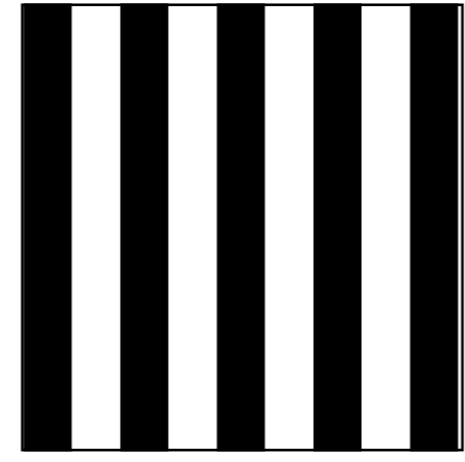
Ref.: i-Revnet, deep invertible networks Jacobsen, Smeulder and EO

Invariant Representations and Deep Learning

Translation



x



y

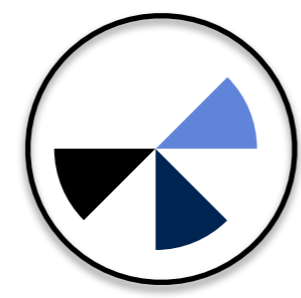


$$\|x - y\|_2 = 2$$

Rotation



x



y

Averaging is the key to get invariants

High dimensionality issues

- Translation invariance? Why not:

$$\Phi x(\omega) = |\hat{x}(\omega)|$$

- Translation invariance? Why not:

$$\Phi x(\omega) = |\hat{x}(\omega)|$$

Doesn't work!

- Translation invariance? Why not:

$$\Phi x(\omega) = |\hat{x}(\omega)|$$

Deformations

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

Doesn't work!

Let $x(u) = e^{i\omega_0 u - \frac{1}{2}u^2}$ and $\tau(u) = su, s > 0$

- Translation invariance? Why not:

$$\Phi x(\omega) = |\hat{x}(\omega)|$$

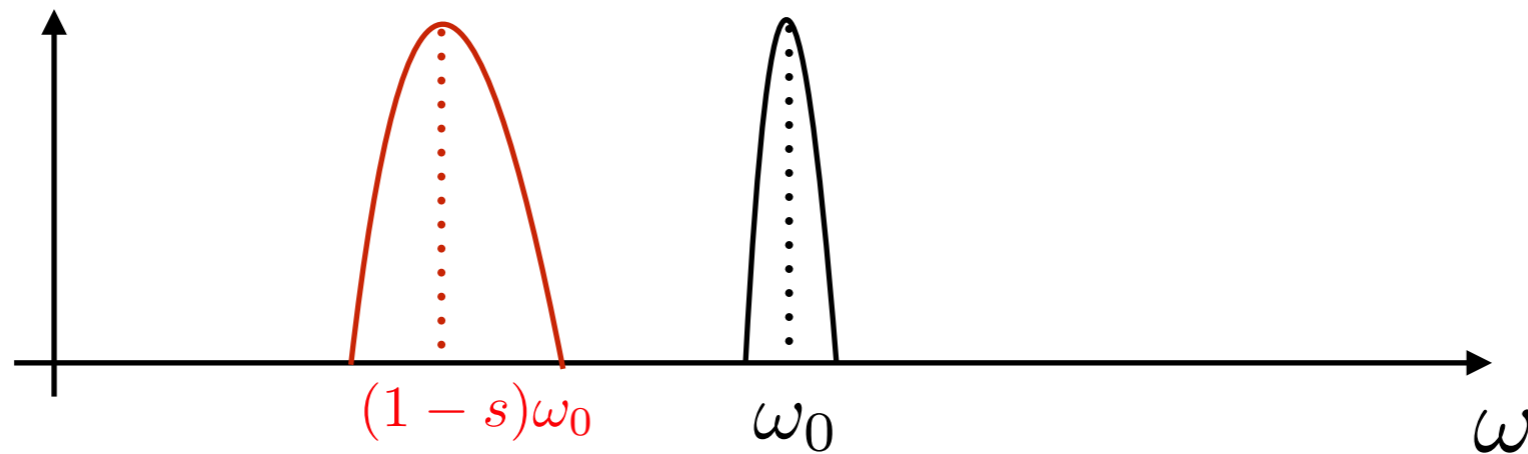
Deformations

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

Doesn't work!

Let $x(u) = e^{i\omega_0 u - \frac{1}{2}u^2}$ and $\tau(u) = su, s > 0$

then:



and:

$$\|\Phi x_\tau - \Phi x\| \gtrsim \omega_0 s = \|\nabla \tau\| \omega_0$$

which for a fixed s diverges quickly...

- We say that L is covariant with W if $WL = LW$

- We say that L is covariant with W if $WL = LW$
- We say that A is invariant to L if $AL = A$

- We say that L is covariant with W if $WL = LW$
- 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:

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

- We say that L is covariant with W if $WL = LW$
- 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:

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

- We say that L is covariant with W if $WL = LW$
- 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 

- Interestingly, CNNs often incorporate some poolings \mathcal{P} , which satisfy for $\|I - \mathcal{L}\| \ll 1$: $\mathcal{P}\mathcal{L} \approx \mathcal{P}$.

- 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)

- 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 that commutes with deformations, ie

$$[\rho, L] = 0 \quad \text{iff} \quad \forall x = (x_1, \dots, x_d), \rho(x) = (\rho(x_1), \dots, \rho(x_d))$$

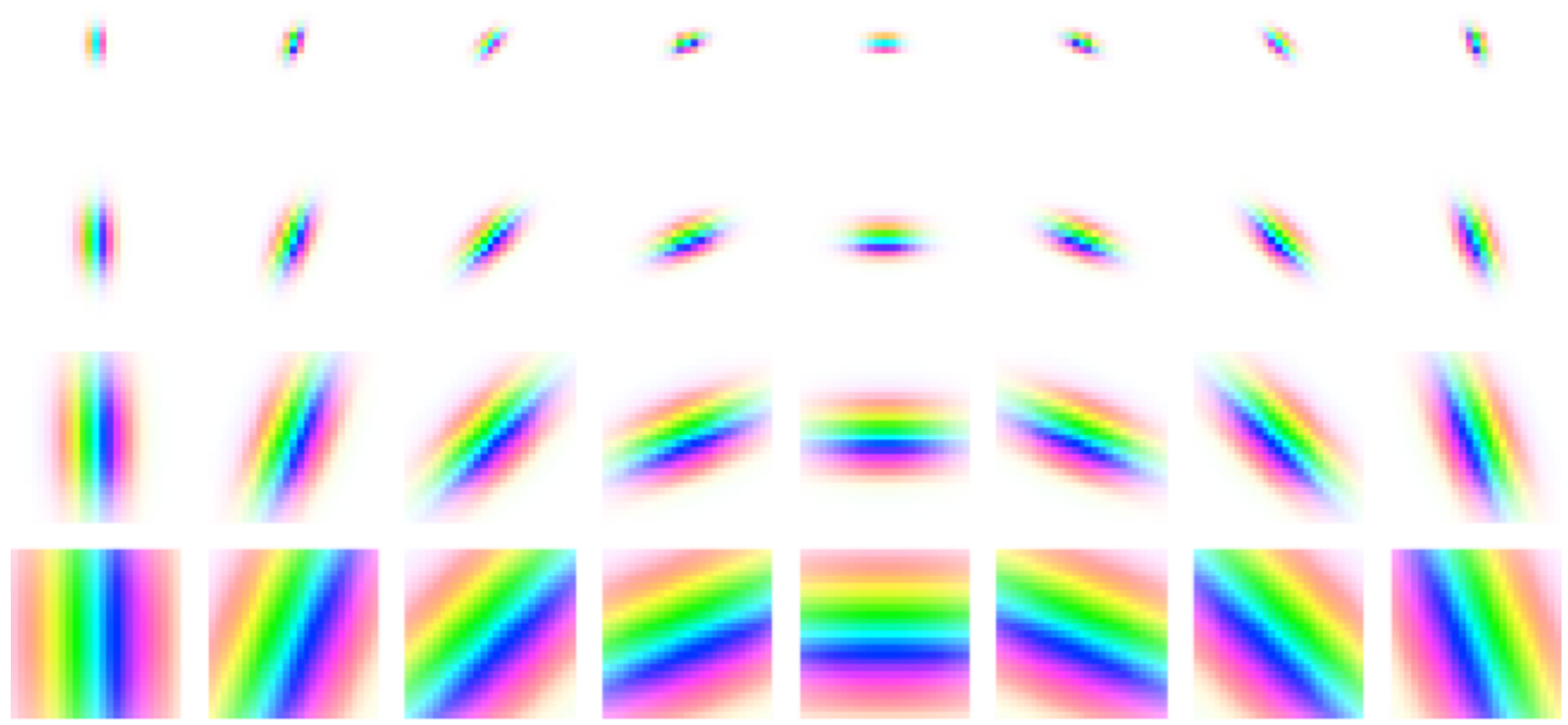
- ψ is a wavelet iff $\int \psi(u)du = 0$ and $\int |\psi|^2(u)du < \infty$
- Typically localised in space and frequency.

- Rotation, dilation of a wavelets:

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

- Design wavelets selective to **rotation** variabilities.





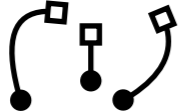
$$\psi(u) = \frac{1}{2\pi\sigma} e^{-\frac{\|u\|^2}{2\sigma}} (e^{i\xi \cdot u} - \kappa)$$

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

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

The Gabor wavelet

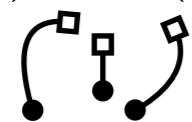
Deformations
 $L_\tau x(u) = x(u - \tau(u))$



Ref.: Group Invariant Scattering, Mallat S



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



- Analytic wavelets permit to build stable invariants

to:

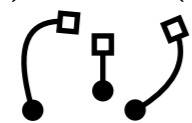
Ref.: Group Invariant Scattering, Mallat S

- small translations by a :



- small deformations:

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

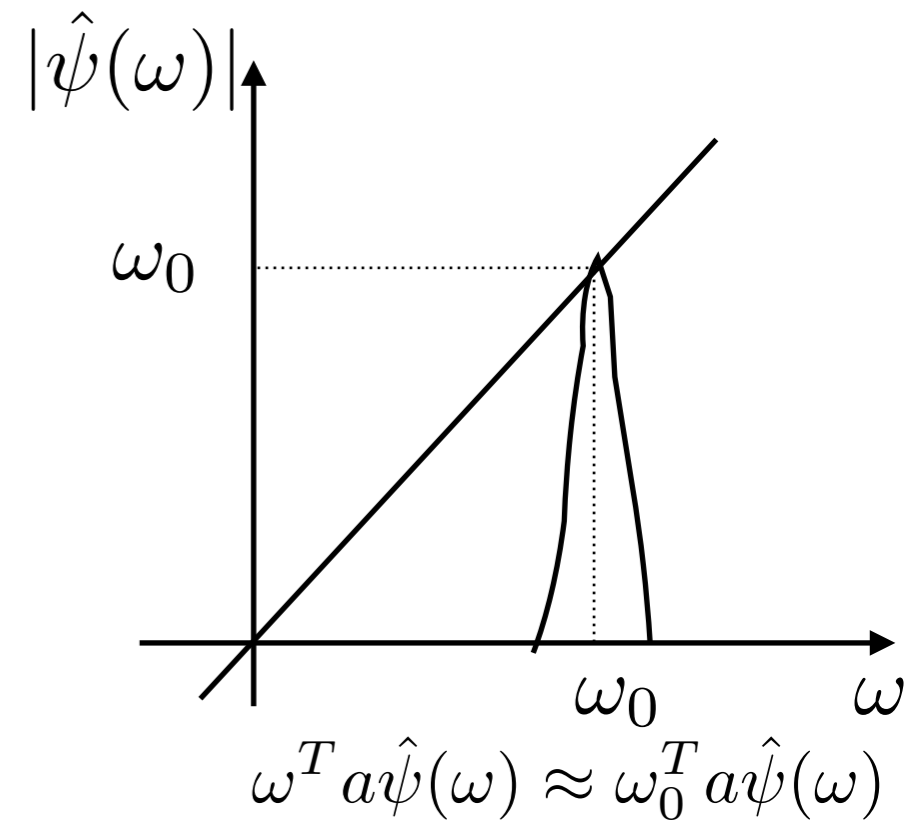


- Analytic wavelets permit to build stable invariants

to:

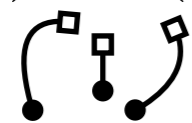
- small translations by a :

Ref.: Group Invariant Scattering, Mallat S



- small deformations:


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



- Analytic wavelets permit to build stable invariants to:

Ref.: Group Invariant Scattering, Mallat S

- small translations by a :

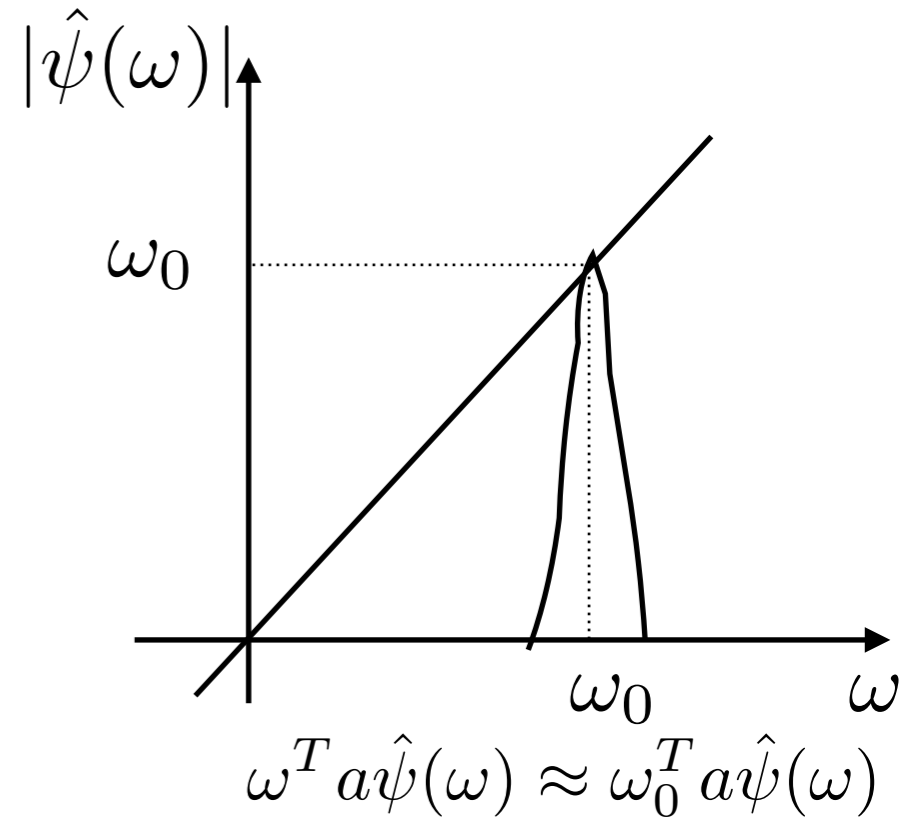


$$\widehat{L_a x \star \psi}(\omega) = e^{i\omega^T a} \hat{x}(\omega) \hat{\psi}(\omega)$$

$$= \sum_n \frac{(i\omega^T a)^n}{n!} \hat{x}(\omega) \hat{\psi}(\omega)$$

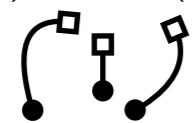
$$\approx \sum_n \frac{(i\omega_0^T a)^n}{n!} \hat{x}(\omega) \hat{\psi}(\omega)$$

$$= e^{i\omega_0^T a} \widehat{x \star \psi}(\omega)$$



- small deformations:


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



- Analytic wavelets permit to build stable invariants to:

Ref.: Group Invariant Scattering, Mallat S

- small translations by a :

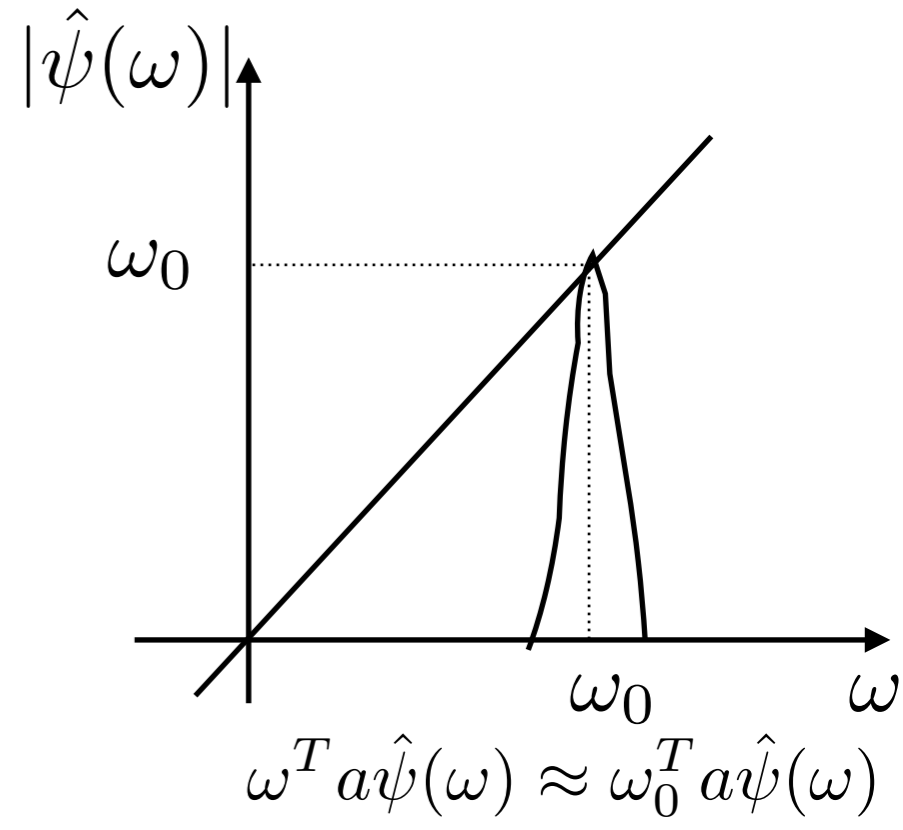


$$\widehat{L_a x \star \psi}(\omega) = e^{i\omega^T a} \hat{x}(\omega) \hat{\psi}(\omega)$$

$$= \sum_n \frac{(i\omega^T a)^n}{n!} \hat{x}(\omega) \hat{\psi}(\omega)$$

$$\approx \sum_n \frac{(i\omega_0^T a)^n}{n!} \hat{x}(\omega) \hat{\psi}(\omega)$$

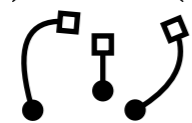
$$= e^{i\omega_0^T a} \widehat{x \star \psi}(\omega)$$



The variability corresponds to a phase multiplication!

- small deformations:


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



- Analytic wavelets permit to build stable invariants to:

Ref.: Group Invariant Scattering, Mallat S

- small translations by a :

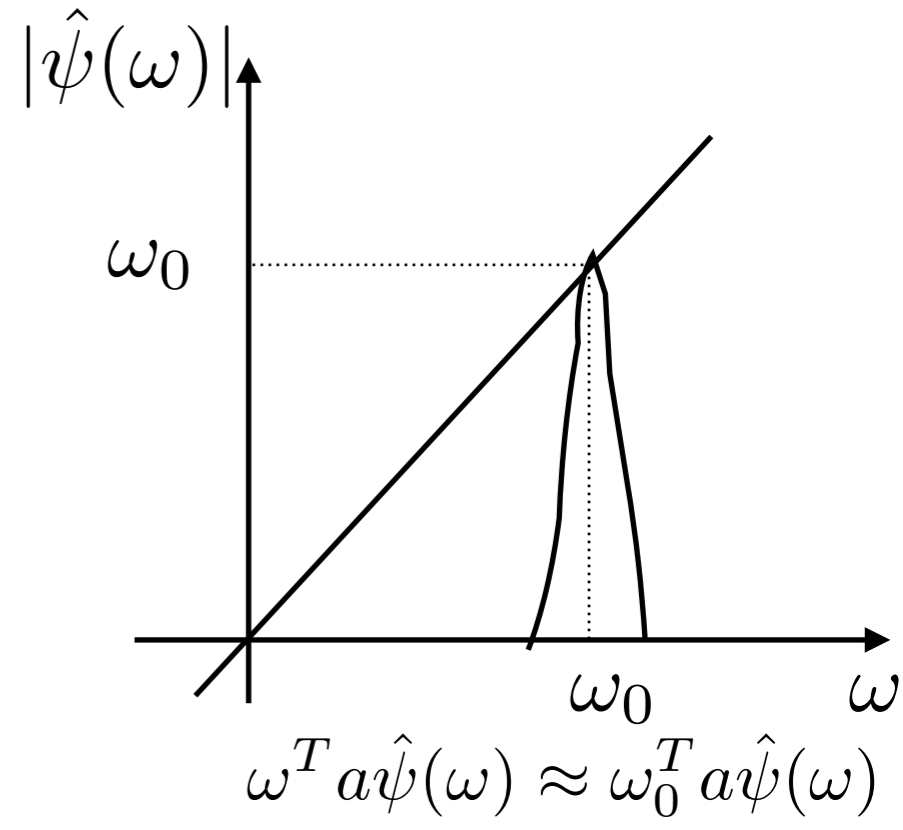


$$\widehat{L_a x \star \psi}(\omega) = e^{i\omega^T a} \hat{x}(\omega) \hat{\psi}(\omega)$$

$$= \sum_n \frac{(i\omega^T a)^n}{n!} \hat{x}(\omega) \hat{\psi}(\omega)$$

$$\approx \sum_n \frac{(i\omega_0^T a)^n}{n!} \hat{x}(\omega) \hat{\psi}(\omega)$$

$$= e^{i\omega_0^T a} \widehat{x \star \psi}(\omega)$$



The variability corresponds to a phase multiplication!

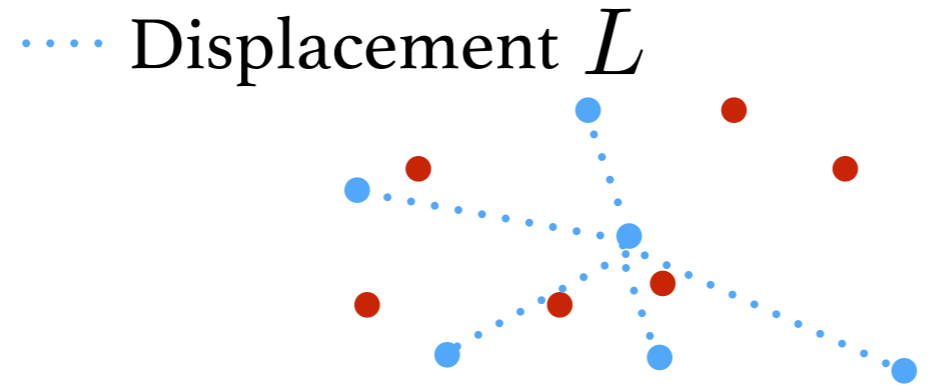
- small deformations:

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

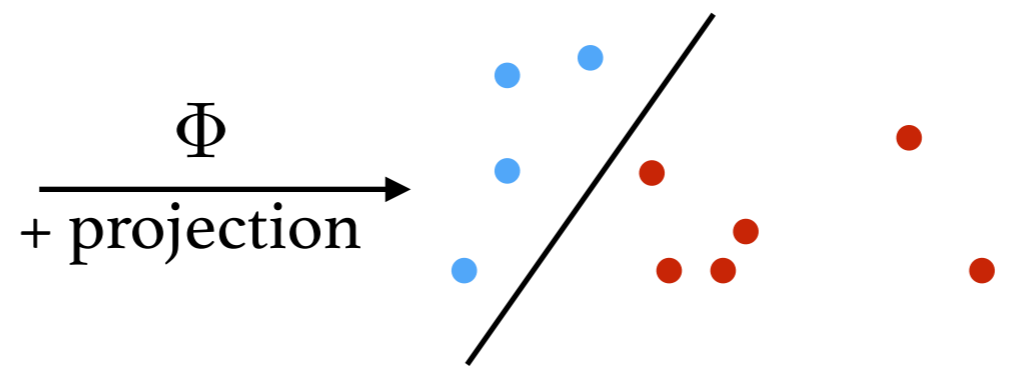
- Weak differentiability property:

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

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



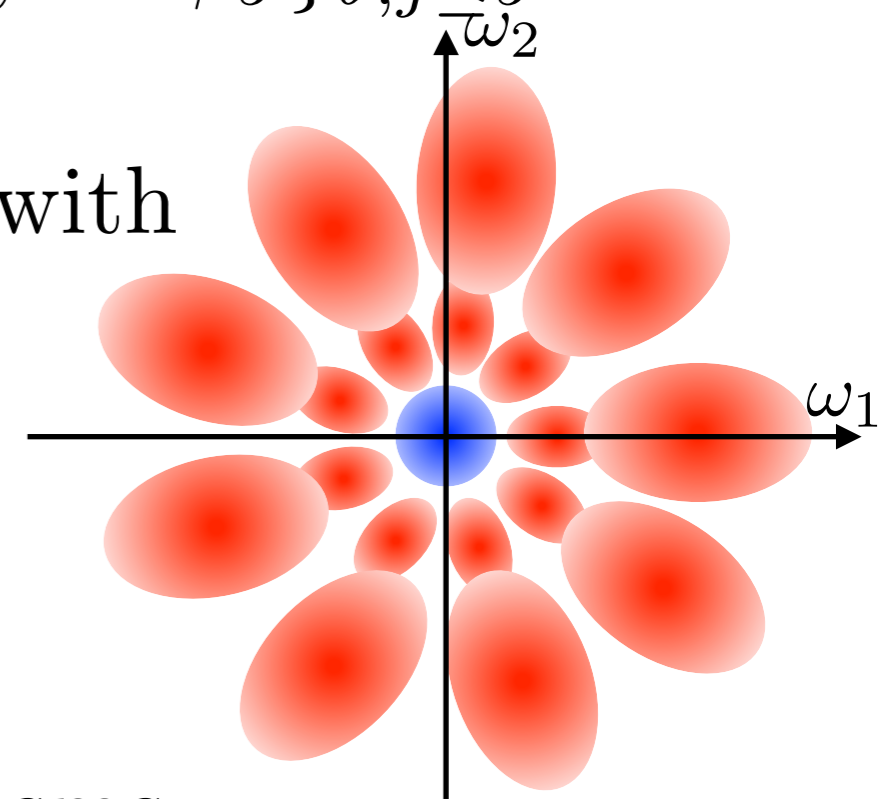
- A linear projection (to kill L) build an invariant



- 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 :

$$WL_a = L_aW$$

- Nearly commutes with diffeomorphisms

$$\|[W, L_\tau]\| \leq C\|\nabla\tau\|$$

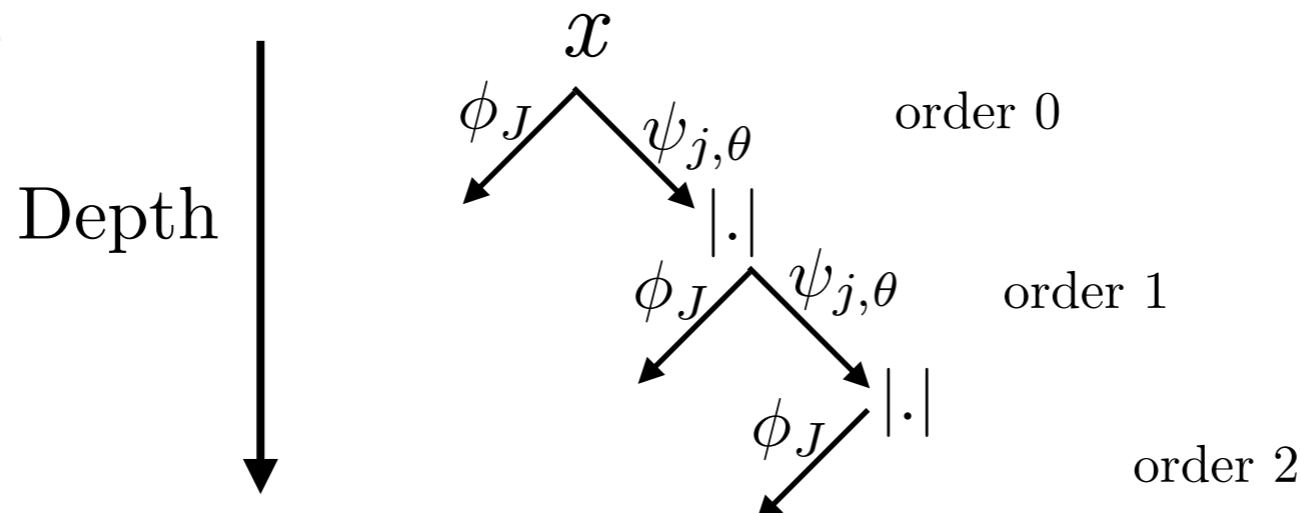
Ref.: Group Invariant Scattering, Mallat S

- A good baseline to describe an image!

- Scattering transform at scale J is the cascading of complex WT with modulus non-linearity, followed by a low pass-filtering:

Ref.: Group Invariant Scattering, Mallat S

$$S_J x = \{x \star \phi_J, |x \star \psi_{j_1, \theta_1}| \star \phi_J, ||x \star \psi_{j_1, \theta_1}| \star \psi_{j_2, \theta_2}| \star \phi_J \}$$

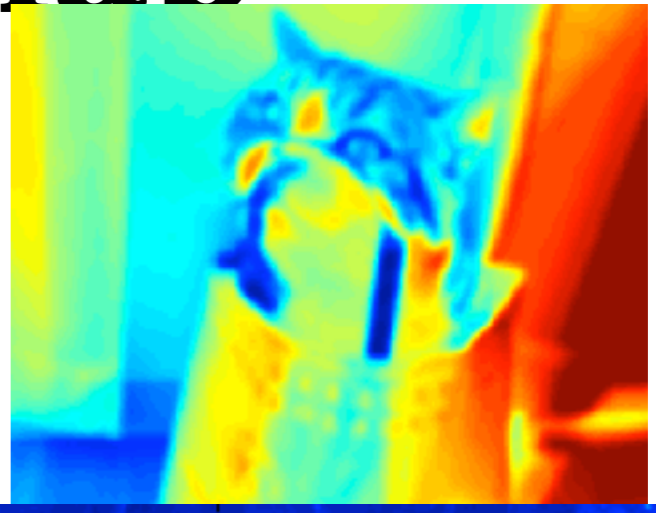
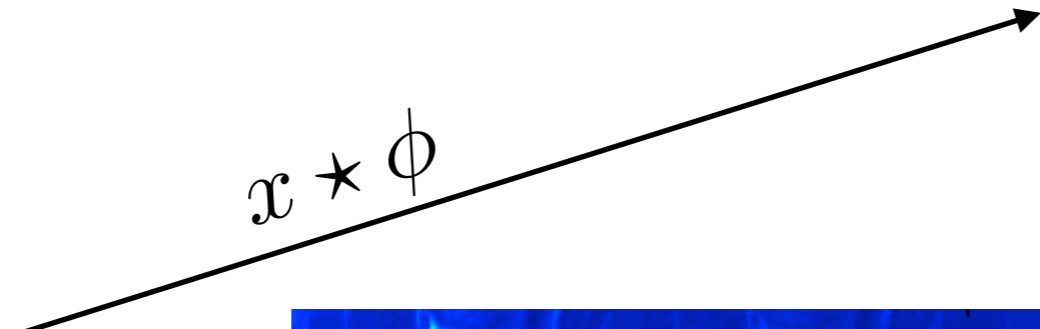


- Mathematically** well defined for a large class of wavelets.

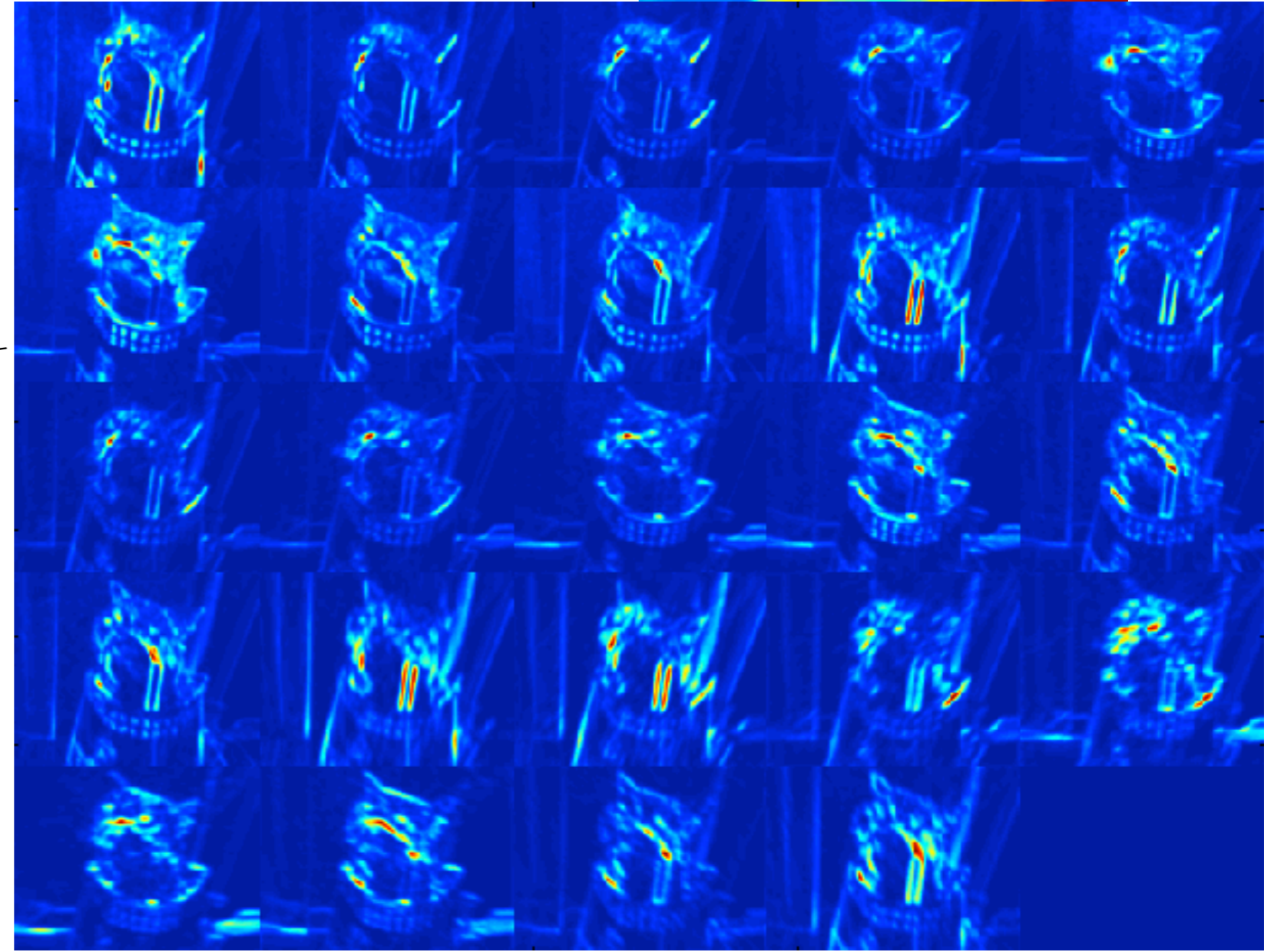
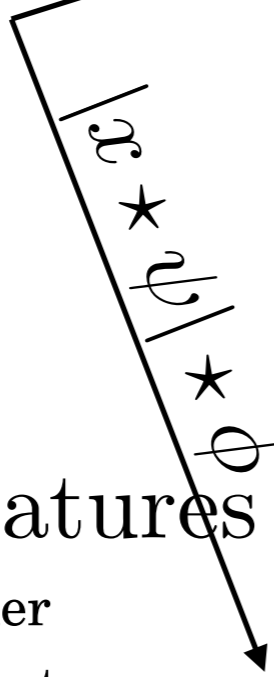
Feature map



x



- Several features
1st order coefficients



Example of Scattering coefficients

- Assume it is possible to find h and g such that

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

- Set:

$$x_j(u, 0) = x \star \phi_j(u) = h \star (x \star \phi_{j-1})(2u) \quad \text{and}$$

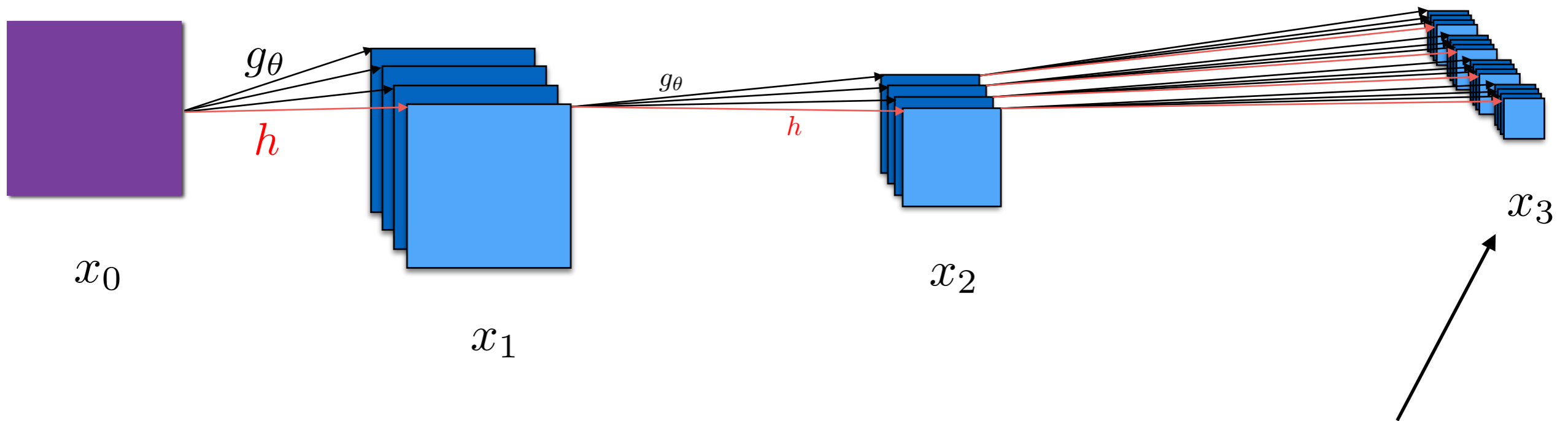
$$x_j(u, \theta) = x \star \psi_{j,\theta}(u) = g_\theta \star (x \star \phi_{j-1})(2u)$$

- The WT is then given by $Wx = \{x_j(\cdot, \theta), x_J(\cdot, 0)\}_{j \leq J, \theta}$
- A WT can be interpreted as a **deep cascade** of linear operator, which is approximatively verified for the Gabor Wavelets.

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

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

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



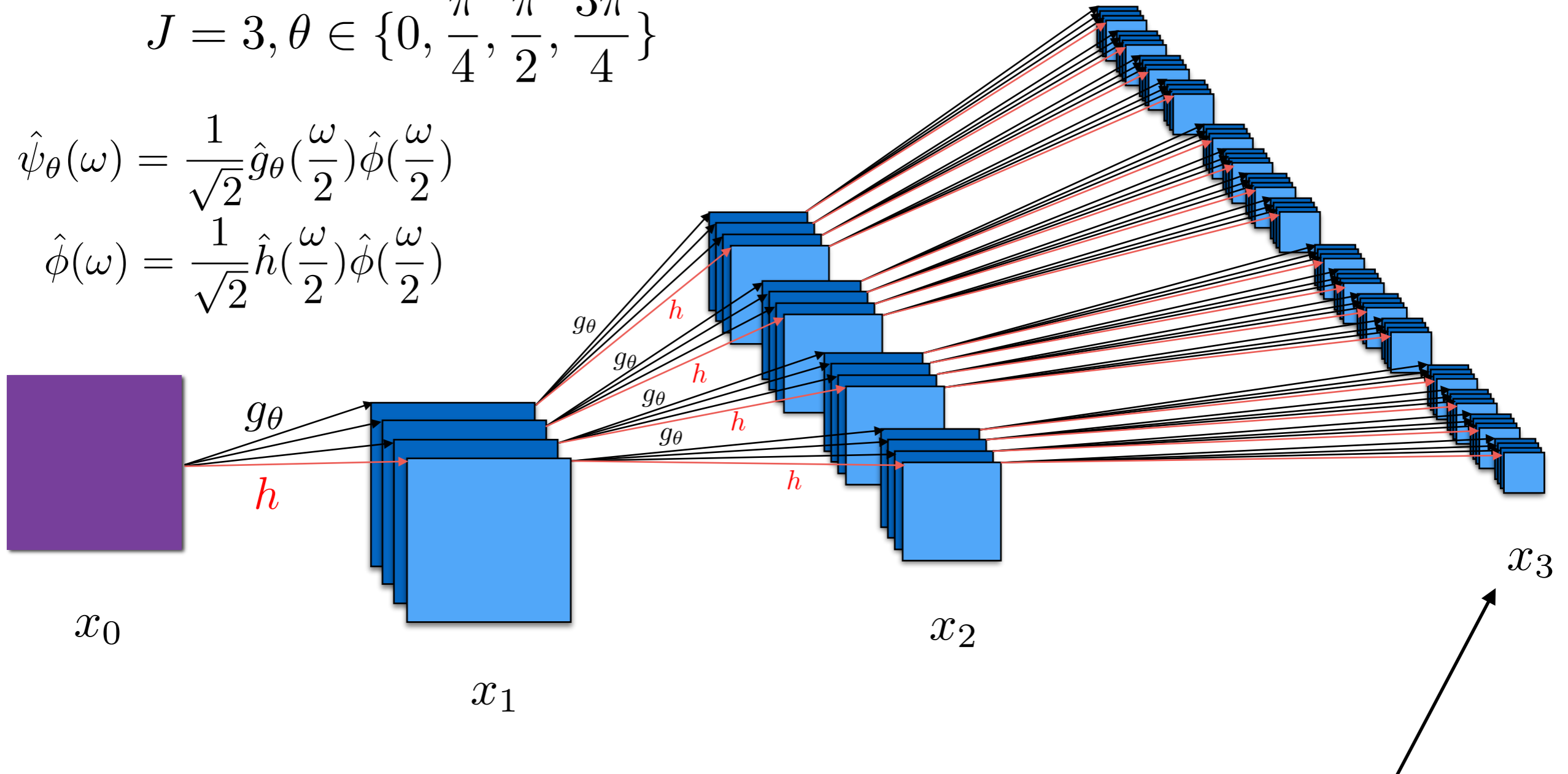
Scattering coefficients are only at the output

Scattering as a CNN

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

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

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



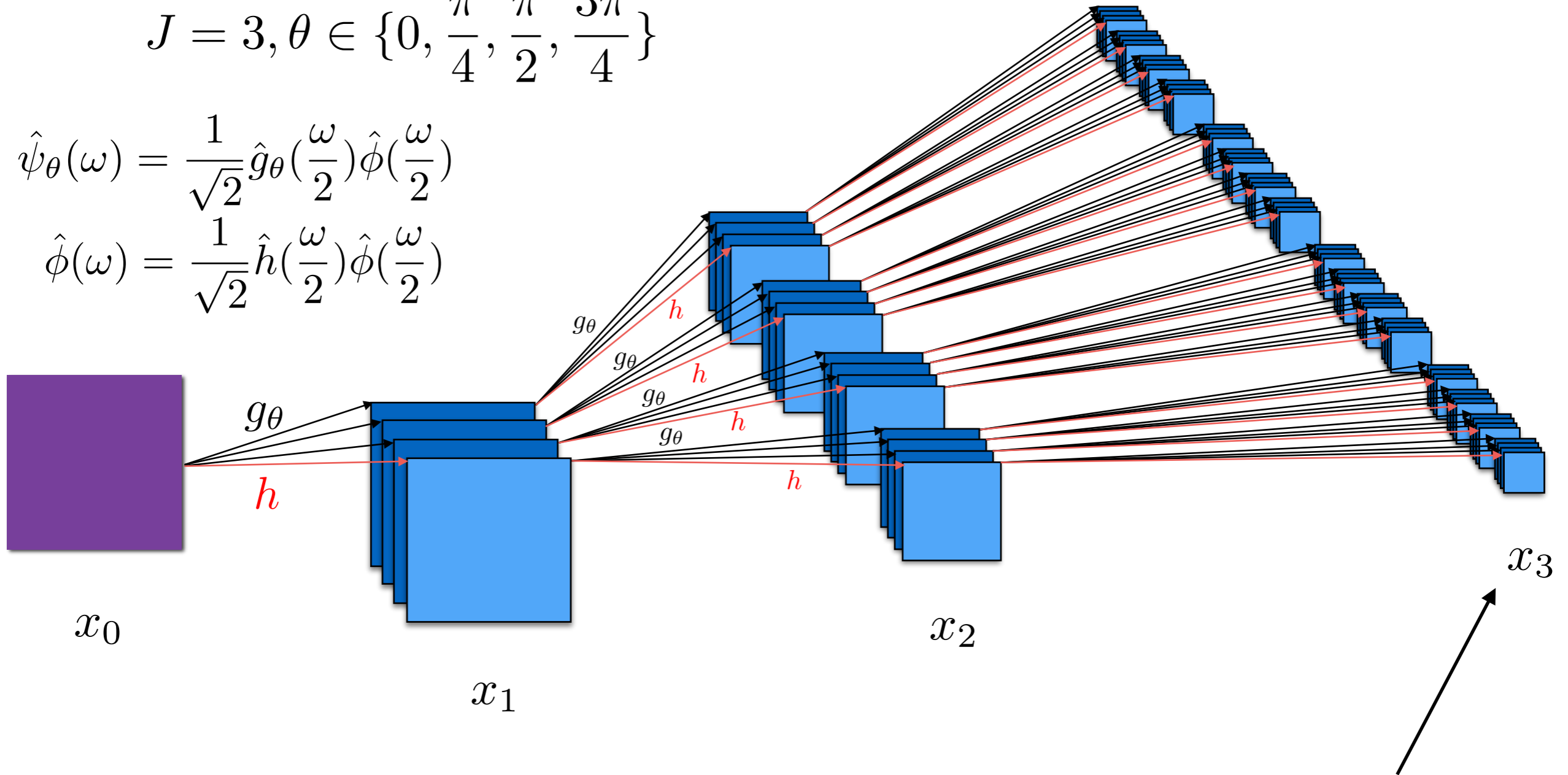
Scattering coefficients are only at the output

Scattering as a CNN

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

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

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



○ Modulus

Scattering coefficients are only at the output

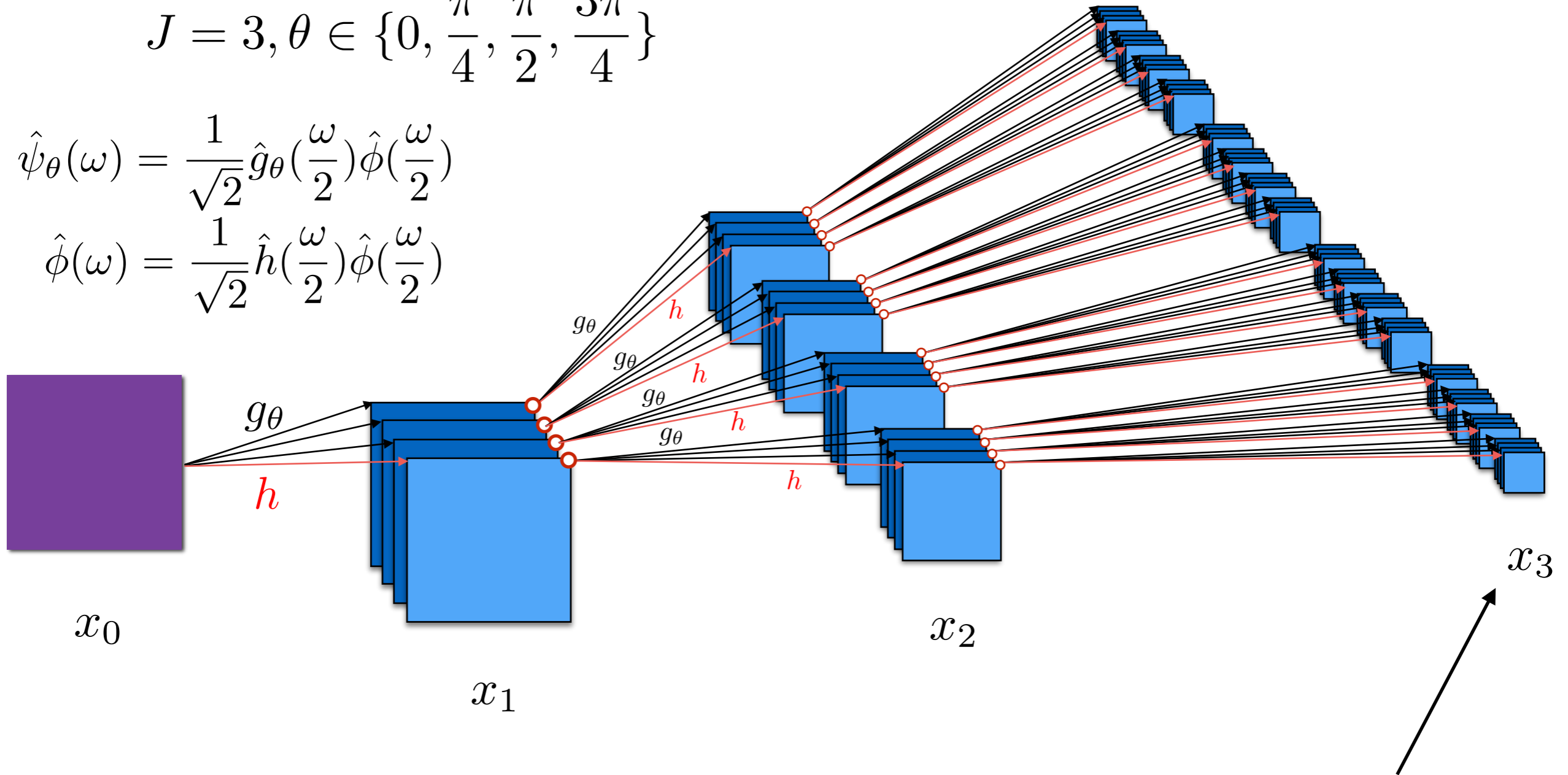
Scattering as a CNN

Ref.: Deep Roto-Translation Scattering for Object Classification. EO and S Mallat
Recursive Interferometric Representations, S Mallat

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

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

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



○ Modulus

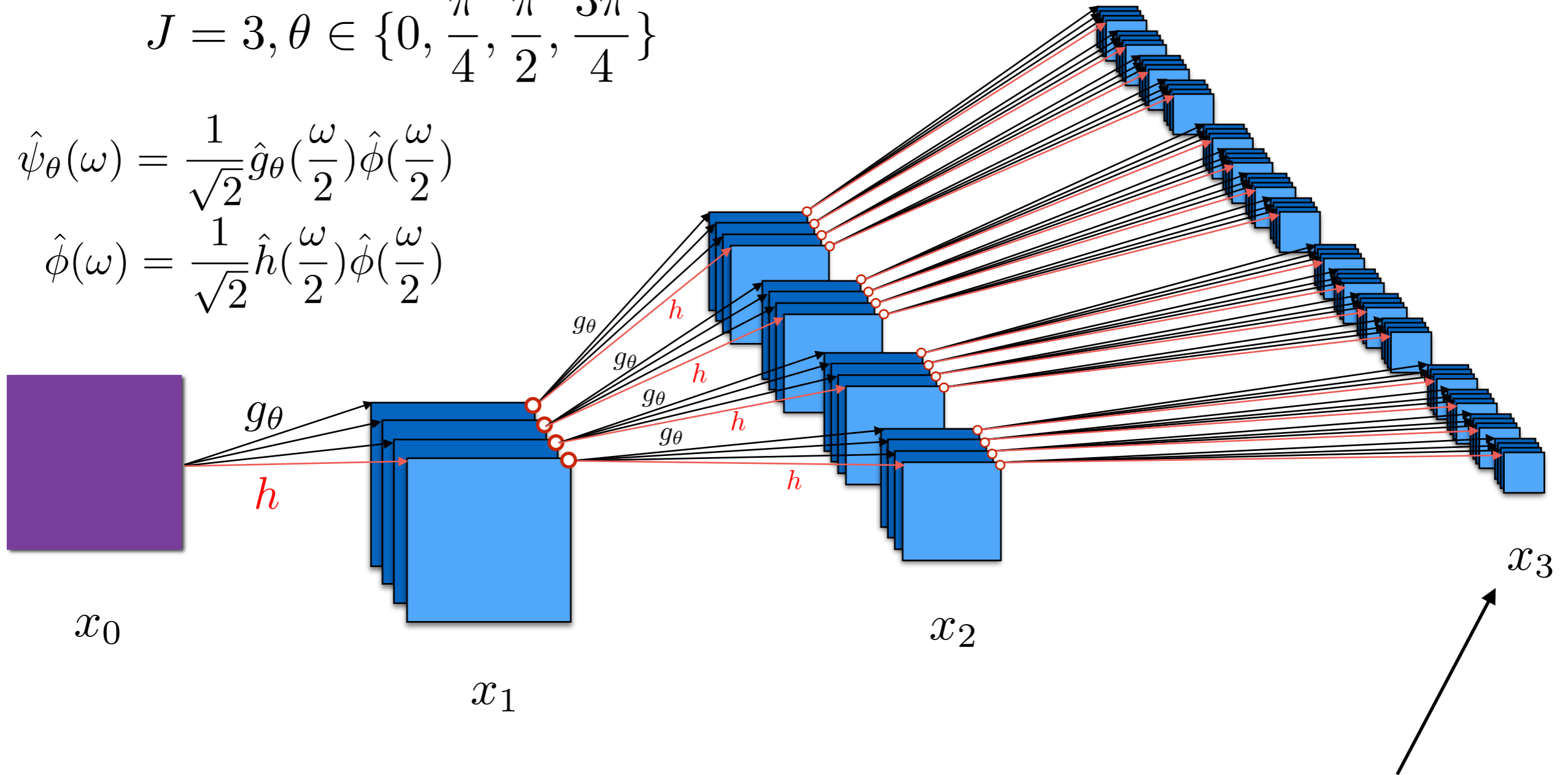
Scattering coefficients are only at the output

Scattering as a CNN

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

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

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



○ Modulus

$$h \geq 0$$

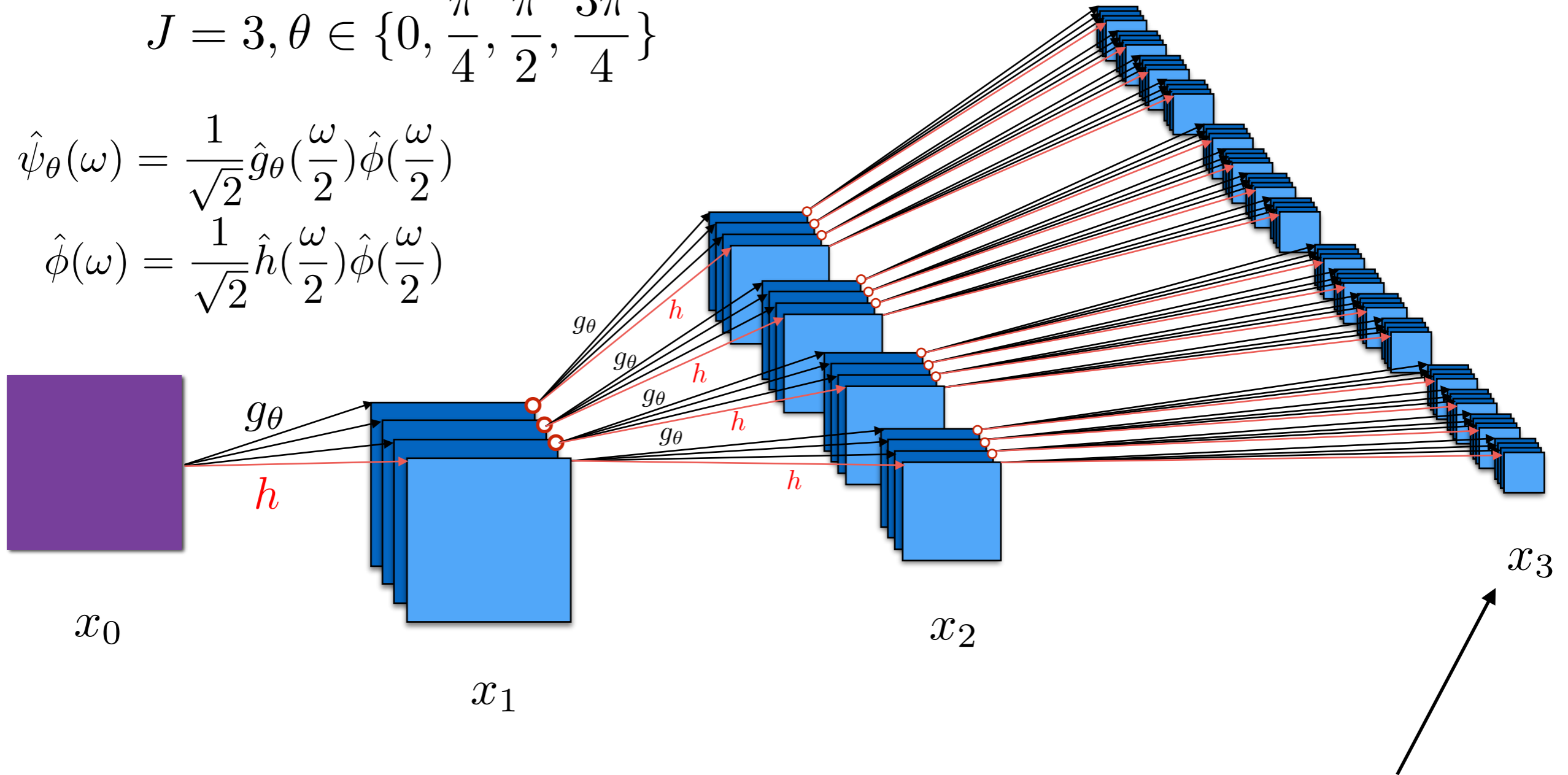
Scattering coefficients are only at the output

Scattering as a CNN

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

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

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



○ Modulus

$$h \geq 0$$

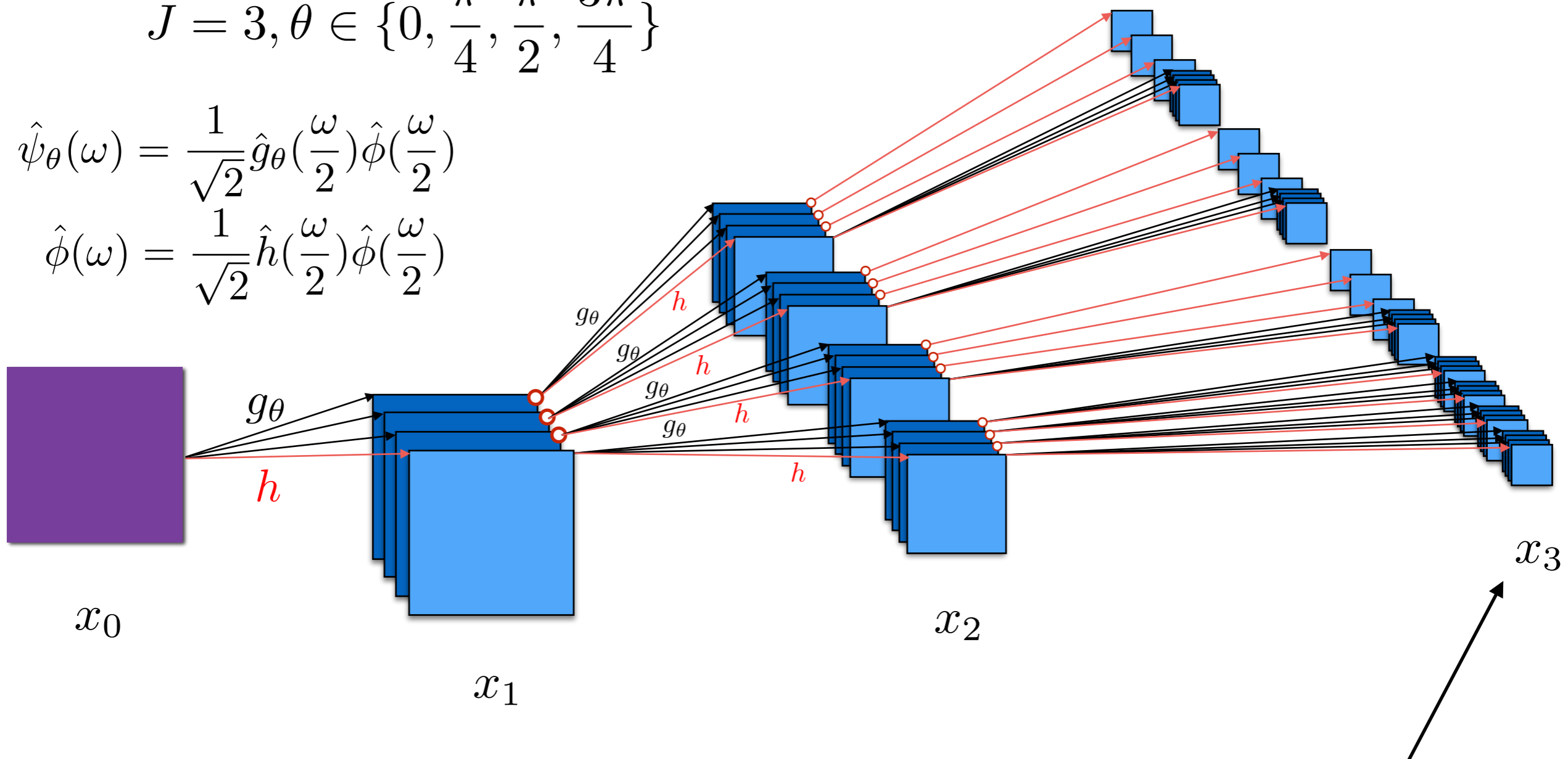
Scattering coefficients are only at the output

Scattering as a CNN

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

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

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



○ Modulus

$$h \geq 0$$

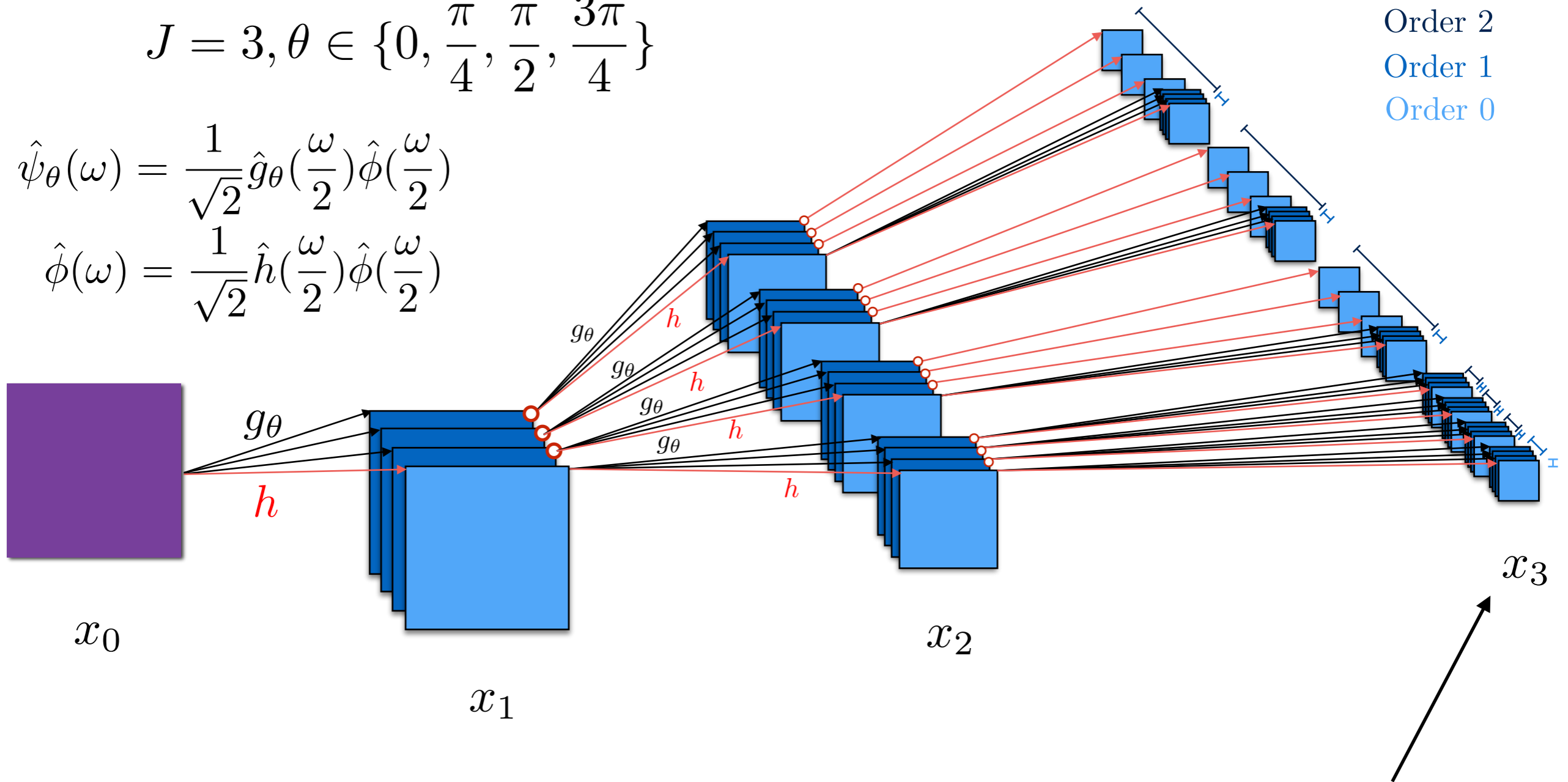
Scattering coefficients are only at the output

Scattering as a CNN

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

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

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



○ Modulus

$$h \geq 0$$

Scattering coefficients are only at the output

Scattering as a CNN

Ref.: Invariant Group Scattering
Mallat S., 2012



Studying:

$$L_\tau A_J x(v) - A_J x(v) = \int_v x(u) (\phi_J(v - \tau(v) - u) - \phi_J(v - u))$$

and upper bounding this kernel leads to:

$$\|L_\tau A_J - A_J\| \leq 2^{-J+d} \|\tau\|_\infty \|\nabla \phi\|_1$$

Ref.: Invariant Group Scattering
Mallat S., 2012



Studying:

$$L_\tau A_J x(v) - A_J x(v) = \int_v x(u) (\phi_J(v - \tau(v) - u) - \phi_J(v - u))$$

and upper bounding this kernel leads to:

$$\|L_\tau A_J - A_J\| \leq 2^{-J+d} \|\tau\|_\infty \|\nabla \phi\|_1$$

Ref.: Invariant Group Scattering
Mallat S., 2012

It is slightly more challenging to obtain:

$$\|[W_j, L_\tau]\| \leq C J \|\nabla \tau\|_\infty \quad \text{where } W_j x(v) = \{x \star \psi_j(v)\}_{0 \leq j \leq J}$$



Studying:

$$L_\tau A_J x(v) - A_J x(v) = \int_v x(u) (\phi_J(v - \tau(v) - u) - \phi_J(v - u))$$

and upper bounding this kernel leads to:

$$\|L_\tau A_J - A_J\| \leq 2^{-J+d} \|\tau\|_\infty \|\nabla \phi\|_1$$

Ref.: Invariant Group Scattering
Mallat S., 2012

It is slightly more challenging to obtain:

$$\|[W_j, L_\tau]\| \leq C J \|\nabla \tau\|_\infty \quad \text{where } W_j x(v) = \{x \star \psi_j(v)\}_{0 \leq j \leq J}$$

For order 1:

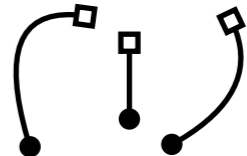
$$\begin{aligned} A_J |W_J| L_\tau - A_J |W_J| &= A_J |W_J| L_\tau - A_J L_\tau |W_J| + A_J L_\tau |W_J| \\ &\quad - L_\tau A_J |W_J| + L_\tau A_J |W_J| - A_J |W_J| \\ &= A_J |[W_J, L_\tau]| + [A_J, L_\tau] |W_J| + (L_\tau A_J - A_J) |W_J| \end{aligned}$$



and we iterate... a tighter bound can be obtained by avoiding redundancy.

Transform

Deformations

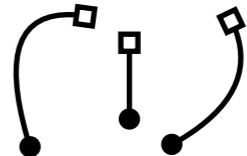
$$L_\tau x(u) = x(u - \tau(u))$$


Transform

- Scattering is stable:

$$\|S_J x - S_J y\| \leq \|x - y\|$$

Deformations

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


Transform

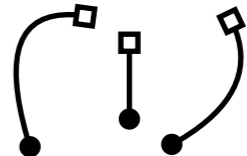
- Scattering is stable:

$$\|S_J x - S_J y\| \leq \|x - y\|$$

- Linearize small deformations:

$$\|S_J L_\tau x - S_J x\| \leq C \|\nabla \tau\| \|x\|$$

Deformations

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


Transform

- Scattering is stable:

$$\|S_J x - S_J y\| \leq \|x - y\|$$

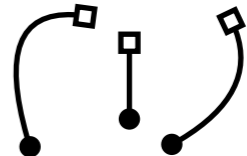
- Linearize small deformations:

$$\|S_J L_\tau x - S_J x\| \leq C \|\nabla \tau\| \|x\|$$

- Invariant to local translation:

$$|a| \ll 2^J \Rightarrow S_J L_a x \approx S_J x$$

Deformations

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


Transform

- Scattering is stable:

$$\|S_J x - S_J y\| \leq \|x - y\|$$

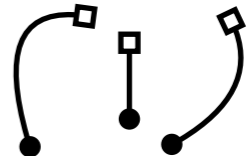
- Linearize small deformations:

$$\|S_J L_\tau x - S_J x\| \leq C \|\nabla \tau\| \|x\|$$

- Invariant to local translation:

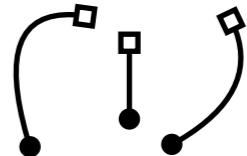
$$|a| \ll 2^J \Rightarrow S_J L_a x \approx S_J x$$

Deformations

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


Transform

Deformations

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


- Scattering is stable:

$$\|S_J x - S_J y\| \leq \|x - y\|$$

- Linearize small deformations:

$$\|S_J L_\tau x - S_J x\| \leq C \|\nabla \tau\| \|x\|$$

- Invariant to local translation:

$$|a| \ll 2^J \Rightarrow S_J L_a x \approx S_J x$$

Ref.: Group Invariant Scattering, Mallat S

- For λ, u , $S_J x(u, \lambda)$ is **covariant** with $SO_2(\mathbb{R})$:

if $\forall u \forall g \in SO_2(\mathbb{R}), g.x(u) \triangleq x(g^{-1}u)$ then,

$$S_J(g.x)(u, \lambda) = S_J x(g^{-1}u, g^{-1}\lambda) \triangleq g.S_J x(u, \lambda)$$

- For a stationary process X (e.g., a texture)

$$E(X \star f) = E(X) \star f$$

- This leads to the Expected Scattering:

$$\bar{S}[\lambda_1] = \mathbb{E}|X \star \psi_{\lambda_1}|$$

$$\bar{S}[\lambda_1, \lambda_2] = \mathbb{E}||X \star \psi_{\lambda_1}| \star \psi_{\lambda_2}|$$

...

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

- For a stationary process X (e.g., a texture)

$$E(X \star f) = E(X) \star f$$

- This leads to the Expected Scattering:

$$\bar{S}[\lambda_1] = \mathbb{E}|X \star \psi_{\lambda_1}|$$

$$\bar{S}[\lambda_1, \lambda_2] = \mathbb{E}||X \star \psi_{\lambda_1}| \star \psi_{\lambda_2}|$$

...

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

- For a stationary process X (e.g., a texture)

$$E(X \star f) = E(X) \star f$$

- This leads to the Expected Scattering:

$$\bar{S}[\lambda_1] = \mathbb{E}|X \star \psi_{\lambda_1}|$$

Modulus is important
because it can be 0!

$$\bar{S}[\lambda_1, \lambda_2] = \mathbb{E}||X \star \psi_{\lambda_1}| \star \psi_{\lambda_2}|$$

...

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

- For a stationary process X (e.g., a texture)

$$E(X \star f) = E(X) \star f$$

- This leads to the Expected Scattering:

$$\bar{S}[\lambda_1] = \mathbb{E}|X \star \psi_{\lambda_1}|$$

Modulus is important
because it can be 0!

$$\bar{S}[\lambda_1, \lambda_2] = \mathbb{E}||X \star \psi_{\lambda_1}| \star \psi_{\lambda_2}|$$

can be estimated via an unbiased estimator:

$$S[\lambda_1, \lambda_2]X = \int ||X \star \psi_{\lambda_1}| \star \psi_{\lambda_2}|$$

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

- For a stationary process X (e.g., a texture)

$$E(X \star f) = E(X) \star f$$

- This leads to the Expected Scattering:

$$\bar{S}[\lambda_1] = \mathbb{E}|X \star \psi_{\lambda_1}|$$

Modulus is important
because it can be 0!

$$\bar{S}[\lambda_1, \lambda_2] = \mathbb{E}||X \star \psi_{\lambda_1}| \star \psi_{\lambda_2}|$$

can be estimated via an unbiased estimator:

$$S[\lambda_1, \lambda_2]X = \int ||X \star \psi_{\lambda_1}| \star \psi_{\lambda_2}|$$

Energy is preserved:

$$\|\bar{S}X\|^2 = \mathbb{E}|X|^2$$

A successful representation⁶⁸ in image classification

A successful representation⁶⁸ in image classification

A successful representation⁶⁸ in image classification

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

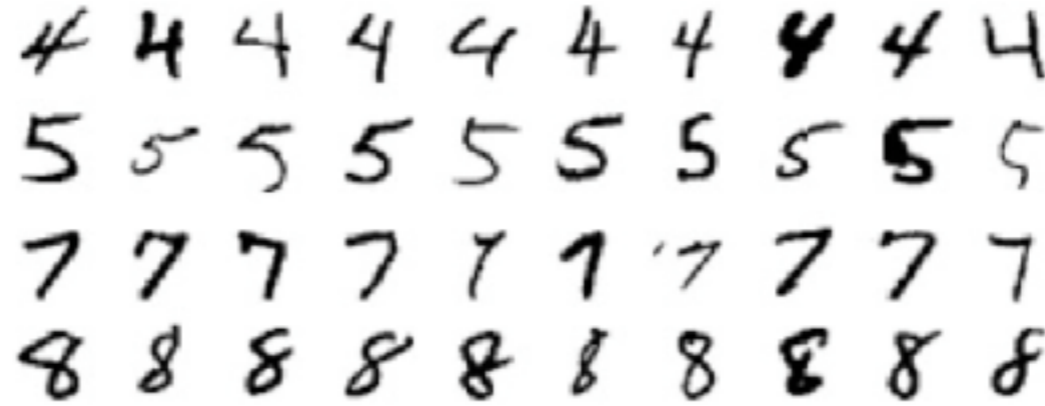
- Successfully used in several applications:

A successful representation⁶⁸ in image classification

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

- Successfully used in several applications:

- Digits

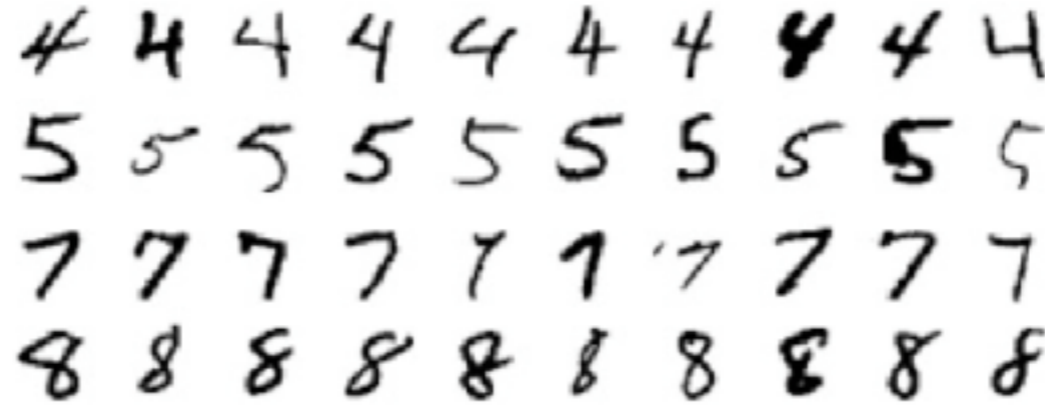


A successful representation⁶⁸ in image classification

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

- Successfully used in several applications:

- Digits

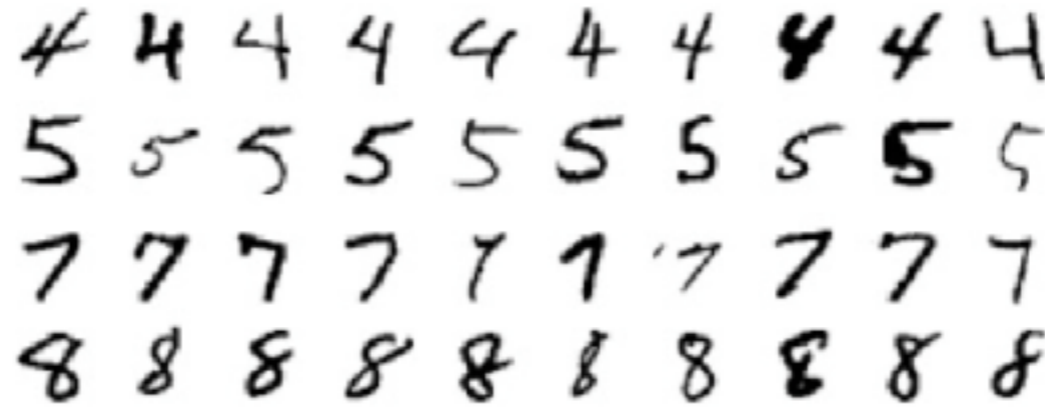


A successful representation⁶⁸ in image classification

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

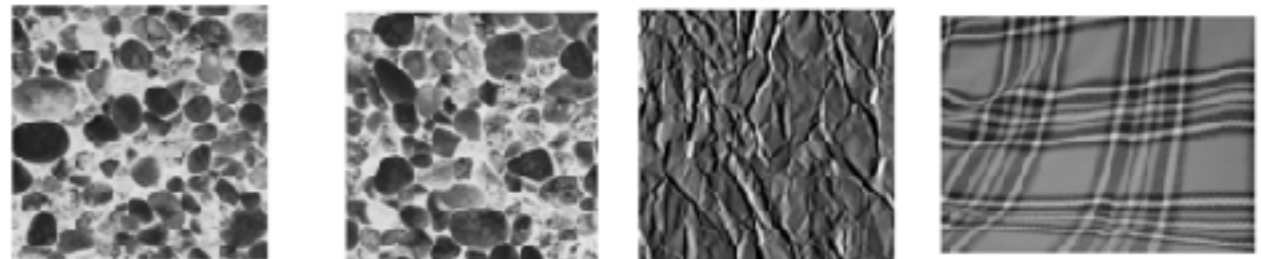
- Successfully used in several applications:

- Digits



- Textures

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



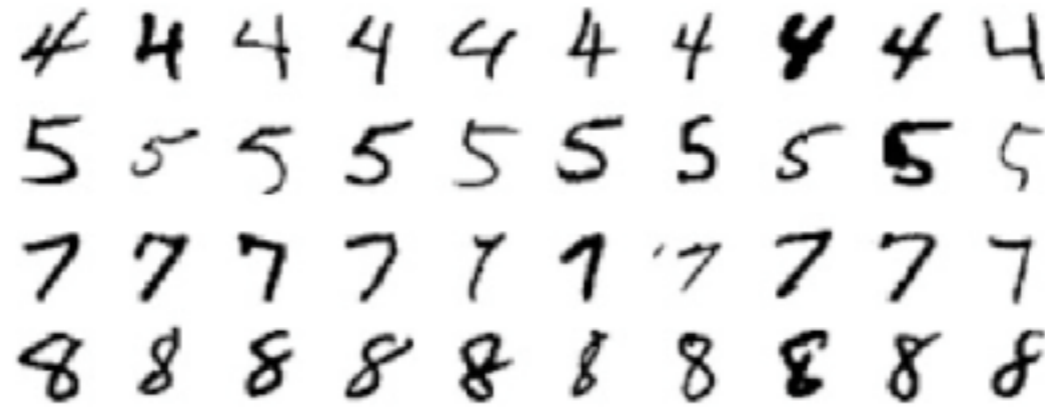
A successful representation⁶⁸ in image classification

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

- Successfully used in several applications:

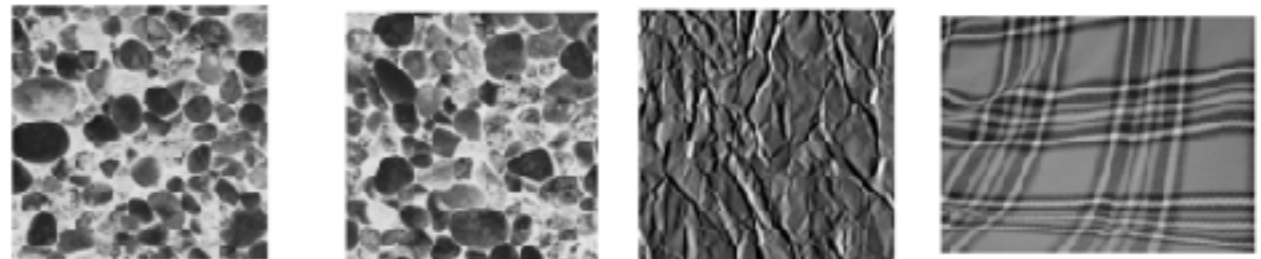
All variabilities
are known

- Digits



- Textures

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

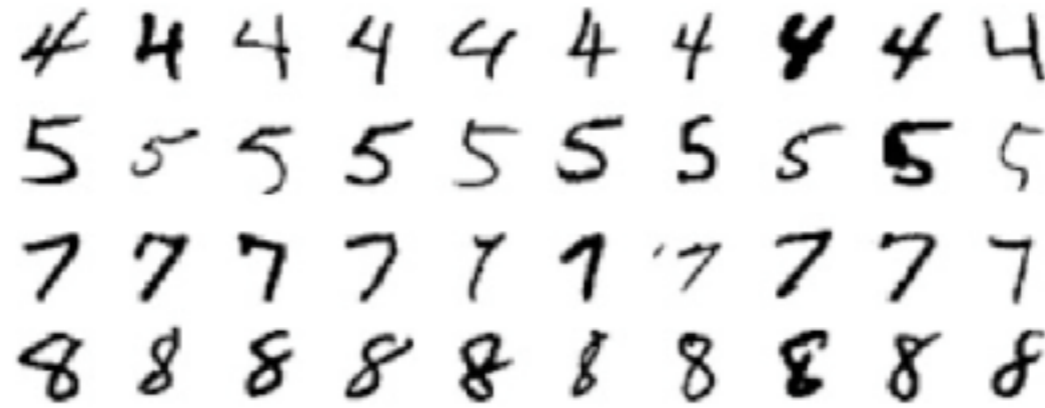


A successful representation⁶⁸ in image classification

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

- Successfully used in several applications:

- Digits

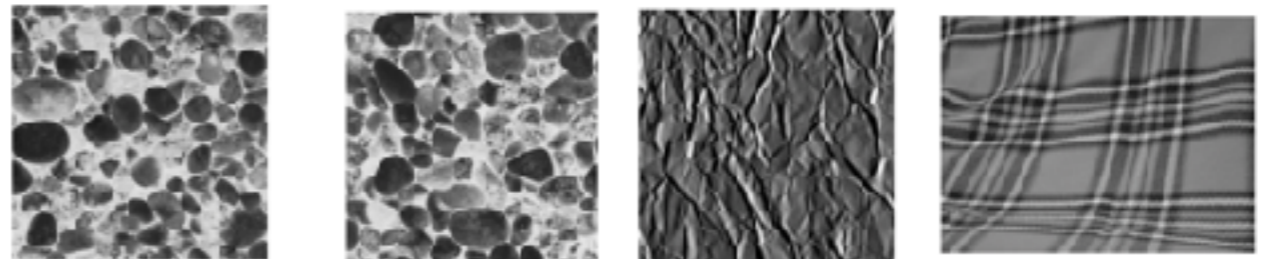


All variabilities
are known

Small deformations
+ Translation

- Textures

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

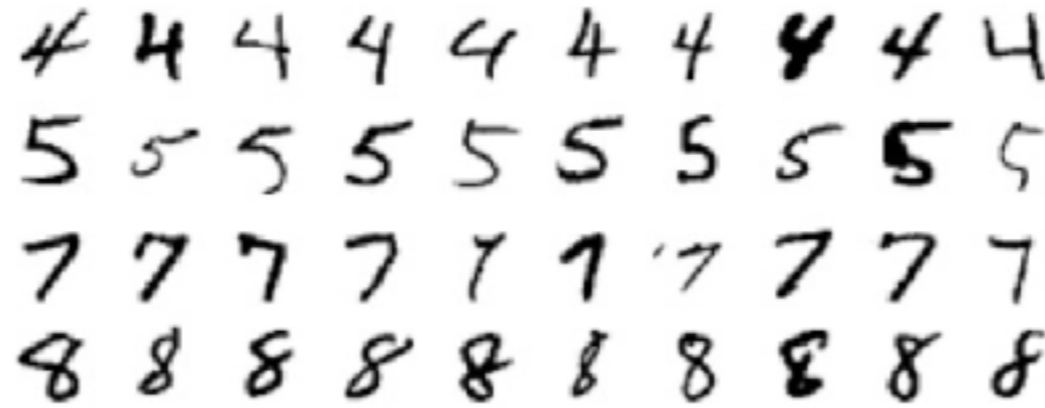


A successful representation⁶⁸ in image classification

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

- Successfully used in several applications:

- Digits

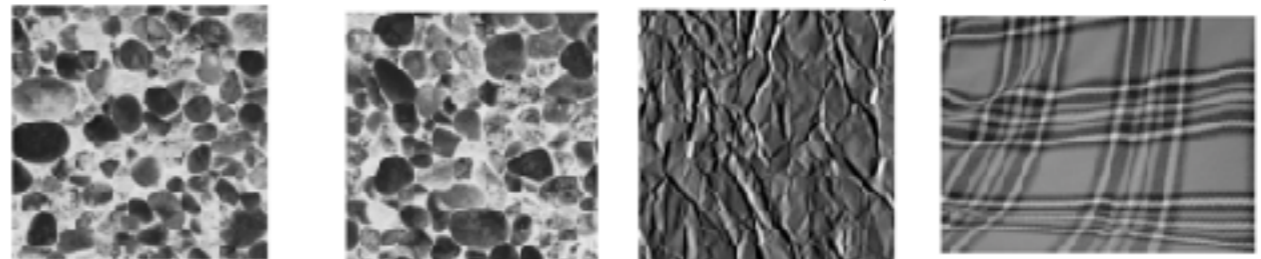


All variabilities
are known

Small deformations
+ Translation

- Textures

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



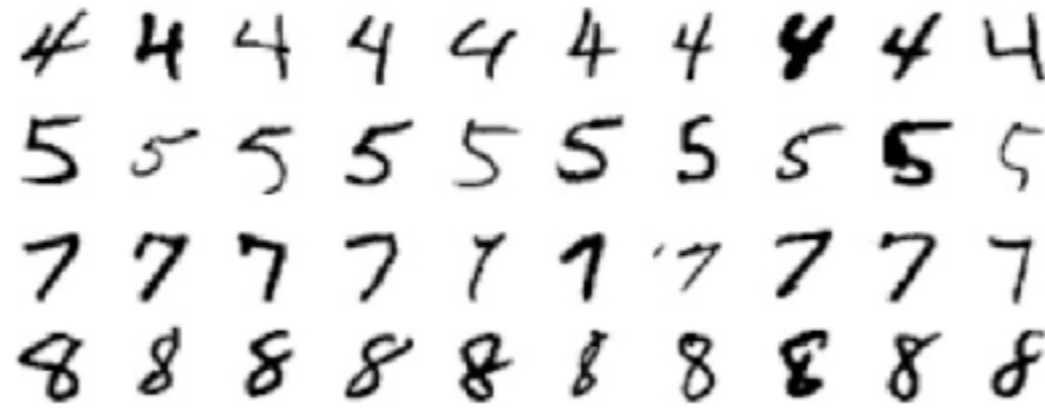
Rotation+Scale

A successful representation⁶⁸ in image classification

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

- Successfully used in several applications:

- Digits

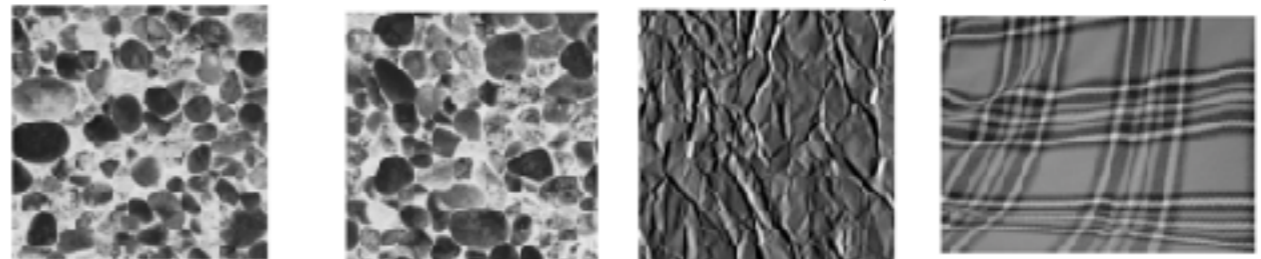


All variabilities
are known

Small deformations
+ Translation

- Textures

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



Rotation+Scale

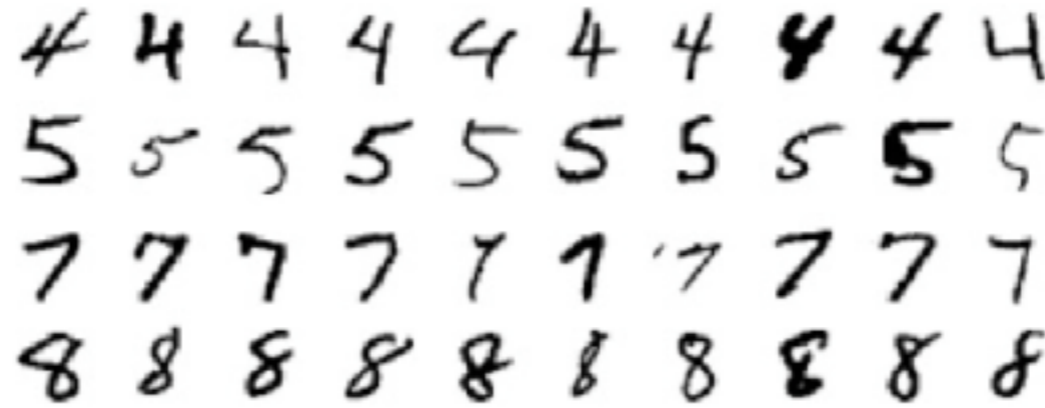
- The design of the scattering transform is guided by the euclidean group

A successful representation⁶⁸ in image classification

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

- Successfully used in several applications:

- Digits

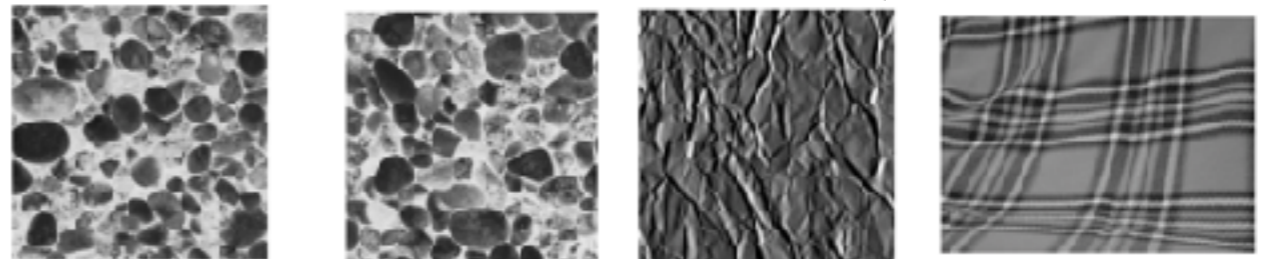


All variabilities
are known

Small deformations
+ Translation

- Textures

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



Rotation+Scale

- The design of the scattering transform is guided by the euclidean group
- To which extent can we compete with other architectures on more complex problems (e.g. variabilities are more complex)?

Extension to higher dimensional groups

Extension to higher dimensional groups

- The notion of convolution can be easily extended on a compact group or a Lie group G via a Haar measure.

Extension to higher dimensional groups

- 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:

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

Extension to higher dimensional groups

- 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:

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

- And thus the convolution operation:

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

Extension to higher dimensional groups

- 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:

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

- 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):

$$\begin{aligned} \rho : G &\rightarrow L^2(G) = \bigoplus_{\omega} E_{\omega} \\ g &\rightarrow \mathcal{L}_g \end{aligned}$$

An example: the roto-translation

Ref.: PhD of L. Sifre

- If the convolution is defined on G, G' , one can extend it to $G \times G', G \rtimes G'$.
- Roto-translation is a non commutative group:

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



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

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

An example: the roto-translation

Ref.: PhD of L. Sifre

- If the convolution is defined on G, G' , one can extend it to $G \times G', G \rtimes G'$.
- Roto-translation is a non commutative group:

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



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

$$\begin{aligned} (Y \circledast \Psi)(g) &= \int_{g'} Y(g') \Psi(g'^{-1}g) dg \\ &= \int_{\mathbb{R}^2} \int_{[0, 2\pi]} Y(u', \theta') \Psi(r_{-\theta'}(u - u'), \theta - \theta') du d\theta \end{aligned}$$

$$S_0 x = \int_u x(u) du \quad \text{and} \quad Y_{j_1}^1(u, \theta_1) = |x \star \psi_{j_1, \theta_1}(u)|$$

- Then Sx is invariant to roto-translation.

Ref.: PhD of L. Sifre

$$S_0 x = \int_u x(u) du \quad \text{and} \quad Y_{j_1}^1(u, \theta_1) = |x \star \psi_{j_1, \theta_1}(u)|$$

$$\text{Let } S_1 x = \int_{u, \theta} Y^1(u, \theta) du d\theta \quad \text{and} \quad \Psi(u, \theta) = \psi_{j_2, \theta_2}(u) \psi_k(\theta)$$

- Then Sx is invariant to roto-translation.

Ref.: PhD of L. Sifre

$$S_0 x = \int_u x(u) du \quad \text{and} \quad Y_{j_1}^1(u, \theta_1) = |x \star \psi_{j_1, \theta_1}(u)|$$

$$\text{Let } S_1 x = \int_{u, \theta} Y^1(u, \theta) du d\theta \quad \text{and} \quad \Psi(u, \theta) = \psi_{j_2, \theta_2}(u) \psi_k(\theta)$$

then, we get:

$$Y_{j_1, j_2, \theta_2, k}^2(\theta, u) = \int_{\theta', u'} |x \star \psi_{j_1, \theta'}(u')| \psi_{j_2, \theta_2 + \theta'}(u - u') \psi_k(\theta - \theta') du d\theta$$

- Then Sx is invariant to roto-translation.

$$S_0 x = \int_u x(u) du \quad \text{and} \quad Y_{j_1}^1(u, \theta_1) = |x \star \psi_{j_1, \theta_1}(u)|$$

$$\text{Let } S_1 x = \int_{u, \theta} Y^1(u, \theta) du d\theta \quad \text{and} \quad \Psi(u, \theta) = \psi_{j_2, \theta_2}(u) \psi_k(\theta)$$

then, we get:

$$Y_{j_1, j_2, \theta_2, k}^2(\theta, u) = \int_{\theta', u'} |x \star \psi_{j_1, \theta'}(u')| \psi_{j_2, \theta_2 + \theta'}(u - u') \psi_k(\theta - \theta') du d\theta$$

$$\text{Let } S_2 x = \int_{u, \theta} Y^2(u, \theta) du d\theta$$

- Then Sx is invariant to roto-translation.

$$S_0 x = \int_u x(u) du \quad \text{and} \quad Y_{j_1}^1(u, \theta_1) = |x \star \psi_{j_1, \theta_1}(u)|$$

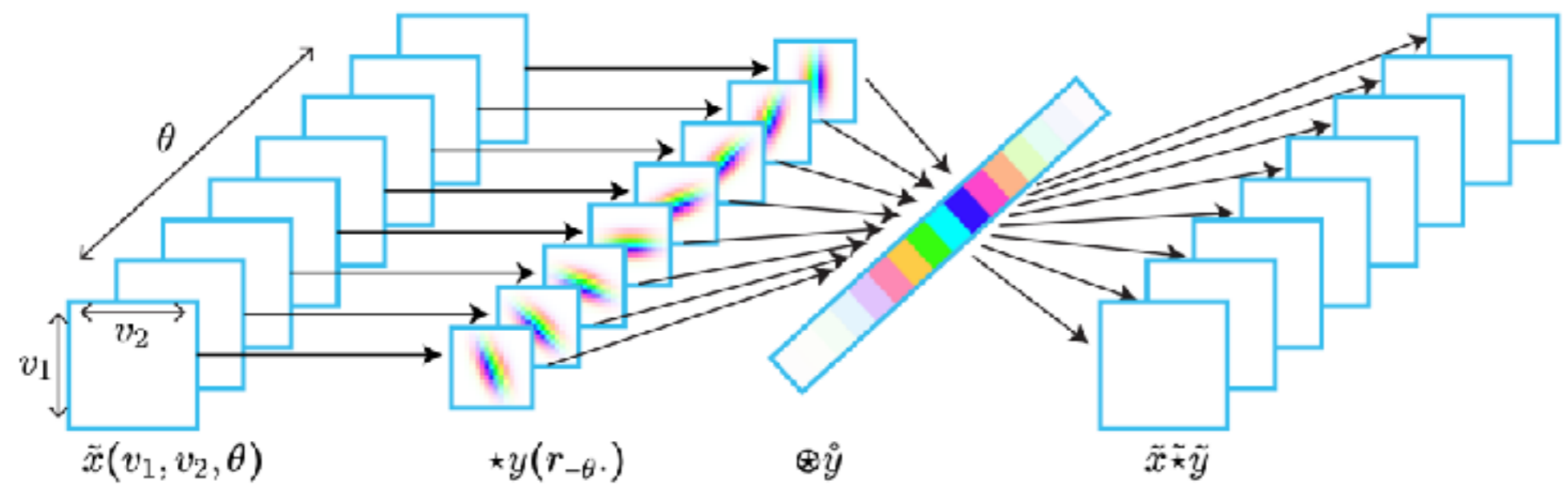
$$\text{Let } S_1 x = \int_{u, \theta} Y^1(u, \theta) du d\theta \quad \text{and} \quad \Psi(u, \theta) = \psi_{j_2, \theta_2}(u) \psi_k(\theta)$$

then, we get:

$$Y_{j_1, j_2, \theta_2, k}^2(\theta, u) = \int_{\theta', u'} |x \star \psi_{j_1, \theta'}(u')| \psi_{j_2, \theta_2 + \theta'}(u - u') \psi_k(\theta - \theta') du d\theta$$

$$\text{Let } S_2 x = \int_{u, \theta} Y^2(u, \theta) du d\theta$$

- Then Sx is invariant to roto-translation.



Symmetry group hypothesis

Symmetry group hypothesis

Ref.: Understanding deep
convolutional networks

S Mallat

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

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

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



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:

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

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

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:

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

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

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:

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

   One generalization among
many

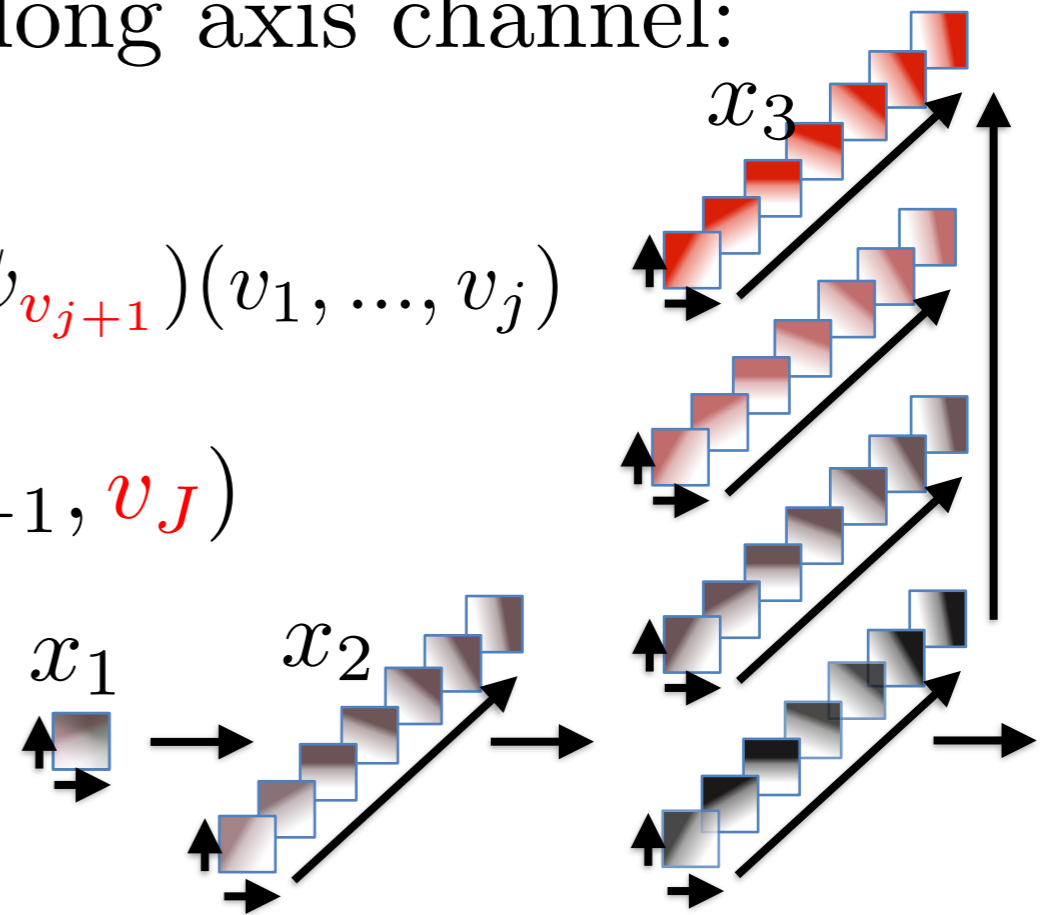
Ref.: Hierarchical CNNs, Jacobsen et al.

many

- CNN that is convolutional along axis channel:

$$x_{j+1}(v_1, \dots, v_j, v_{j+1}) = \rho_j(x_j \star^{v_1, \dots, v_j} \psi_{v_{j+1}})(v_1, \dots, v_j)$$

$$x_J(v_J) = \sum_{v_1, \dots, v_{J-1}} x_{J-1}(v_1, \dots, v_{J-1}, v_J)$$

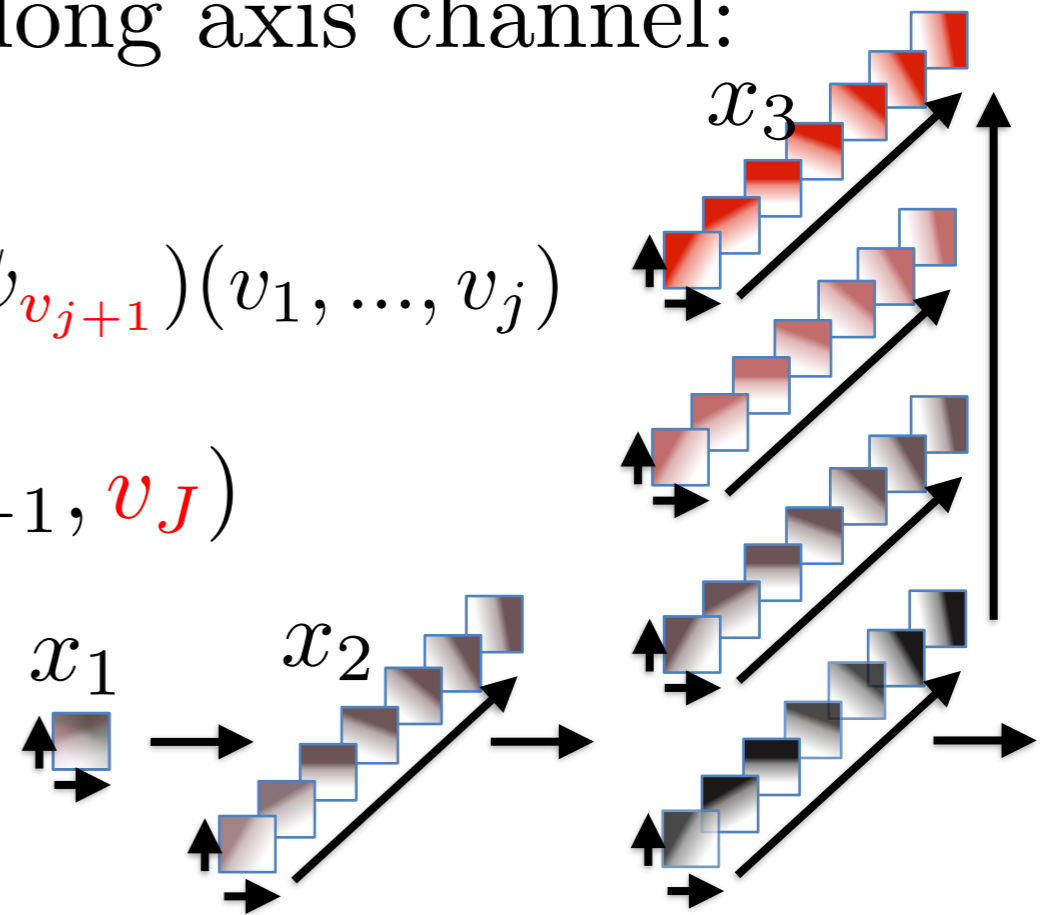


Ref.: Hierarchical CNNs, Jacobsen et al.

- CNN that is convolutional along axis channel:

$$x_{j+1}(v_1, \dots, v_j, v_{j+1}) = \rho_j(x_j \star^{v_1, \dots, v_j} \psi_{v_{j+1}})(v_1, \dots, v_j)$$

$$x_J(v_J) = \sum_{v_1, \dots, v_{J-1}} x_{J-1}(v_1, \dots, v_{J-1}, v_J)$$



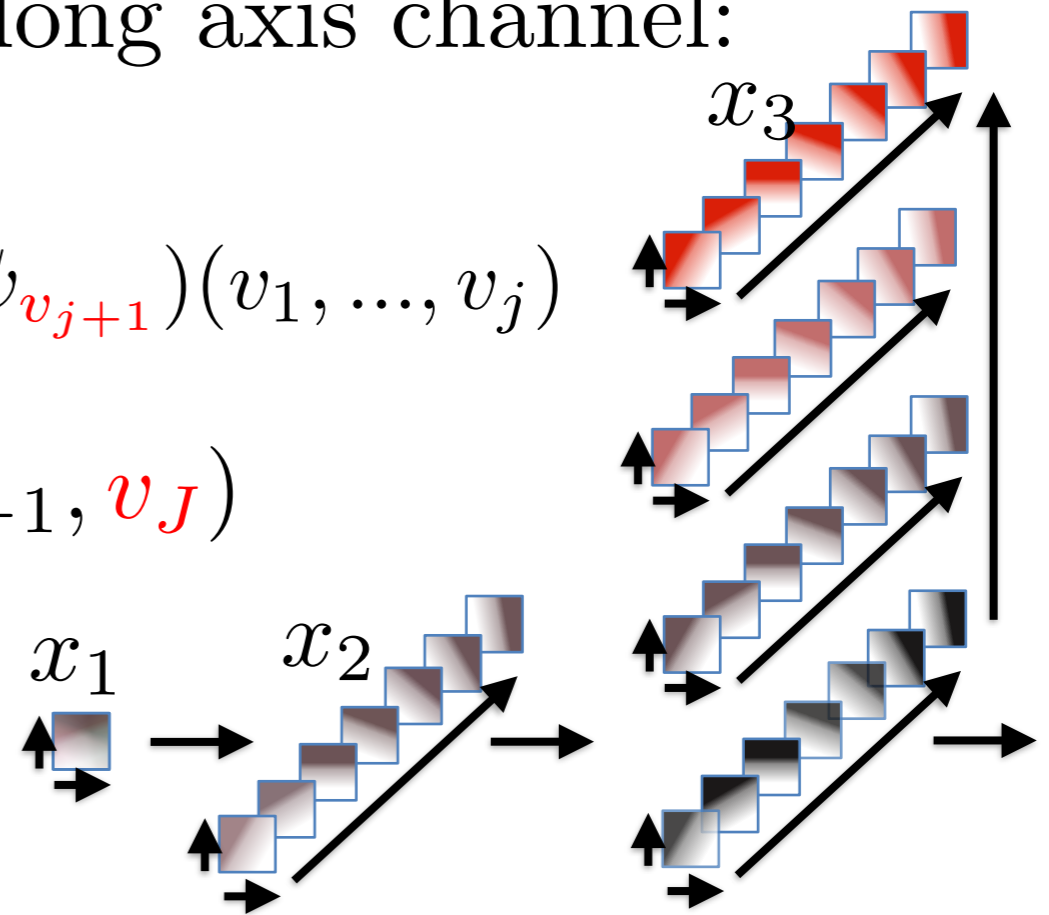
Ref.: Hierarchical CNNs, Jacobsen et al.

many

- CNN that is convolutional along axis channel:

$$x_{j+1}(v_1, \dots, v_j, v_{j+1}) = \rho_j(x_j \star^{v_1, \dots, v_j} \psi_{v_{j+1}})(v_1, \dots, v_j)$$

$$x_J(v_J) = \sum_{v_1, \dots, v_{J-1}} x_{J-1}(v_1, \dots, v_{J-1}, v_J)$$



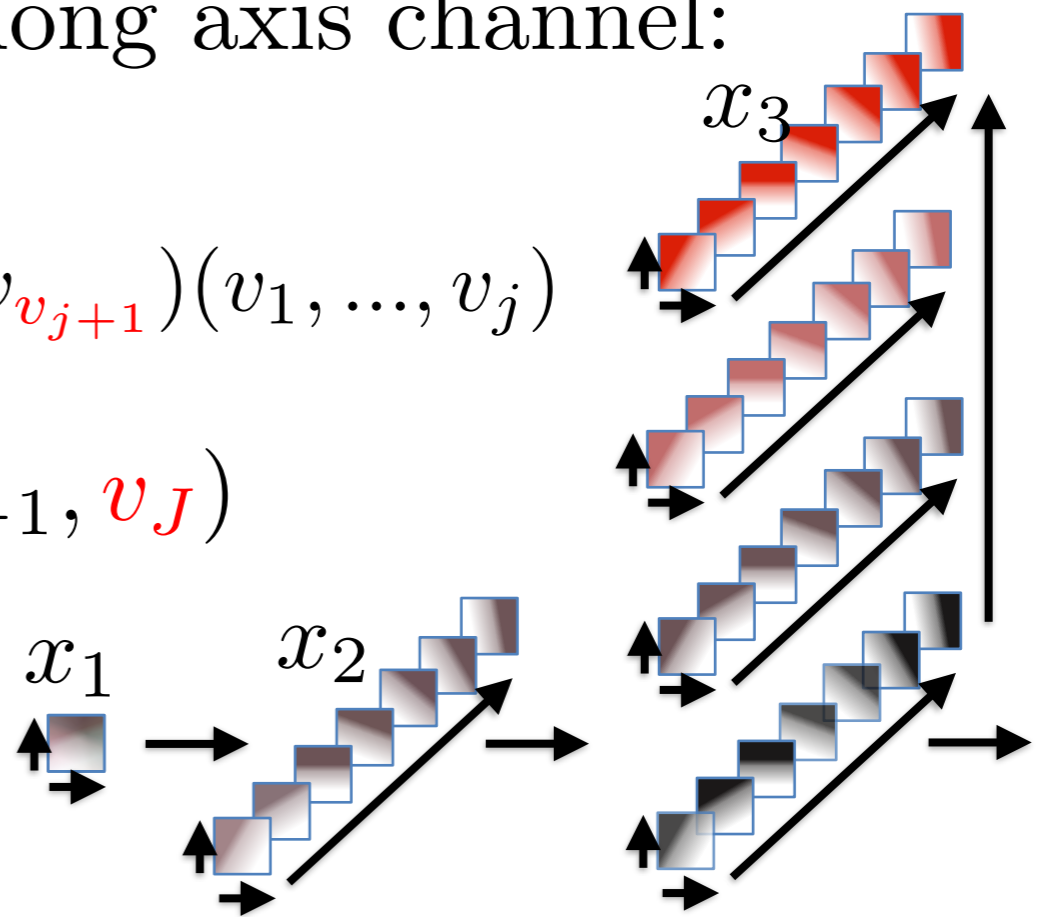
Ref.: Hierarchical CNNs, Jacobsen et al.

many

- CNN that is convolutional along axis channel:

$$x_{j+1}(v_1, \dots, v_j, v_{j+1}) = \rho_j(x_j \star^{v_1, \dots, v_j} \psi_{v_{j+1}})(v_1, \dots, v_j)$$

$$x_J(v_J) = \sum_{v_1, \dots, v_{J-1}} x_{J-1}(v_1, \dots, v_{J-1}, v_J)$$



Ref.: Hierarchical CNNs, Jacobsen et al.

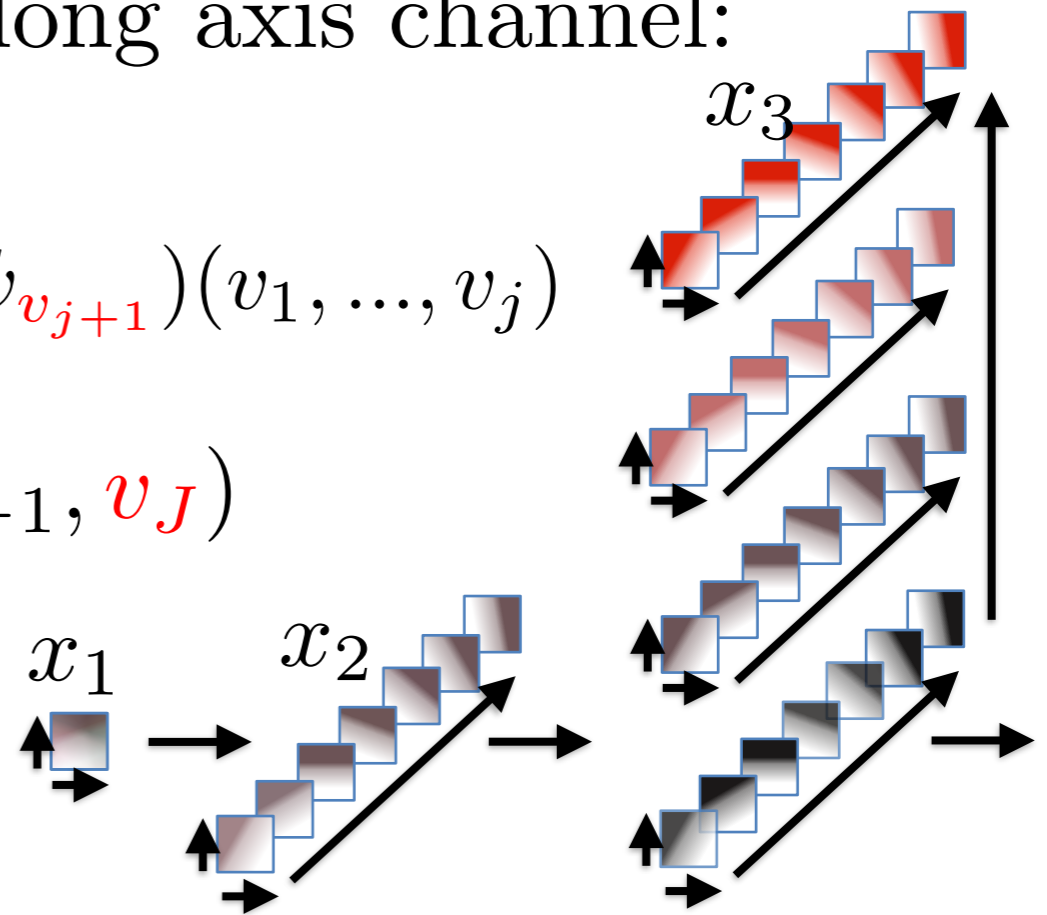
- For x_j , we refer to the variable v_j as an attribute that discriminates previously obtained layer.

many

- CNN that is convolutional along axis channel:

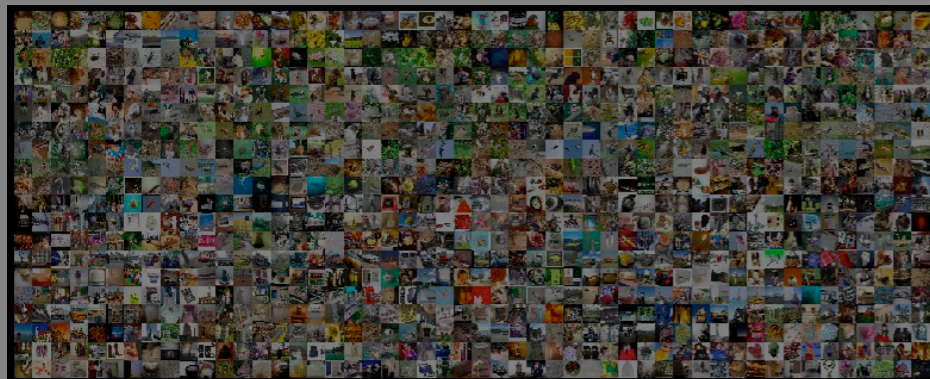
$$x_{j+1}(v_1, \dots, v_j, v_{j+1}) = \rho_j(x_j \star^{v_1, \dots, v_j} \psi_{v_{j+1}})(v_1, \dots, v_j)$$

$$x_J(v_J) = \sum_{v_1, \dots, v_{J-1}} x_{J-1}(v_1, \dots, v_{J-1}, v_J)$$

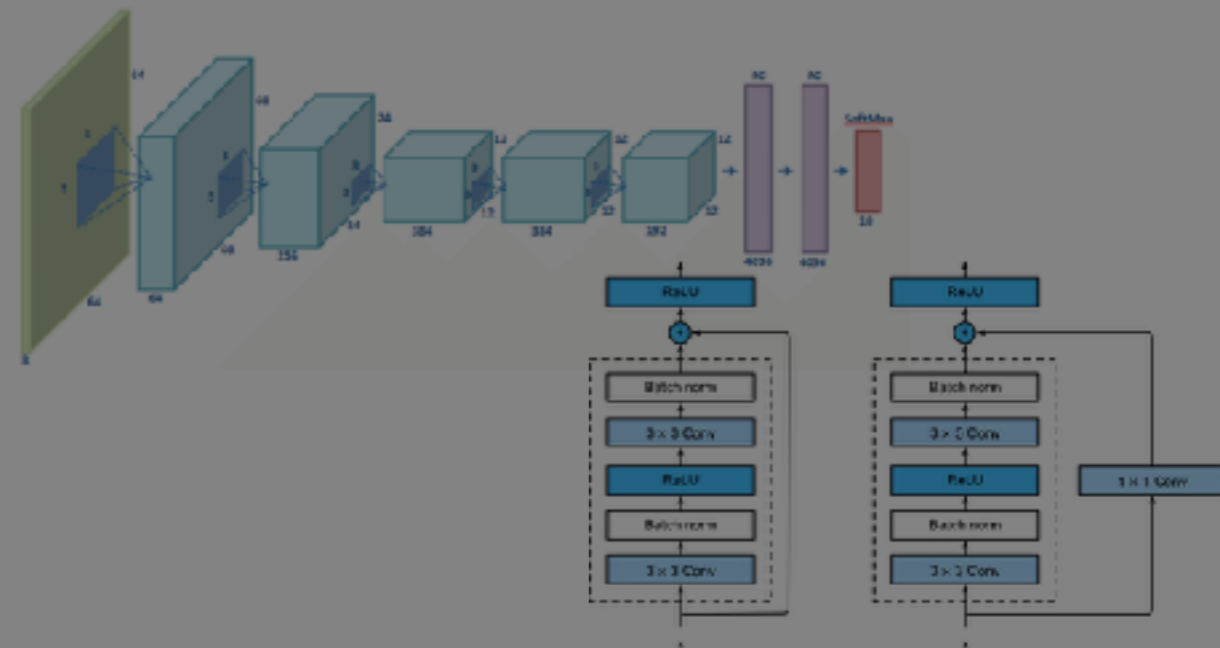
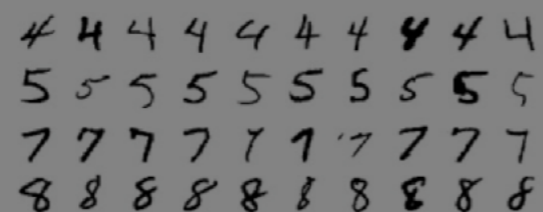


Ref.: Hierarchical CNNs, Jacobsen et al.

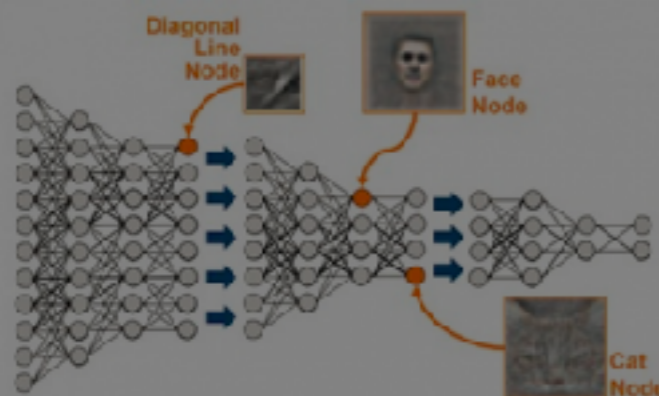
- For x_j , we refer to the variable v_j as an attribute that discriminates previously obtained layer.
- Representation is finally averaged: invariant along translations by v . Very similar to equivariant CNNs



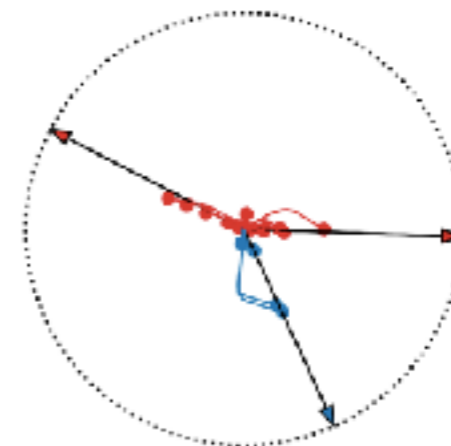
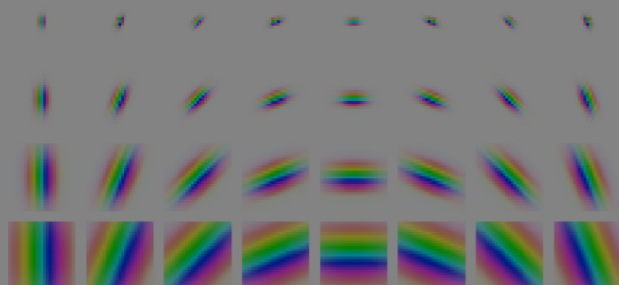
Introduction to image classification



Fighting the curse of dimensionality with Deep Neural Networks



Interpretability in Deep Learning



Statistical learning results

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

An opaque black-box

Risk decomposition

Risk decomposition

- For a fixed loss ℓ , consider the expected and empirical risk:

$$\mathcal{R}(\Phi) = \mathbb{E}[\ell(\Phi X, Y)] \quad \text{and} \quad \mathcal{R}_n(\Phi) = \frac{1}{n} \sum_{i \leq n} \ell(\Phi X_i, Y_i)$$

with: $\mathbb{E}[\mathcal{R}_n(\Phi)] = \mathcal{R}(\Phi)$

Risk decomposition

- For a fixed loss ℓ , consider the expected and empirical risk:

$$\mathcal{R}(\Phi) = \mathbb{E}[\ell(\Phi X, Y)] \quad \text{and} \quad \mathcal{R}_n(\Phi) = \frac{1}{n} \sum_{i \leq n} \ell(\Phi X_i, Y_i)$$

with: $\mathbb{E}[\mathcal{R}_n(\Phi)] = \mathcal{R}(\Phi)$

Risk decomposition

- For a fixed loss ℓ , consider the expected and empirical risk:

$$\mathcal{R}(\Phi) = \mathbb{E}[\ell(\Phi X, Y)] \quad \text{and} \quad \mathcal{R}_n(\Phi) = \frac{1}{n} \sum_{i \leq n} \ell(\Phi X_i, Y_i)$$

with: $\mathbb{E}[\mathcal{R}_n(\Phi)] = \mathcal{R}(\Phi)$

- We might be interested in those 3 quantities

Approximation Error




$$\mathcal{R}(\hat{\Phi}_n) - \inf_{\Phi \in \mathcal{F}} \mathcal{R}(\Phi) \leq \mathbb{E}[\underbrace{\inf_{\Phi \in \mathcal{F}} \mathcal{R}_n(\Phi) - \inf_{\Phi \in \mathcal{F}} \mathcal{R}(\Phi)}_{\text{Approximation Error}}] +$$

$$\underbrace{\mathbb{E}[2 \sup_{\Phi \in \mathcal{F}} |\mathcal{R}_n(\Phi) - \mathcal{R}(\Phi)|]}_{\text{Estimation Error}} +$$

$$\mathbb{E}[\underbrace{\mathcal{R}_n(\hat{\Phi}_n) - \inf_{\Phi \in \mathcal{F}} \mathcal{R}_n(\Phi)}_{\text{Optimization error}}]$$

Optimization error

\mathcal{F} : set of functions of interest

   You might be interested
in...

- Several implicit biases results (e.g., double gradient descent) .. it was already discussed on Monday
- Discussions around the optimization landscape .. sort of discussed yesterday
- Best approximation results of very deep neural networks.

- The complexity measure allows to compare richness of classes of functions. *Rademacher complexity* can be defined as:

$$\mathcal{Rad}_n(\mathcal{F}) = \mathbb{E}_{(\epsilon_i, X_i)_i} \left[\sup_{\Phi \in \mathcal{F}} \frac{1}{n} \sum_{i \leq n} \epsilon_i \Phi(X_i) \right]$$

- The complexity measure allows to compare richness of classes of functions. *Rademacher complexity* can be defined as:

$$\mathcal{Rad}_n(\mathcal{F}) = \mathbb{E}_{(\epsilon_i, X_i)_i} \left[\sup_{\Phi \in \mathcal{F}} \frac{1}{n} \sum_{i \leq n} \epsilon_i \Phi(X_i) \right]$$

- One can link Rademacher complexity to the generalization error, as (via symmetrisation+loss Lipschitz):

$$\mathcal{R}(\Phi) \leq \mathbb{E}[\mathcal{R}_n(\Phi)] + 2\mathcal{Rad}_n(\mathcal{F}) + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$$

- The complexity measure allows to compare richness of classes of functions. *Rademacher complexity* can be defined as:

$$\mathcal{Rad}_n(\mathcal{F}) = \mathbb{E}_{(\epsilon_i, X_i)_i} \left[\sup_{\Phi \in \mathcal{F}} \frac{1}{n} \sum_{i \leq n} \epsilon_i \Phi(X_i) \right]$$

- One can link Rademacher complexity to the generalization error, as (via symmetrisation+loss Lipschitz):

$$\mathcal{R}(\Phi) \leq \mathbb{E}[\mathcal{R}_n(\Phi)] + 2\mathcal{Rad}_n(\mathcal{F}) + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$$

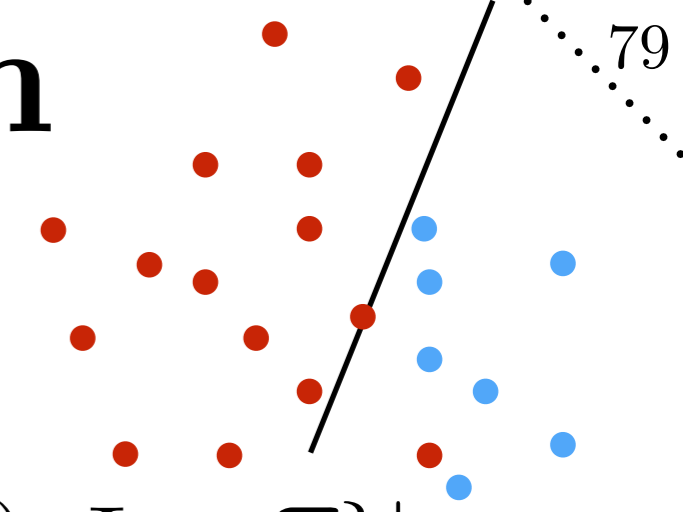
- The complexity measure allows to compare richness of classes of functions. *Rademacher complexity* can be defined as:

$$\mathcal{Rad}_n(\mathcal{F}) = \mathbb{E}_{(\epsilon_i, X_i)_i} \left[\sup_{\Phi \in \mathcal{F}} \frac{1}{n} \sum_{i \leq n} \epsilon_i \Phi(X_i) \right]$$

- One can link Rademacher complexity to the generalization error, as (via symmetrisation+loss Lipschitz):

$$\mathcal{R}(\Phi) \leq \mathbb{E}[\mathcal{R}_n(\Phi)] + 2\mathcal{Rad}_n(\mathcal{F}) + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$$

- In practice, it can be *difficult* to estimate.

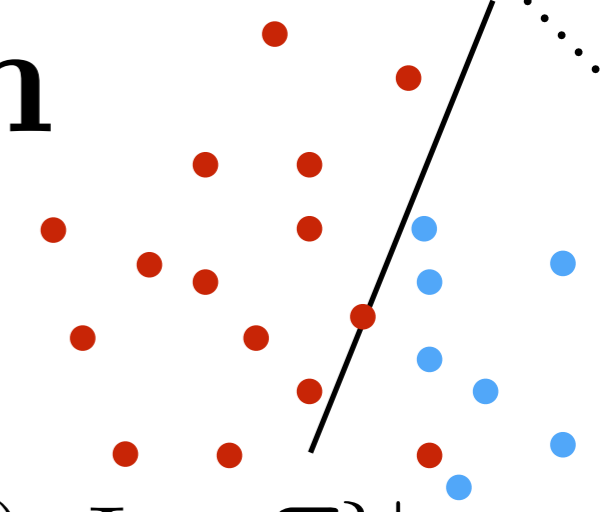


- For n points (x_1, \dots, x_m) let:

$$\Pi_{\mathcal{F}}(m) = \sup_{x_1, \dots, x_m \in \mathcal{X}} \#\{(\Phi(x_1), \dots, \Phi(x_m)), \Phi \in \mathcal{F}\}$$

If $\Pi_{\mathcal{F}}(m) = 2^m$ we say that \mathcal{F} shatters the set. For a dataset \mathcal{X} , the VC dimension is the largest m such that

$$\Pi_{\mathcal{F}}(m) = 2^m$$



- For n points (x_1, \dots, x_m) let:

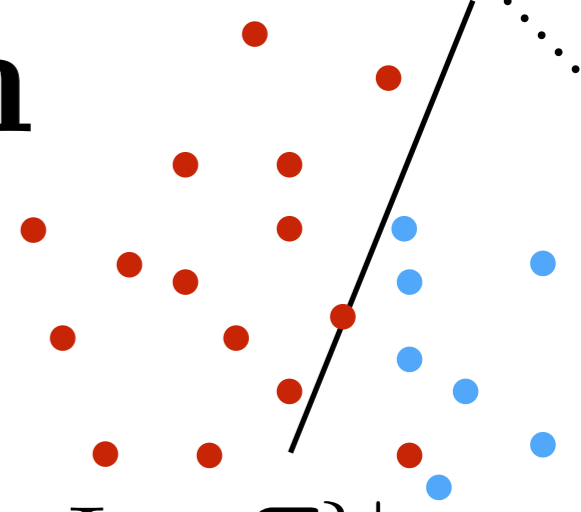
$$\Pi_{\mathcal{F}}(m) = \sup_{x_1, \dots, x_m \in \mathcal{X}} \#\{(\Phi(x_1), \dots, \Phi(x_m)), \Phi \in \mathcal{F}\}$$

If $\Pi_{\mathcal{F}}(m) = 2^m$ we say that \mathcal{F} shatters the set. For a dataset \mathcal{X} , the VC dimension is the largest m such that

$$\Pi_{\mathcal{F}}(m) = 2^m$$

- It can be linked to Rademacher complexity via:

$$\mathcal{R}_m(\mathcal{F}) \leq \sqrt{\frac{2 \log \Pi_{\mathcal{F}}(m)}{m}}$$



- For n points (x_1, \dots, x_m) let:

$$\Pi_{\mathcal{F}}(m) = \sup_{x_1, \dots, x_m \in \mathcal{X}} \#\{(\Phi(x_1), \dots, \Phi(x_m)), \Phi \in \mathcal{F}\}$$

If $\Pi_{\mathcal{F}}(m) = 2^m$ we say that \mathcal{F} shatters the set. For a dataset \mathcal{X} , the VC dimension is the largest m such that

$$\Pi_{\mathcal{F}}(m) = 2^m$$

- It can be linked to Rademacher complexity via:

$$\mathcal{R}_m(\mathcal{F}) \leq \sqrt{\frac{2 \log \Pi_{\mathcal{F}}(m)}{m}}$$

- For a neural network of depth L and with W parameters, Bartlett et al showed that:

$$VCdim(\mathcal{F}) = \mathcal{O}(WL \log(W) + WL^2)$$

Empirical Rademacher complexity:

$$\mathcal{Rad}_n(\mathcal{F})((X_i)_i) = \mathbb{E}_{(\epsilon_i)_i} \left[\sup_{\Phi \in \mathcal{F}} \frac{1}{n} \sum_{i \leq n} \epsilon_i \Phi(X_i) \right]$$

Empirical Rademacher complexity:

$$\mathcal{Rad}_n(\mathcal{F})((X_i)_i) = \mathbb{E}_{(\epsilon_i)_i} \left[\sup_{\Phi \in \mathcal{F}} \frac{1}{n} \sum_{i \leq n} \epsilon_i \Phi(X_i) \right]$$

- Typical decomposition of generalisation via concentration looks like: with high probability $1 - \delta$,

$$\text{Generalization error} \leq \mathcal{Rad}_n(\mathcal{F})((X_i)_i) + \mathcal{O}\left(\sqrt{\frac{\log \frac{1}{\delta}}{n}}\right)$$

Empirical Rademacher complexity:

$$\mathcal{Rad}_n(\mathcal{F})((X_i)_i) = \mathbb{E}_{(\epsilon_i)_i} \left[\sup_{\Phi \in \mathcal{F}} \frac{1}{n} \sum_{i \leq n} \epsilon_i \Phi(X_i) \right]$$

- Typical decomposition of generalisation via concentration looks like: with high probability $1 - \delta$,

$$\text{Generalization error} \leq \mathcal{Rad}_n(\mathcal{F})((X_i)_i) + \mathcal{O}\left(\sqrt{\frac{\log \frac{1}{\delta}}{n}}\right)$$

- In fact, it is empirically shown that CNN can fit random labels... Thus:

Ref.: Understanding Deep Learning requires rethinking generalization, C Zhang et al.

$$\mathcal{Rad}_n(\mathcal{F})((X_i)) \approx 1$$

- For a given trained neural network, it is possible to introduce a spectral complexity that can be no smaller than:

$$\mathcal{S}(\Phi) \geq \prod_{j=1}^J \|W_j\| \sqrt{\frac{J^3}{n}}$$

Ref.: Spectrally-normalized margin bounds for neural networks, Bartlett et al.

- For a given trained neural network, it is possible to introduce a spectral complexity that can be no smaller than:

$$\mathcal{S}(\Phi) \geq \prod_{j=1}^J \|W_j\| \sqrt{\frac{J^3}{n}}$$

Ref.: Spectrally-normalized margin bounds for neural networks, Bartlett et al.

- The main idea of the proof is to employ an ϵ -covering of a cascade of Lipschitz functions parametrised by $\{W_i\}_i$.

- For a given trained neural network, it is possible to introduce a spectral complexity that can be no smaller than:

$$\mathcal{S}(\Phi) \geq \prod_{j=1}^J \|W_j\| \sqrt{\frac{J^3}{n}}$$

Ref.: Spectrally-normalized margin bounds for neural networks, Bartlett et al.

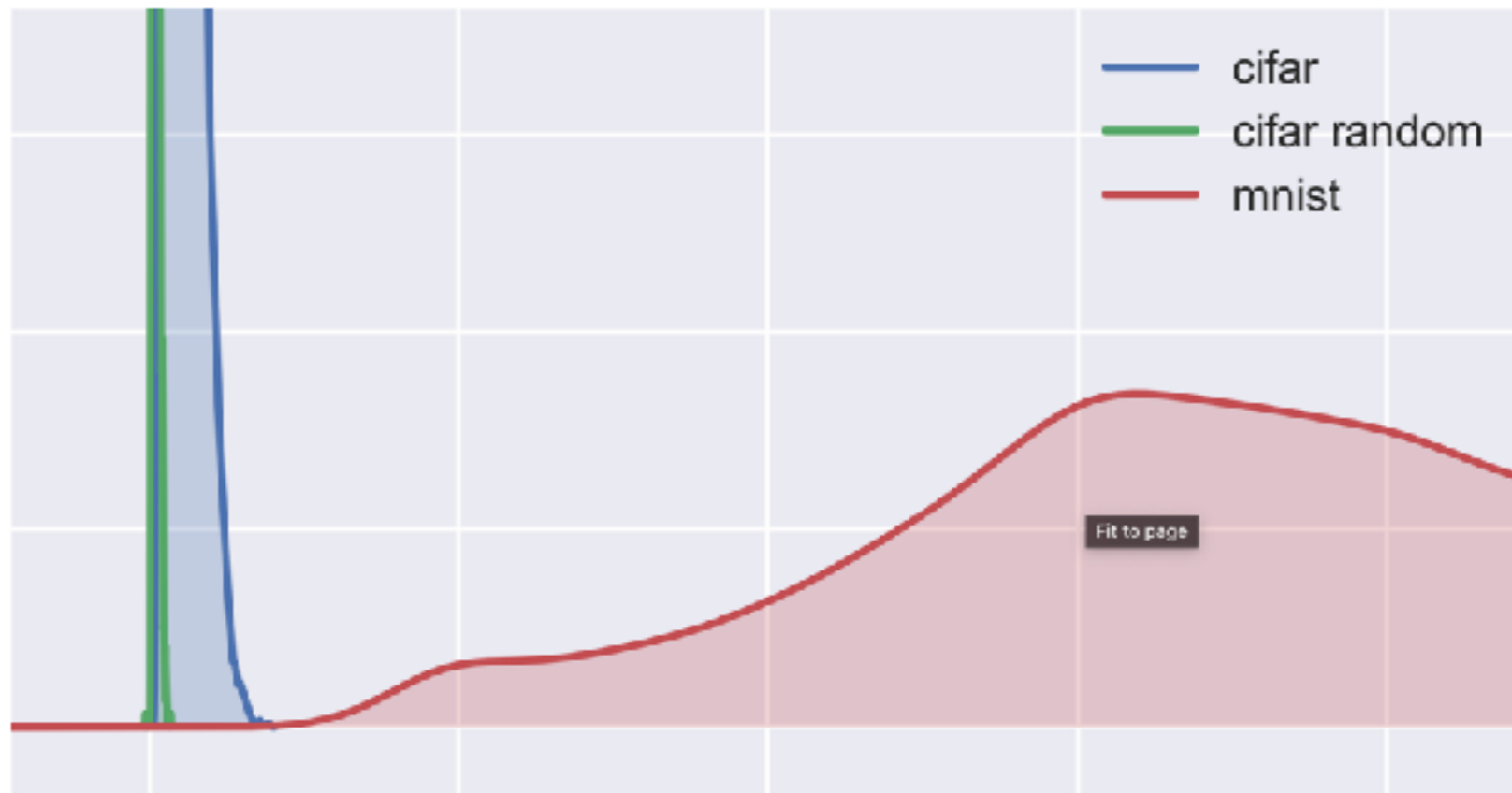
- The main idea of the proof is to employ an ϵ -covering of a cascade of Lipschitz functions parametrised by $\{W_i\}_i$.
- Those bounds also imply for the Rademacher complexity that, with high probability $\mathcal{S}(\Phi) \geq \mathcal{Rad}(\mathcal{F})$.
Yet they also be combined with margins.

Define a normalised margin:

Ref.: Spectrally-normalized margin bounds for neural networks, P Barlett et al.

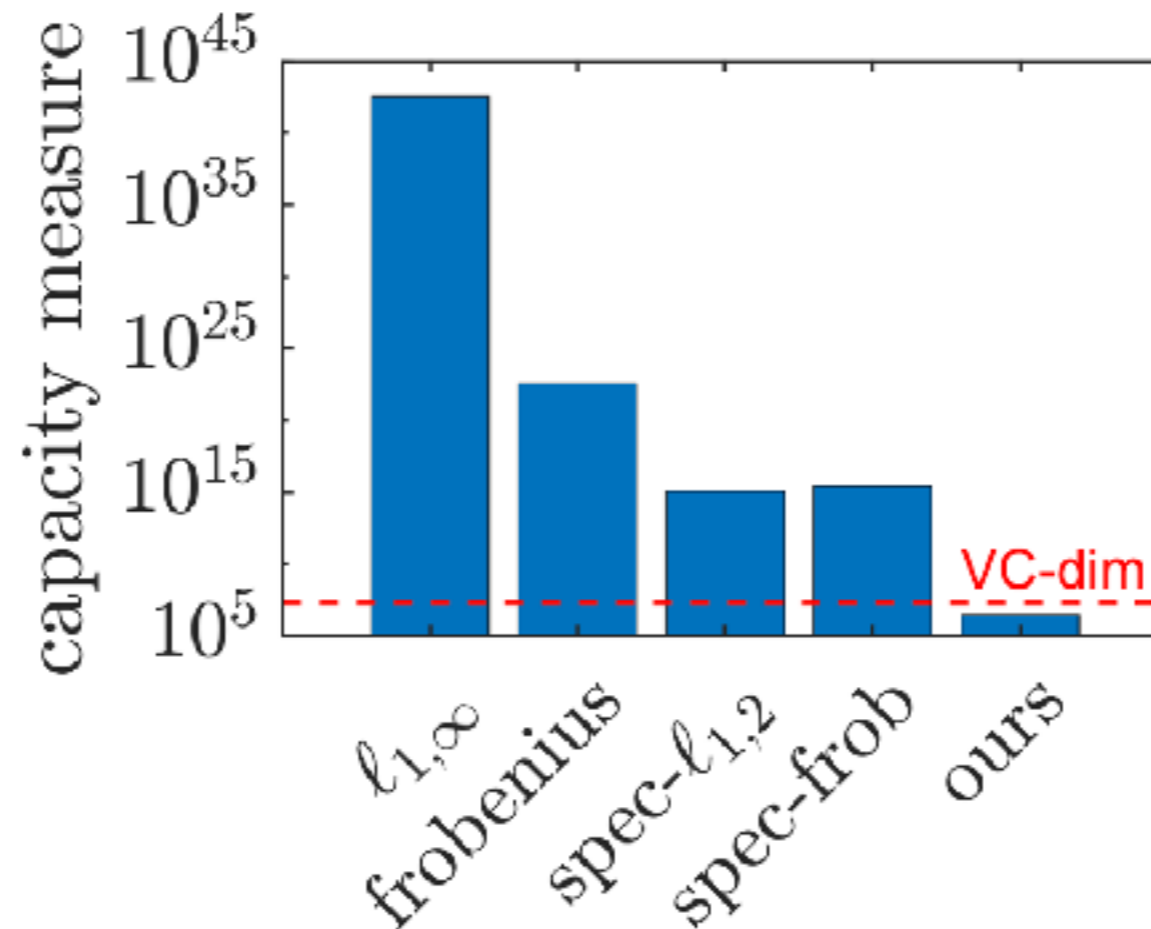
$$(x, y) \rightarrow \frac{(\Phi x)_y - \max_{i \neq y} (\Phi x)_i}{\mathcal{R}(\Phi) \|X\|_2}$$

Interestingly, the margin distribution is sensitive to this spectral complexity. It allows to quantify the hardness of datasets:



Bounds comparisons (with margins)

Generalization error



Ref.: Stronger generalization bounds for deep nets via a compression approach, Arora et al

Figure 4: **Left)** Comparing neural net generalization bounds.

$\ell_{1,\infty}$: $\frac{1}{\gamma^2} \prod_{i=1}^d \|A^i\|_{1,\infty}$ Bartlett and Mendelson [2002]

Frobenius: $\frac{1}{\gamma^2} \prod_{i=1}^d \|A^i\|_F^2$ Neyshabur et al. [2015b],

spec $\ell_{1,2}$: $\frac{1}{\gamma^2} \prod_{i=1}^d \|A_i\|_2^2 \sum_{i=1}^d \frac{\|A^i\|_{1,2}^2}{\|A^i\|_2^2}$ Bartlett et al. [2017]

spec-fro: $\frac{1}{\gamma^2} \prod_{i=1}^d \|A^i\|_2^2 \sum_{i=1}^d h_i \frac{\|A^i\|_F^2}{\|A^i\|_2^2}$ Neyshabur et al. [2017a]

ours: $\frac{1}{\gamma^2} \max_{x \in S} \|f(x)\|_2^2 \sum_{i=1}^d \frac{\beta^2 c_i^2 [\kappa/s]^2}{\mu_i^2 \mu_{i \rightarrow}^2}$

- In the general case, it is difficult to do better than the best "non-convex bounds" (i.e., vacuous)

- In the general case, it is difficult to do better than the best "non-convex bounds" (i.e., vacuous)
- **Example Gradient Descent** (SGD is straightforward to extend):

$$\theta_{t+1} = \theta_t - \alpha_t \nabla V(\theta_t)$$



Fix $V(\theta) = \ell(\Phi(\theta)) - \inf_{\omega} \ell(\Phi(\omega))$ then assuming V is L -smooth:

$$V(\theta_2) \leq V(\theta_1) + \nabla V(\theta_1)^T (\theta_2 - \theta_1) + \frac{L}{2} \|\theta_2 - \theta_1\|^2$$

one gets with assumptions on the step size:

$$\inf_{t \leq T} \|V(\theta_t)\| = o(1)$$

- Consider the following NN (without bias), with "NTK" renormalisation:

$$\Phi_J x = \frac{1}{\sqrt{w_J}} W_J \rho \frac{1}{\sqrt{w_{J-1}}} W_{J-1} \dots \rho \frac{1}{\sqrt{w_0}} W_0 x_0$$

Assume that each entry is initialised as:

- Consider the following NN (without bias), with "NTK" renormalisation:

$$\Phi_J x = \frac{1}{\sqrt{w_J}} W_J \rho \frac{1}{\sqrt{w_{J-1}}} W_{J-1} \dots \rho \frac{1}{\sqrt{w_0}} W_0 x_0$$

Assume that each entry is initialised as: $(W_j)_{mn} \sim \mathcal{N}(0, 1)$

- Then, in the infinite width limit, each element of $\Phi_0 x$ is an i.i.d. centered Gaussian Process with covariance $\Sigma_0(x, x') = \frac{1}{w_0} x^T x'$

- Consider the following NN (without bias), with "NTK" renormalisation:

$$\Phi_J x = \frac{1}{\sqrt{w_J}} W_J \rho \frac{1}{\sqrt{w_{J-1}}} W_{J-1} \dots \rho \frac{1}{\sqrt{w_0}} W_0 x_0$$

Assume that each entry is initialised as: $(W_j)_{mn} \sim \mathcal{N}(0, 1)$

- Then, in the infinite width limit, each element of $\Phi_0 x$ is an i.i.d. centered Gaussian Process with covariance $\Sigma_0(x, x') = \frac{1}{w_0} x^T x'$
- Similarly, $w_0 \rightarrow \infty, \dots, w_{j+1} \rightarrow \infty$ we get that $\Phi_{j+1} x$ is a GP:

$$\Sigma_{j+1}(x, x') = \mathbb{E}_{(u, v) \sim \mathcal{N}\left(0, \begin{bmatrix} \Sigma_j(x, x) & \Sigma_j(x', x) \\ \Sigma_j(x, x') & \Sigma_j(x', x') \end{bmatrix}\right)} [\rho(u)\rho(v)]$$

- Consider the following NN (without bias), with "NTK" renormalisation:

$$\Phi_J x = \frac{1}{\sqrt{w_J}} W_J \rho \frac{1}{\sqrt{w_{J-1}}} W_{J-1} \dots \rho \frac{1}{\sqrt{w_0}} W_0 x_0$$

Assume that each entry is initialised as: $(W_j)_{mn} \sim \mathcal{N}(0, 1)$

- Then, in the infinite width limit, each element of $\Phi_0 x$ is an i.i.d. centered Gaussian Process with covariance $\Sigma_0(x, x') = \frac{1}{w_0} x^T x'$

- Similarly, $w_0 \rightarrow \infty, \dots, w_{j+1} \rightarrow \infty$ we get that $\Phi_{j+1} x$ is a GP:

$$\Sigma_{j+1}(x, x') = \mathbb{E}_{(u,v) \sim \mathcal{N}(0, \begin{bmatrix} \Sigma_j(x, x) & \Sigma_j(x', x) \\ \Sigma_j(x, x') & \Sigma_j(x', x') \end{bmatrix})} [\rho(u)\rho(v)]$$

proof:

$$(\Phi_{j+1} x)_k (\Phi_{j+1} x')_k = \frac{1}{w_{j+1}} (\Phi_j x)^T \underbrace{(W_j)_k^T (W_j)_k}_{(W_j)_k \sim \mathcal{N}(0, I_{w_j})} \Phi_j x$$

- Assume Φ_J is real valued, define the NTK as:

$$\Theta_W(x, x') = \sum_{j=0}^J (\partial_{W_i} \Phi_J)^T \partial_{W_i} \Phi_J$$

- Assume Φ_J is real valued, define the NTK as:

$$\Theta_W(x, x') = \sum_{j=0}^J (\partial_{W_i} \Phi_J)^T \partial_{W_i} \Phi_J$$

- Then the dynamic of a NN for a given loss is given by

$$\begin{aligned} \frac{d}{dt} \Phi_J(x; W(t)) &= - \sum_{j=0}^J \mathbb{E}_n (\partial_{W_i} \Phi_J(x; W(t)))^T \partial_{W_i} \Phi_J(X; W(t)) \ell'(\Phi_J(X; W(t))) \\ &= - \mathbb{E}_n [\Theta_{W(t)}(x, X) \ell'(\Phi_J(X; W))] \end{aligned}$$

where $\mathbb{E}_n = \frac{1}{n} \sum_i \delta_{x_i}$

Let:

$$\dot{\Sigma}_{j+1}(x, x') = \mathbb{E}_{(u, v) \sim \mathcal{N}\left(0, \begin{bmatrix} \Sigma_j(x, x) & \Sigma_j(x', x) \\ \Sigma_j(x, x') & \Sigma_j(x', x') \end{bmatrix}\right)} [\dot{\rho}(u) \dot{\rho}(v)]$$

Let:

$$\dot{\Sigma}_{j+1}(x, x') = \mathbb{E}_{(u, v) \sim \mathcal{N}\left(0, \begin{bmatrix} \Sigma_j(x, x) & \Sigma_j(x', x) \\ \Sigma_j(x, x') & \Sigma_j(x', x') \end{bmatrix}\right)} [\dot{\rho}(u)\dot{\rho}(v)]$$

- Theorem (a): In the infinite width limit, we get:

$$\Theta_{W(0)}(x, x') = \sum_{j=0}^J \Sigma_j(x, x') \dot{\Sigma}_{j+1}(x, x') \dots \dot{\Sigma}_J(x, x')$$

Let:

$$\dot{\Sigma}_{j+1}(x, x') = \mathbb{E}_{(u, v) \sim \mathcal{N}\left(0, \begin{bmatrix} \Sigma_j(x, x) & \Sigma_j(x', x) \\ \Sigma_j(x, x') & \Sigma_j(x', x') \end{bmatrix}\right)} [\dot{\rho}(u)\dot{\rho}(v)]$$

- Theorem (a): In the infinite width limit, we get:

$$\Theta_{W(0)}(x, x') = \sum_{j=0}^J \Sigma_j(x, x') \dot{\Sigma}_{j+1}(x, x') \dots \dot{\Sigma}_J(x, x')$$

- Theorem (b): In the infinite width limit, we also get:

$$\Theta_{W(t)}(x, x') = \Theta_{W(0)}(x, x')$$

Let:

$$\dot{\Sigma}_{j+1}(x, x') = \mathbb{E}_{(u, v) \sim \mathcal{N}\left(0, \begin{bmatrix} \Sigma_j(x, x) & \Sigma_j(x', x) \\ \Sigma_j(x, x') & \Sigma_j(x', x') \end{bmatrix}\right)} [\dot{\rho}(u)\dot{\rho}(v)]$$

- Theorem (a): In the infinite width limit, we get:

$$\Theta_{W(0)}(x, x') = \sum_{j=0}^J \Sigma_j(x, x') \dot{\Sigma}_{j+1}(x, x') \dots \dot{\Sigma}_J(x, x')$$

- Theorem (b): In the infinite width limit, we also get:

$$\Theta_{W(t)}(x, x') = \Theta_{W(0)}(x, x')$$

- Consequence for a least square: $\frac{d}{dt}\Phi_t = A(\Phi_t - \Phi_*)$

Let:

$$\dot{\Sigma}_{j+1}(x, x') = \mathbb{E}_{(u,v) \sim \mathcal{N}(0, \begin{bmatrix} \Sigma_j(x, x) & \Sigma_j(x', x) \\ \Sigma_j(x, x') & \Sigma_j(x', x') \end{bmatrix})} [\dot{\rho}(u)\dot{\rho}(v)]$$

- Theorem (a): In the infinite width limit, we get:

$$\Theta_{W(0)}(x, x') = \sum_{j=0}^J \Sigma_j(x, x') \dot{\Sigma}_{j+1}(x, x') \dots \dot{\Sigma}_J(x, x')$$

- Theorem (b): In the infinite width limit, we also get:

$$\Theta_{W(t)}(x, x') = \Theta_{W(0)}(x, x')$$

- Consequence for a least square: $\frac{d}{dt}\Phi_t = A(\Phi_t - \Phi_*)$

- For a ReLU this kernel is equal to and is semi definite positive, thus $A > 0$.

- Consider any neural network Φ and let:

$$\bar{\Phi}(W) \triangleq \Phi(W(0)) + \nabla_W \Phi(W(0))^T (W - W(0))$$

- Consider any neural network Φ and let:

$$\bar{\Phi}(W) \triangleq \Phi(W(0)) + \nabla_W \Phi(W(0))^T (W - W(0))$$

- The dynamic of this parametrisation is given by:

$$\frac{d}{dt} \bar{\Phi}(\bar{W}(t)) = -\mathbb{E}_n [\Theta_{\bar{W}(0)} \ell'(\bar{\Phi}(\bar{W}(t)))]$$

- Consider any neural network Φ and let:

$$\bar{\Phi}(W) \triangleq \Phi(W(0)) + \nabla_W \Phi(W(0))^T (W - W(0))$$

- The dynamic of this parametrisation is given by:

$$\frac{d}{dt} \bar{\Phi}(\bar{W}(t)) = -\mathbb{E}_n [\Theta_{\bar{W}(0)} \ell'(\bar{\Phi}(\bar{W}(t)))]$$

- Consider any neural network Φ and let:

$$\bar{\Phi}(W) \triangleq \Phi(W(0)) + \nabla_W \Phi(W(0))^T (W - W(0))$$

- The dynamic of this parametrisation is given by:

$$\frac{d}{dt} \bar{\Phi}(\bar{W}(t)) = -\mathbb{E}_n[\Theta_{\bar{W}(0)} \ell'(\bar{\Phi}(\bar{W}(t)))]$$

- Then, without having the previous NTK parametrisation, we have:

$$\sup_{t \in [0, T]} \|\bar{W}(t) - W(t)\| = o\left(\frac{1}{\text{width}}\right)$$

Consider: $\bar{\Phi}(\Theta) \triangleq \Phi(\Theta_0) + D\Phi(\Theta_0) \cdot (\Theta - \Theta_0)$

Consider: $\bar{\Phi}(\Theta) \triangleq \Phi(\Theta_0) + D\Phi(\Theta_0) \cdot (\Theta - \Theta_0)$

as well as: $\bar{\mathcal{L}}(\Theta) \triangleq \hat{\mathcal{R}}(\bar{\Phi}(\Theta))$ and $\mathcal{L}(\Theta) \triangleq \hat{\mathcal{R}}(\Phi(\Theta))$.

Consider: $\bar{\Phi}(\Theta) \triangleq \Phi(\Theta_0) + D\Phi(\Theta_0) \cdot (\Theta - \Theta_0)$

as well as: $\bar{\mathcal{L}}(\Theta) \triangleq \hat{\mathcal{R}}(\bar{\Phi}(\Theta))$ and $\mathcal{L}(\Theta) \triangleq \hat{\mathcal{R}}(\Phi(\Theta))$.

- **Definition (informal):** A lazy regime occurs if the optimization paths of \mathcal{L} and $\bar{\mathcal{L}}$ remain close.

Consider: $\bar{\Phi}(\Theta) \triangleq \Phi(\Theta_0) + D\Phi(\Theta_0) \cdot (\Theta - \Theta_0)$

as well as: $\bar{\mathcal{L}}(\Theta) \triangleq \hat{\mathcal{R}}(\bar{\Phi}(\Theta))$ and $\mathcal{L}(\Theta) \triangleq \hat{\mathcal{R}}(\Phi(\Theta))$.

- **Definition (informal):** A lazy regime occurs if the optimization paths of \mathcal{L} and $\bar{\mathcal{L}}$ remain close.
- This is in particular true if:

$$\frac{\|\nabla \mathcal{L}(\Theta_0)\|}{\mathcal{L}(\Theta_0)} \gg \frac{\|D^2\Phi(\Theta_0)\|}{\|D\Phi(\Theta_0)\|} \quad \text{thus let: } \kappa(\Theta_0) \triangleq \frac{\mathcal{L}(\Theta_0)}{\|\nabla \mathcal{L}(\Theta_0)\|} \frac{\|D^2\Phi(\Theta_0)\|}{\|D\Phi(\Theta_0)\|}$$

Consider: $\bar{\Phi}(\Theta) \triangleq \Phi(\Theta_0) + D\Phi(\Theta_0) \cdot (\Theta - \Theta_0)$

as well as: $\bar{\mathcal{L}}(\Theta) \triangleq \hat{\mathcal{R}}(\bar{\Phi}(\Theta))$ and $\mathcal{L}(\Theta) \triangleq \hat{\mathcal{R}}(\Phi(\Theta))$.

- **Definition (informal):** A lazy regime occurs if the optimization paths of \mathcal{L} and $\bar{\mathcal{L}}$ remain close.
- This is in particular true if:

$$\frac{\|\nabla \mathcal{L}(\Theta_0)\|}{\mathcal{L}(\Theta_0)} \gg \frac{\|D^2\Phi(\Theta_0)\|}{\|D\Phi(\Theta_0)\|} \quad \text{thus let: } \kappa(\Theta_0) \triangleq \frac{\mathcal{L}(\Theta_0)}{\|\nabla \mathcal{L}(\Theta_0)\|} \frac{\|D^2\Phi(\Theta_0)\|}{\|D\Phi(\Theta_0)\|}$$

Consider: $\bar{\Phi}(\Theta) \triangleq \Phi(\Theta_0) + D\Phi(\Theta_0) \cdot (\Theta - \Theta_0)$

as well as: $\bar{\mathcal{L}}(\Theta) \triangleq \hat{\mathcal{R}}(\bar{\Phi}(\Theta))$ and $\mathcal{L}(\Theta) \triangleq \hat{\mathcal{R}}(\Phi(\Theta))$.

- **Definition (informal):** A lazy regime occurs if the optimization paths of \mathcal{L} and $\bar{\mathcal{L}}$ remain close.
- This is in particular true if:

$$\frac{\|\nabla \mathcal{L}(\Theta_0)\|}{\mathcal{L}(\Theta_0)} \gg \frac{\|D^2\Phi(\Theta_0)\|}{\|D\Phi(\Theta_0)\|} \quad \text{thus let: } \kappa(\Theta_0) \triangleq \frac{\mathcal{L}(\Theta_0)}{\|\nabla \mathcal{L}(\Theta_0)\|} \frac{\|D^2\Phi(\Theta_0)\|}{\|D\Phi(\Theta_0)\|}$$

- For a squared loss:

$$\kappa(\Theta_0) = \|\Phi(\Theta_0) - y^*\| \frac{\|D^2\Phi(\Theta_0)\|}{\|D\Phi(\Theta_0)\|}.$$

- Introduce the rescaled loss:

$$\mathcal{L}_\alpha(\Theta) \triangleq \frac{1}{\alpha^2} \hat{\mathcal{R}}(\alpha\Phi(\Theta)).$$

- Introduce the rescaled loss:

$$\mathcal{L}_\alpha(\Theta) \triangleq \frac{1}{\alpha^2} \hat{\mathcal{R}}(\alpha\Phi(\Theta)).$$


This rescaling is always implicitly present.

- Introduce the rescaled loss:

$$\mathcal{L}_\alpha(\Theta) \triangleq \frac{1}{\alpha^2} \hat{\mathcal{R}}(\alpha\Phi(\Theta)).$$

This rescaling is always implicitly present.

- Introduce the rescaled loss:


$$\mathcal{L}_\alpha(\Theta) \triangleq \frac{1}{\alpha^2} \hat{\mathcal{R}}(\alpha\Phi(\Theta)).$$


This rescaling is always implicitly present.

- For a MSE loss, the corresponding laziness is:

$$\kappa_\alpha(\Theta_0) = \frac{1}{\alpha} \|\alpha\Phi(\Theta_0) - y^*\| \frac{\|D^2\Phi(\Theta_0)\|}{\|D\Phi(\Theta_0)\|}.$$

- Introduce the rescaled loss:

$$\mathcal{L}_\alpha(\Theta) \triangleq \frac{1}{\alpha^2} \hat{\mathcal{R}}(\alpha\Phi(\Theta)).$$


This rescaling is always implicitly present.

- For a MSE loss, the corresponding laziness is:

$$\kappa_\alpha(\Theta_0) = \frac{1}{\alpha} \|\alpha\Phi(\Theta_0) - y^*\| \frac{\|D^2\Phi(\Theta_0)\|}{\|D\Phi(\Theta_0)\|}.$$

- Introduce the rescaled loss:

$$\mathcal{L}_\alpha(\Theta) \triangleq \frac{1}{\alpha^2} \hat{\mathcal{R}}(\alpha\Phi(\Theta)).$$

This rescaling is always implicitly present.

- For a MSE loss, the corresponding laziness is:

$$\kappa_\alpha(\Theta_0) = \frac{1}{\alpha} \|\alpha\Phi(\Theta_0) - y^*\| \frac{\|D^2\Phi(\Theta_0)\|}{\|D\Phi(\Theta_0)\|}.$$

- Other losses do not require specific theoretical adaptations (for finite horizons) to our measure of laziness and numerically we observed almost no differences: *no lack in generality.*



- The case of 1-hidden layer models (simpler to analyze):

$$\Phi(\Theta_0; x) = \alpha(m) \sum_{i=1}^m b_i \rho(w_i^T x), \quad \Theta_0 = \{b_i, w_i\}_{i \leq m}$$



- Generic case of deep CNNs (NTK-like):

- The case of 1-hidden layer models (simpler to analyze):

$$\Phi(\Theta_0; x) = \alpha(m) \sum_{i=1}^m b_i \rho(w_i^T x), \quad \Theta_0 = \{b_i, w_i\}_{i \leq m}$$

If $D\Phi(\Theta) \neq 0$ in a neighbourhood of Θ_0 and $\Theta_0 \sim \mathcal{N}(0, \sigma^2 I_{2m})$.

$$\mathbb{E}[\kappa_{\alpha(m)}(\Theta_0)] \lesssim m^{-\frac{1}{2}} + (m\alpha(m))^{-1}$$



- Generic case of deep CNNs (NTK-like):

- The case of 1-hidden layer models (simpler to analyze):

$$\Phi(\Theta_0; x) = \alpha(m) \sum_{i=1}^m b_i \rho(w_i^T x), \quad \Theta_0 = \{b_i, w_i\}_{i \leq m}$$

If $D\Phi(\Theta) \neq 0$ in a neighbourhood of Θ_0 and $\Theta_0 \sim \mathcal{N}(0, \sigma^2 I_{2m})$.

$$\mathbb{E}[\kappa_{\alpha(m)}(\Theta_0)] \lesssim m^{-\frac{1}{2}} + (m\alpha(m))^{-1}$$



This holds for several aforementioned references, with $\alpha(m) = \frac{1}{\sqrt{m}}$.

- Generic case of deep CNNs (NTK-like):

- The case of 1-hidden layer models (simpler to analyze):

$$\Phi(\Theta_0; x) = \alpha(m) \sum_{i=1}^m b_i \rho(w_i^T x), \quad \Theta_0 = \{b_i, w_i\}_{i \leq m}$$

If $D\Phi(\Theta) \neq 0$ in a neighbourhood of Θ_0 and $\Theta_0 \sim \mathcal{N}(0, \sigma^2 I_{2m})$.

$$\mathbb{E}[\kappa_{\alpha(m)}(\Theta_0)] \lesssim m^{-\frac{1}{2}} + (m\alpha(m))^{-1}$$



This holds for several aforementioned references, with $\alpha(m) = \frac{1}{\sqrt{m}}$.

Chizat and Bach have studied the setting $\alpha(m) = \frac{1}{m}$.

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

- Generic case of deep CNNs (NTK-like):

- The case of 1-hidden layer models (simpler to analyze):

$$\Phi(\Theta_0; x) = \alpha(m) \sum_{i=1}^m b_i \rho(w_i^T x), \quad \Theta_0 = \{b_i, w_i\}_{i \leq m}$$

If $D\Phi(\Theta) \neq 0$ in a neighbourhood of Θ_0 and $\Theta_0 \sim \mathcal{N}(0, \sigma^2 I_{2m})$.

$$\mathbb{E}[\kappa_{\alpha(m)}(\Theta_0)] \lesssim m^{-\frac{1}{2}} + (m\alpha(m))^{-1}$$



This holds for several aforementioned references, with $\alpha(m) = \frac{1}{\sqrt{m}}$.

Chizat and Bach have studied the setting $\alpha(m) = \frac{1}{m}$.

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

- Generic case of deep CNNs (NTK-like):

$$\kappa_{\alpha}(\Theta_0) = \frac{1}{\alpha} \|\alpha\Phi(\Theta_0) - y^*\| \frac{\|D^2\Phi\|}{\|D\Phi\|^2}$$

- The case of 1-hidden layer models (simpler to analyze):

$$\Phi(\Theta_0; x) = \alpha(m) \sum_{i=1}^m b_i \rho(w_i^T x), \quad \Theta_0 = \{b_i, w_i\}_{i \leq m}$$

If $D\Phi(\Theta) \neq 0$ in a neighbourhood of Θ_0 and $\Theta_0 \sim \mathcal{N}(0, \sigma^2 I_{2m})$.

$$\mathbb{E}[\kappa_{\alpha(m)}(\Theta_0)] \lesssim m^{-\frac{1}{2}} + (m\alpha(m))^{-1}$$



This holds for several aforementioned references, with $\alpha(m) = \frac{1}{\sqrt{m}}$.

Chizat and Bach have studied the setting $\alpha(m) = \frac{1}{m}$.

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

- Generic case of deep CNNs (NTK-like):

$$\kappa_{\alpha}(\Theta_0) = \frac{1}{\alpha} \|\alpha\Phi(\Theta_0) - y^*\| \frac{\|D^2\Phi\|}{\|D\Phi\|^2}$$

If $\Phi(\Theta_0) = 0$ and $\alpha \gg 1$ implying that $\kappa_{\alpha} \ll 1$

Lazy dynamic

standard dynamic: $\Theta'_\alpha(t) = -\nabla \mathcal{L}_\alpha(\Theta_\alpha(t))$

linearized dynamic: $\bar{\Theta}'_\alpha(t) = -\nabla \bar{\mathcal{L}}_\alpha(\bar{\Theta}_\alpha(t))$

standard dynamic: $\Theta'_\alpha(t) = -\nabla \mathcal{L}_\alpha(\Theta_\alpha(t))$

linearized dynamic: $\bar{\Theta}'_\alpha(t) = -\nabla \bar{\mathcal{L}}_\alpha(\bar{\Theta}_\alpha(t))$

- **Theorem (Chizat):** Assume that $\Phi(\Theta_0) = 0$. Given $T > 0$:

$$\sup_{t \in [0, T]} \|\Theta_\alpha(t) - \Theta_0\| = O\left(\frac{1}{\alpha}\right), \quad \sup_{t \in [0, T]} \|\Theta_\alpha(t) - \bar{\Theta}_\alpha(t)\| = O\left(\frac{1}{\alpha^2}\right)$$

and
$$\sup_{t \in [0, T]} \|\alpha \Phi(\Theta_\alpha(t)) - \alpha \bar{\Phi}(\bar{\Theta}_\alpha(t))\| = O\left(\frac{1}{\alpha}\right).$$

In other words, as alpha is large, the dynamic is close to the linearized dynamic

standard dynamic: $\Theta'_\alpha(t) = -\nabla \mathcal{L}_\alpha(\Theta_\alpha(t))$

linearized dynamic: $\bar{\Theta}'_\alpha(t) = -\nabla \bar{\mathcal{L}}_\alpha(\bar{\Theta}_\alpha(t))$

- **Theorem (Chizat):** Assume that $\Phi(\Theta_0) = 0$. Given $T > 0$:

$$\sup_{t \in [0, T]} \|\Theta_\alpha(t) - \Theta_0\| = O\left(\frac{1}{\alpha}\right), \quad \sup_{t \in [0, T]} \|\Theta_\alpha(t) - \bar{\Theta}_\alpha(t)\| = O\left(\frac{1}{\alpha^2}\right)$$

$$\text{and} \quad \sup_{t \in [0, T]} \|\alpha \Phi(\Theta_\alpha(t)) - \alpha \bar{\Phi}(\bar{\Theta}_\alpha(t))\| = O\left(\frac{1}{\alpha}\right).$$

In other words, as alpha is large, the dynamic is close to the linearized dynamic

- **Theorem (Chizat):** If $\hat{\mathcal{R}}$ is strongly convex, $\Phi(\Theta_0) = 0$, $\text{rk}(D\Phi(\Theta))$ is locally constant. Then, there exists α_0, C_1, C_2 :

$$\forall \alpha > \alpha_0, \exists \Theta_\infty^\alpha : \|\Phi(\Theta_\infty^\alpha) - \Phi(\Theta_t^\alpha)\| \leq C_1 \|\Phi(\Theta_\infty^\alpha)\| e^{-C_2 t}$$

$$\text{and } \nabla \mathcal{L}_\alpha(\Theta_\infty^\alpha) = 0$$

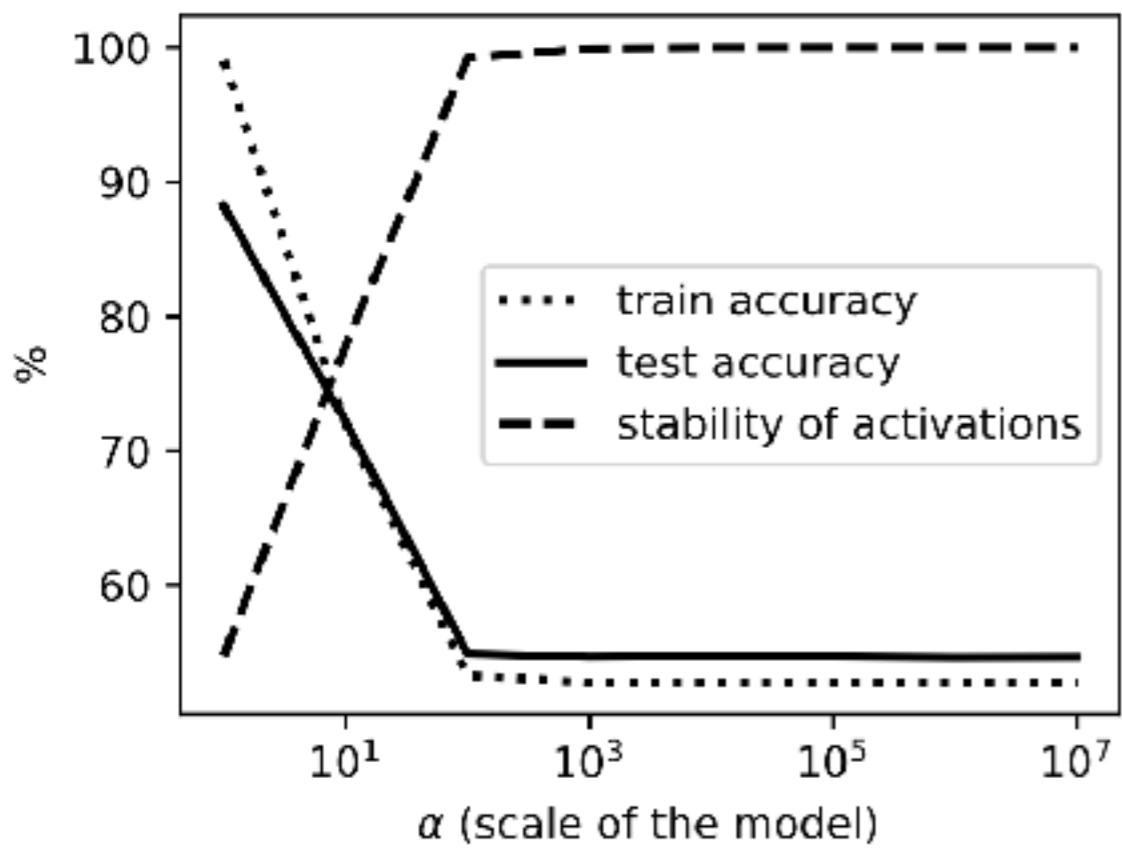
In other word, lazy regime allows to reach a local minimum.

Numerically, can lazy training be competitive?

CIFAR10 experiments using standard practice!!

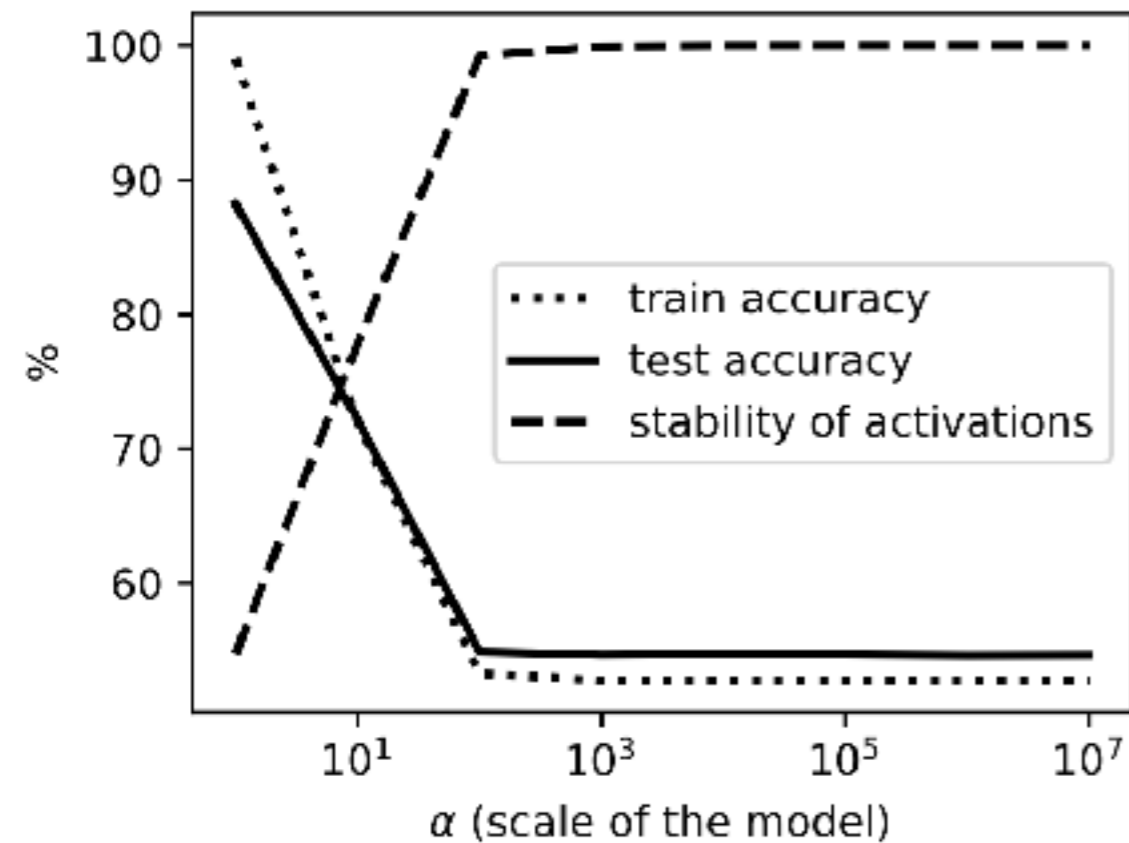
Numerically, can lazy training be competitive?

CIFAR10 experiments using standard practice!!



Numerically, can lazy training be competitive?

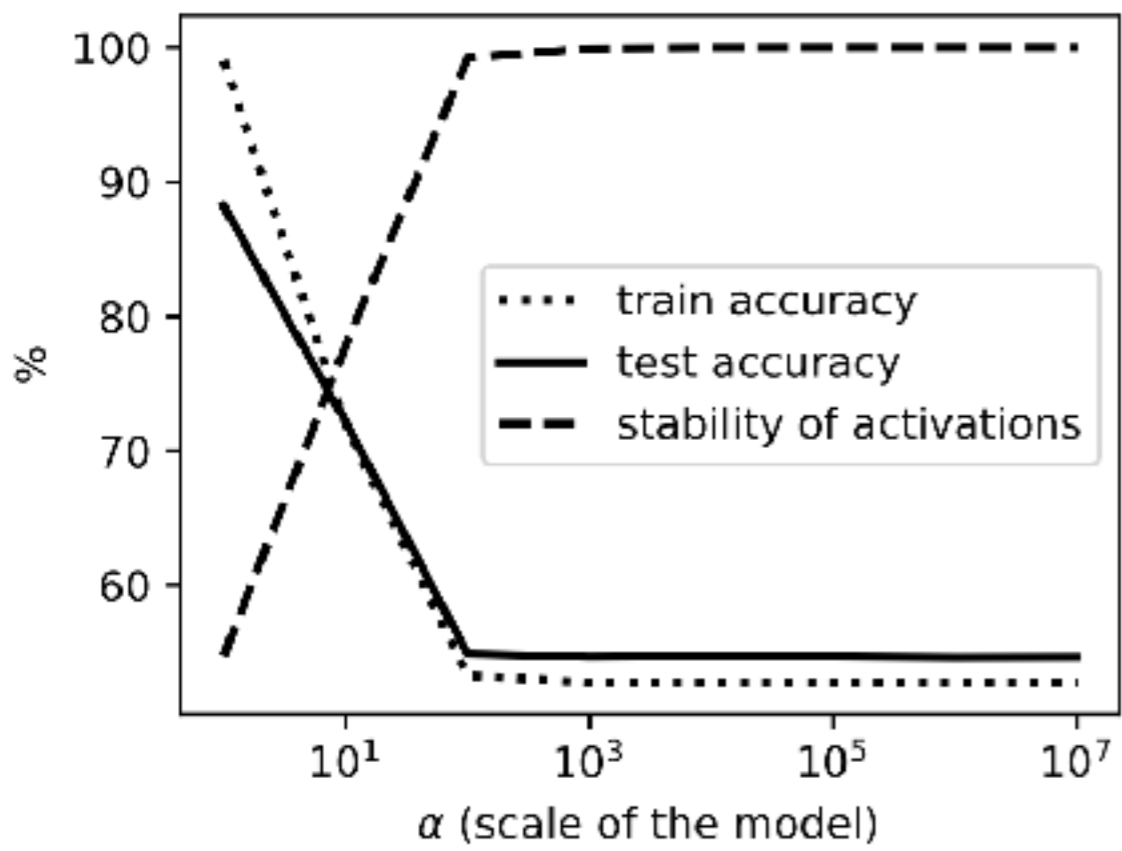
CIFAR10 experiments using standard practice!!



When $\alpha \rightarrow \infty$ there is a clear convergence.

Numerically, can lazy training be competitive?

CIFAR10 experiments using standard practice!!

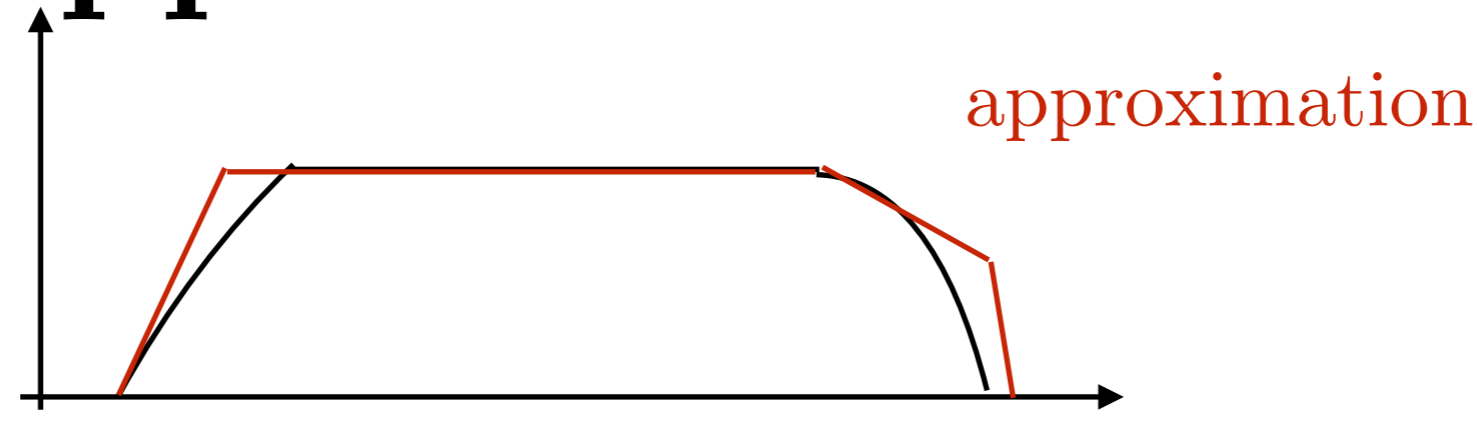


When $\alpha \rightarrow \infty$ there is a clear convergence.

For ReLU, if linearized:

$$\alpha \rho(w^T x) = \alpha \rho(w_0^T x) + \alpha \rho'(w_0^T x) x^T (w - w_0)$$

Thus, if linearized, the activations are stable.



- **Generic idea:** for a given function space, find a good approximation bound of a generator.

Ex.:
 Ref.: ResNet with one-neuron hidden layers is a Universal Approximator, Lin and Jegelka

- Infinite depth + a single neuron is an universal approximator in L^1 .
- Deep NNs lead to better approximations than linear methods for Besov, Sobolev, Hölder space...
- What about infinite width 1-hidden layer?

Ref.: Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: optimal rate and curse of dimensionality, T Suzuki

Theoretical results for 1-hidden layer Neural Networks

- Let f be a compactly smooth supported function:

$$f : \mathbb{R}^D \rightarrow \mathbb{R} \text{ and a smoothness measure: } C_f = \int_{\mathbb{R}^D} \|\omega\|_1 |\hat{f}(\omega)| d\omega$$

and ρ a non-linearity, bounded, strictly monotonically increasing and continuous (e.g. tanh)

- Let f be a compactly smooth supported function:

$$f : \mathbb{R}^D \rightarrow \mathbb{R} \text{ and a smoothness measure: } C_f = \int_{\mathbb{R}^D} \|\omega\|_1 |\hat{f}(\omega)| d\omega$$

and ρ a non-linearity, bounded, strictly monotonically increasing and continuous (e.g. tanh)

- Theorem: Universal approximation (Cybenko, 1991)

Let's note: $F^P : \{a_i, w_i\}_{i \leq P}$ and $F^P(x) = \sum_{i \leq P} a_i \rho(w_i^T x + b_i)$

Then: $\forall \epsilon, \exists F^P : \|F^P - f\|_\infty < \epsilon$

- Let f be a compactly smooth supported function:

$$f : \mathbb{R}^D \rightarrow \mathbb{R} \text{ and a smoothness measure: } C_f = \int_{\mathbb{R}^D} \|\omega\|_1 |\hat{f}(\omega)| d\omega$$

and ρ a non-linearity, bounded, strictly monotonically increasing and continuous (e.g. tanh)

- Theorem: Universal approximation (Cybenko, 1991)

Let's note: $F^P : \{a_i, w_i\}_{i \leq P}$ and $F^P(x) = \sum_{i \leq P} a_i \rho(w_i^T x + b_i)$

Then: $\forall \epsilon, \exists F^P : \|F^P - f\|_\infty < \epsilon$

- Let f be a compactly smooth supported function:

$$f : \mathbb{R}^D \rightarrow \mathbb{R} \text{ and a smoothness measure: } C_f = \int_{\mathbb{R}^D} \|\omega\|_1 |\hat{f}(\omega)| d\omega$$

and ρ a non-linearity, bounded, strictly monotonically increasing and continuous (e.g. tanh)

- Theorem: Universal approximation (Cybenko, 1991)

Let's note: $F^P : \{a_i, w_i\}_{i \leq P}$ and $F^P(x) = \sum_{i \leq P} a_i \rho(w_i^T x + b_i)$

Then: $\forall \epsilon, \exists F^P : \|F^P - f\|_\infty < \epsilon$

- Theorem: Approximation and estimation bounds (Barron, 1994)

If: $F^{N,P} = \arg \inf_{F^P} \sum_{j=1}^N \|F^P(X_j) - f(X_j)\|^2$

then: $E \|F^{N,P} - f\|^2 \leq \mathcal{O}\left(\frac{C_f^2}{N}\right) + \mathcal{O}\left(\frac{DN}{P} \log(P)\right)$

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

- We will explain (one of) the strategy from (Bach, 2014), only in the RKHS setting (more refined bound can be obtained outside the rkhs)

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

- We will explain (one of) the strategy from (Bach, 2014), only in the RKHS setting (more refined bound can be obtained outside the rkhs)

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

- Fix a measure τ, Ω compact and introduce:

$$\mathcal{F} = \left\{ f, f(x) = \int_{(v,b) \in \Omega} p(v) \rho(v^T x + b) d\tau(v, b), p \in L^2(\tau) \right\}$$

e.g., for a finite number of neurons: $f(x) = \sum_i p_i \rho(v_i^T x + b_i)$

- We will explain (one of) the strategy from (Bach, 2014), only in the RKHS setting (more refined bound can be obtained outside the rkhs)

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

- Fix a measure τ, Ω compact and introduce:

$$\mathcal{F} = \left\{ f, f(x) = \int_{(v,b) \in \Omega} p(v) \rho(v^T x + b) d\tau(v, b), p \in L^2(\tau) \right\}$$

e.g., for a finite number of neurons: $f(x) = \sum_i p_i \rho(v_i^T x + b_i)$

- Then, \mathcal{F} is a RKHS with kernel

$$k(x, y) = \int_{v \in \mathcal{V}} \rho(v^T x + b) \rho(v^T y + b) dv db$$

and norm

$$\|f\| = \inf_{f(x) = \int_{\Omega} p(v) \rho(v^T x + b) dv db} \|p\|, p \in L^2(\tau, \mathcal{V})$$

$$d = 1$$

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

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

$$f(\theta) = \int_{[0, 2\pi]} p(\theta') \rho(\cos(\theta - \theta')) d\theta' \quad g(\theta) = \sum_k c_k(g) e^{ik\theta}$$

$$c_k(f) = c_k(p) \gamma_k \quad \text{and for a ReLU} \quad \gamma_k = \begin{cases} 0, & \text{if } k = 2p + 1 \\ (-1)^p \frac{2}{k^2 - 1}, & \text{if } k = 2p \end{cases}$$

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

$$f(\theta) = \int_{[0, 2\pi]} p(\theta') \rho(\cos(\theta - \theta')) d\theta' \quad g(\theta) = \sum_k c_k(g) e^{ik\theta}$$

$$c_k(f) = c_k(p) \gamma_k \text{ and for a ReLU } \gamma_k = \begin{cases} 0, & \text{if } k = 2p + 1 \\ (-1)^p \frac{2}{k^2 - 1}, & \text{if } k = 2p \end{cases}$$

- Let's focus on functions defined over the 2D sphere:

$$\|p\|^2 = \sum_k |c_k(p)|^2 = \sum_{k, \gamma_k \neq 0} \frac{|c_k(f)|^2}{\gamma_k^2} + \sum_{k, \gamma_k = 0} |c_k(p)|^2$$

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

$$f(\theta) = \int_{[0,2\pi]} p(\theta') \rho(\cos(\theta - \theta')) d\theta' \quad g(\theta) = \sum_k c_k(g) e^{ik\theta}$$

$$c_k(f) = c_k(p) \gamma_k \text{ and for a ReLU } \gamma_k = \begin{cases} 0, & \text{if } k = 2p + 1 \\ (-1)^p \frac{2}{k^2 - 1}, & \text{if } k = 2p \end{cases}$$

- Let's focus on functions defined over the 2D sphere:

$$\|p\|^2 = \sum_k |c_k(p)|^2 = \sum_{k, \gamma_k \neq 0} \frac{|c_k(f)|^2}{\gamma_k^2} + \sum_{k, \gamma_k = 0} |c_k(p)|^2$$

- Lipschitz functions are in this RKHS.
(proof via a Poisson kernel)

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

$$f(\theta) = \int_{[0,2\pi]} p(\theta') \rho(\cos(\theta - \theta')) d\theta' \quad g(\theta) = \sum_k c_k(g) e^{ik\theta}$$

$$c_k(f) = c_k(p) \gamma_k \text{ and for a ReLU } \gamma_k = \begin{cases} 0, & \text{if } k = 2p + 1 \\ (-1)^p \frac{2}{k^2 - 1}, & \text{if } k = 2p \end{cases}$$

- Let's focus on functions defined over the 2D sphere:

$$\|p\|^2 = \sum_k |c_k(p)|^2 = \sum_{k, \gamma_k \neq 0} \frac{|c_k(f)|^2}{\gamma_k^2} + \sum_{k, \gamma_k = 0} |c_k(p)|^2$$

- Lipschitz functions are in this RKHS.
(proof via a Poisson kernel)

- Approximation results are easier with this type of results

Optimization of 1-hidden layer NN

$$F(\mu) = R\left(\int \varphi d\mu\right) + \text{regularization}(\mu)$$

pde given by: $\partial_t \mu_t = -\text{div}(v_t \mu_t)$, $v_t \in -\partial F'(\mu_t)$

(a) if: $\mu_n(t) = \frac{1}{n} \sum_{i=1}^n \delta_{x_i(t)}$ then $\mu_n(t) \rightarrow \mu(t)$

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

(b) there exists under "nice" conditions μ^* s.t.:

$$\mathcal{W}_p(\mu_t, \mu^*) \rightarrow 0$$

layer NN

- In the mean field limit, one can get convergence guarantees on the flow of an infinite width NN.

$$F(\mu) = R\left(\int \varphi d\mu\right) + \text{regularization}(\mu)$$

pde given by: $\partial_t \mu_t = -\text{div}(v_t \mu_t), v_t \in -\partial F'(\mu_t)$

(a) if: $\mu_n(t) = \frac{1}{n} \sum_{i=1}^n \delta_{x_i(t)}$ then $\mu_n(t) \rightarrow \mu(t)$

Ref.: On the global convergence of gradient descent for over-parameterized models using

optimal transport
Chizat and Bach

(b) there exists under "nice" conditions μ^* s.t.:

$$\mathcal{W}_p(\mu_t, \mu^*) \rightarrow 0$$

- In the mean field limit, one can get convergence guarantees on the flow of an infinite width NN.

$$F(\mu) = R\left(\int \varphi d\mu\right) + \text{regularization}(\mu)$$

pde given by: $\partial_t \mu_t = -\text{div}(v_t \mu_t), v_t \in -\partial F'(\mu_t)$

(a) if: $\mu_n(t) = \frac{1}{n} \sum_{i=1}^n \delta_{x_i(t)}$ then $\mu_n(t) \rightarrow \mu(t)$

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

(b) there exists under "nice" conditions μ^* s.t.:

$$\mathcal{W}_p(\mu_t, \mu^*) \rightarrow 0$$

- In the mean field limit, one can get convergence guarantees on the flow of an infinite width NN.

$$F(\mu) = R\left(\int \varphi d\mu\right) + \text{regularization}(\mu)$$

pde given by: $\partial_t \mu_t = -\text{div}(v_t \mu_t), v_t \in -\partial F'(\mu_t)$

(a) if: $\mu_n(t) = \frac{1}{n} \sum_{i=1}^n \delta_{x_i(t)}$ then $\mu_n(t) \rightarrow \mu(t)$

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

(b) there exists under "nice" conditions μ^* s.t.:

$$\mathcal{W}_p(\mu_t, \mu^*) \rightarrow 0$$

layer NN

- In the mean field limit, one can get convergence guarantees on the flow of an infinite width NN.

$$F(\mu) = R\left(\int \varphi d\mu\right) + \text{regularization}(\mu)$$

pde given by: $\partial_t \mu_t = -\text{div}(v_t \mu_t), v_t \in -\partial F'(\mu_t)$

(a) if: $\mu_n(t) = \frac{1}{n} \sum_{i=1}^n \delta_{x_i(t)}$ then $\mu_n(t) \rightarrow \mu(t)$

Ref.: On the global convergence of gradient descent for over-parameterized models using

optimal transport
Chizat and Bach

(b) there exists under "nice" conditions μ^* s.t.:

$$\mathcal{W}_p(\mu_t, \mu^*) \rightarrow 0$$

- Those guarantees are purely asymptotic and seem difficult to extend to deeper NNs.

Ref.: The Power of Depth for Feedforward
Neural Networks, R Eldan and O Shamir

- 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 $g : \mathbb{R}^d \rightarrow \mathbb{R}$:

- 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 $g : \mathbb{R}^d \rightarrow \mathbb{R}$:
- 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.

- 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 $g : \mathbb{R}^d \rightarrow \mathbb{R}$:
- 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.

- Please try the first tutorial (classifying CIFAR10) on your own.
- <https://edouardoyallon.github.io/cirm2021/>

- Deep neural networks are difficult tools to analyse...

- Deep neural networks are difficult tools to analyse...
- ... that can lead to super exciting new results.