Foundations of Deep Learning from a Kernel Point of View

Julien Mairal

Inria Grenoble

Berlin, CoSIP winter school, 2017

1. Several Paradigms in Machine Learning
   - Deep Neural Networks
   - Kernel Methods
   - Sparse Estimation

2. Convolutional Kernel Networks
   - Challenges of Deep Kernel Machines
   - Construction of CKNs
   - Applications to Image Data
   - CKNs for Biological Sequences

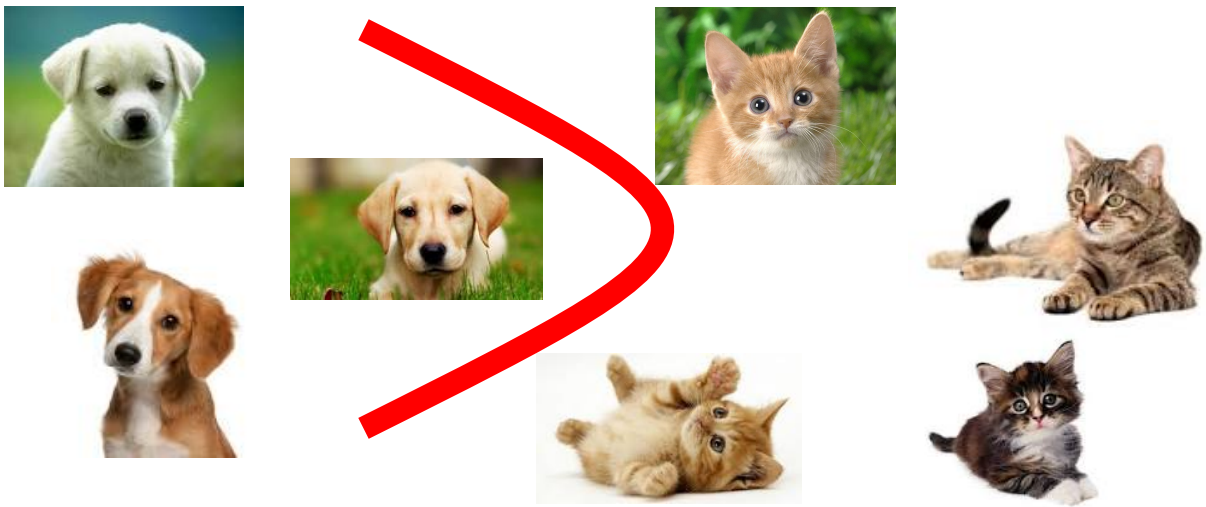
3. Invariance, Stability, and Complexity of Deep Convolutional Representations
   - Invariance and Stability to Deformations
   - Signal Recovery
   - Complexity and Learning Capacity

Part I: Several Paradigms in Machine Learning

Common paradigm: optimization for machine learning

Optimization is central to machine learning. For instance, in supervised learning, the goal is to learn a **prediction function** $f : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$



[Vapnik, 1995, Bottou, Curtis, and Nocedal, 2016]...

Common paradigm: optimization for machine learning

Optimization is central to machine learning. For instance, in supervised learning, the goal is to learn a **prediction function** $f : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\ldots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$

The scalars y_i are in

- $\{-1, +1\}$ for **binary** classification problems.
- $\{1, \ldots, K\}$ for **multi-class** classification problems.
- $\mathbb{R}$ for **regression** problems.
- $\mathbb{R}^k$ for **multivariate regression** problems.

Common paradigm: optimization for machine learning

Optimization is central to machine learning. For instance, in supervised learning, the goal is to learn a **prediction function** $f : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\ldots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$

Example with linear models: logistic regression, SVMs, *etc.*

- assume there exists a linear relation between y and features x in $\mathbb{R}^p$.
- $f(x) = w^\top x + b$ is parametrized by w, b in $\mathbb{R}^{p+1}$;
- L is often a **convex** loss function;
- $\Omega(f)$ is often the squared ℓ_2-norm $\|w\|^2$.

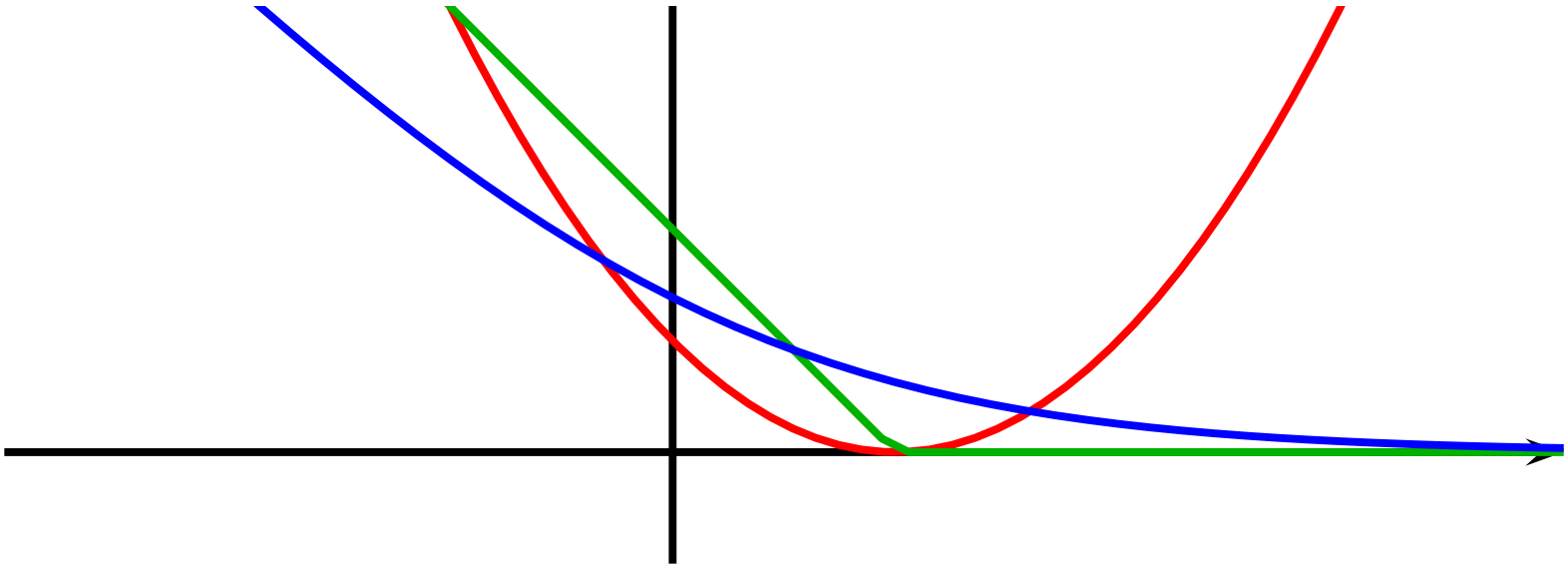
Common paradigm: optimization for machine learning

A few examples of linear models with no bias b:

Ridge regression:
$$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2}(y_i - w^\top x_i)^2 + \lambda \|w\|_2^2.$$

Linear SVM:
$$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i w^\top x_i) + \lambda \|w\|_2^2.$$

Logistic regression:
$$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i w^\top x_i}\right) + \lambda \|w\|_2^2.$$



Common paradigm: optimization for machine learning

The previous formulation is called *empirical risk minimization*; it follows a classical scientific paradigm:

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

Common paradigm: optimization for machine learning

The previous formulation is called *empirical risk minimization*; it follows a classical scientific paradigm:

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

A general principle

It underlies many paradigms:

- **deep neural networks,**
- **kernel methods,**
- sparse estimation.

Common paradigm: optimization for machine learning

The previous formulation is called *empirical risk minimization*; it follows a classical scientific paradigm:

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

Even with simple linear models, it leads to challenging problems in optimization: develop algorithms that

- **scale** both in the problem size n and dimension p;
- are able to **exploit the problem structure** (sum, composite);
- come with **convergence and numerical stability** guarantees;
- come with **statistical guarantees**.

Common paradigm: optimization for machine learning

The previous formulation is called *empirical risk minimization*; it follows a classical scientific paradigm:

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

It is not limited to supervised learning

$$\min_{f \in \mathcal{F}} \quad \frac{1}{n} \sum_{i=1}^{n} L(f(x_i)) \; + \; \lambda \Omega(f).$$

- L is not a classification loss any more;
- K-means, PCA, EM with mixture of Gaussian, matrix factorization,... can be expressed that way.

Paradigm 1: Deep neural networks

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}} .$$

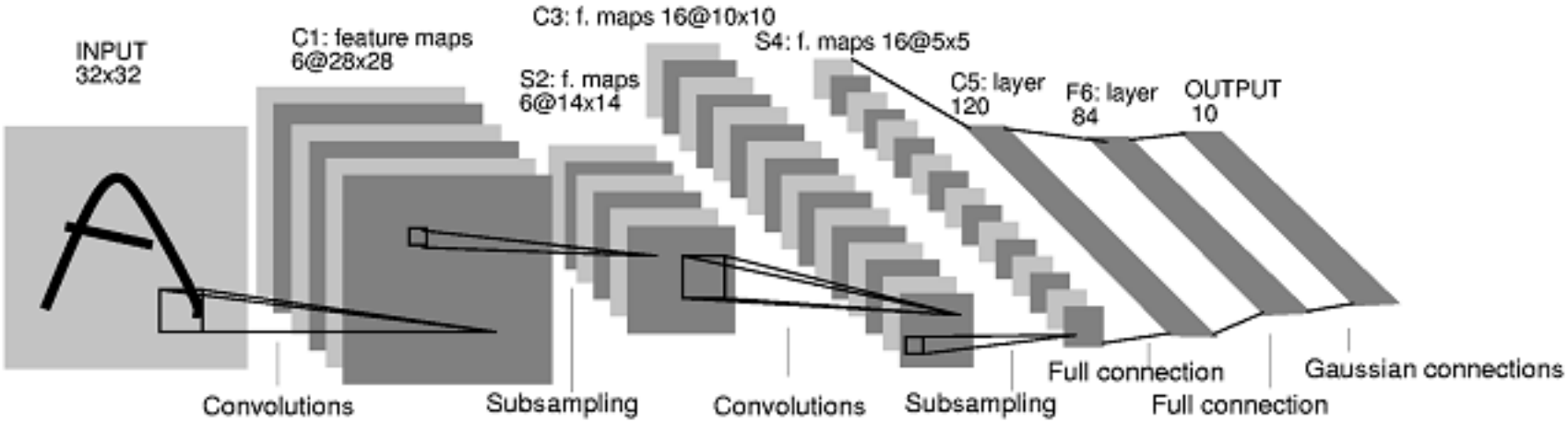
- The "deep learning" space $\mathcal{F}$ is parametrized:

$$f(x) = \sigma_k(A_k \sigma_{k-1}(A_{k-1} \ldots \sigma_2(A_2 \sigma_1(A_1 x)) \ldots)).$$

- Finding the optimal $A_1, A_2, \ldots, A_k$ yields an (intractable) **non-convex** optimization problem in **huge dimension.**
- Linear operations are either unconstrained (fully connected) or involve parameter sharing (e.g., convolutions).

Paradigm 1: Deep neural networks

A quick zoom on convolutional neural networks



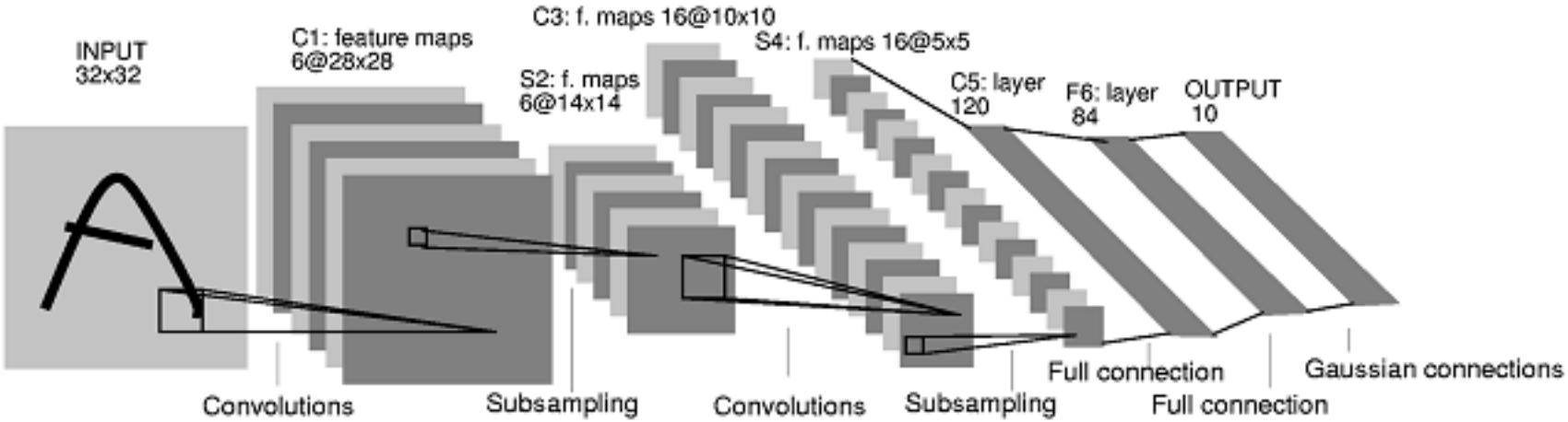
What are the main features of CNNs?

- they capture **compositional** and **multiscale** structures in images;
- they provide some **invariance**;
- they model **local stationarity** of images at several scales.
- **state-of-the-art** in many fields.

[LeCun et al., 1989, 1998, Ciresan et al., 2012, Krizhevsky et al., 2012]...

Paradigm 1: Deep neural networks

A quick zoom on convolutional neural networks



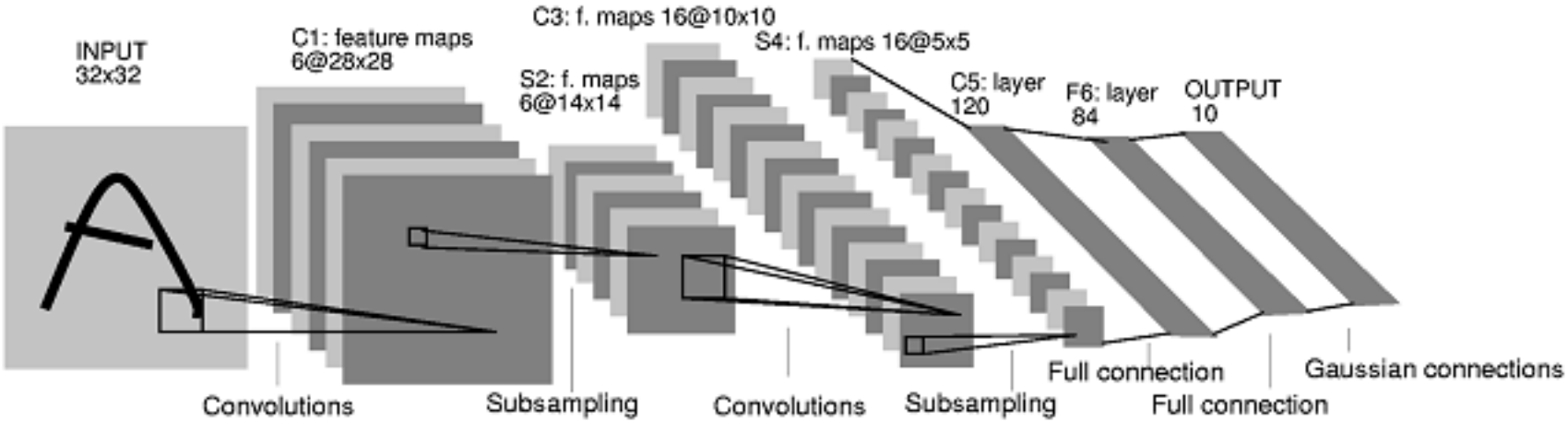
What are the main open problems?

- very little **theoretical understanding**;
- they require **large amounts of labeled data**;
- they require **manual design and parameter tuning**;
- how to **regularize** is unclear;

[LeCun et al., 1989, 1998, Ciresan et al., 2012, Krizhevsky et al., 2012]...

Paradigm 1: Deep neural networks

A quick zoom on convolutional neural networks



How to use them?

- they are the focus of a **huge academic and industrial effort**;
- there is **efficient and well-documented open-source software**;

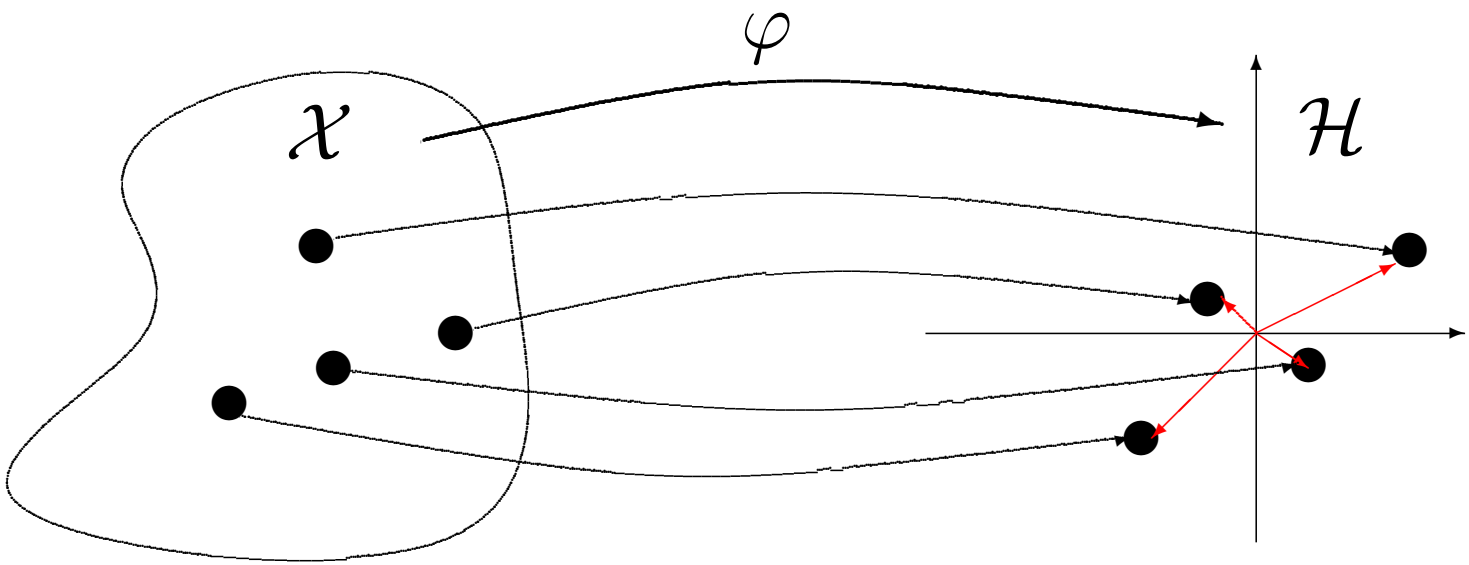
[LeCun et al., 1989, 1998, Ciresan et al., 2012, Krizhevsky et al., 2012]...

Paradigm 2: Kernel methods

$$\min_{f \in \mathcal{H}} \quad \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) \; + \; \lambda \|f\|_{\mathcal{H}}^2.$$

- **map** data x in $\mathcal{X}$ to a Hilbert space and work with **linear forms**:

$$\varphi : \mathcal{X} \to \mathcal{H} \qquad \text{and} \qquad f(x) = \langle \varphi(x), f \rangle_{\mathcal{H}}.$$



[Shawe-Taylor and Cristianini, 2004, Schölkopf and Smola, 2002]...

Paradigm 2: Kernel methods

$$\min_{f \in \mathcal{H}} \quad \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) \; + \; \lambda \|f\|_{\mathcal{H}}^2.$$

First purpose: embed data in a vectorial space where

- many **geometrical operations** exist (angle computation, projection on linear subspaces, definition of barycenters....).
- one may learn potentially **rich infinite-dimensional models**.
- **regularization** is natural (see next...)

Paradigm 2: Kernel methods

$$\min_{f \in \mathcal{H}} \quad \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) \; + \; \lambda \|f\|_{\mathcal{H}}^2.$$

First purpose: embed data in a vectorial space where

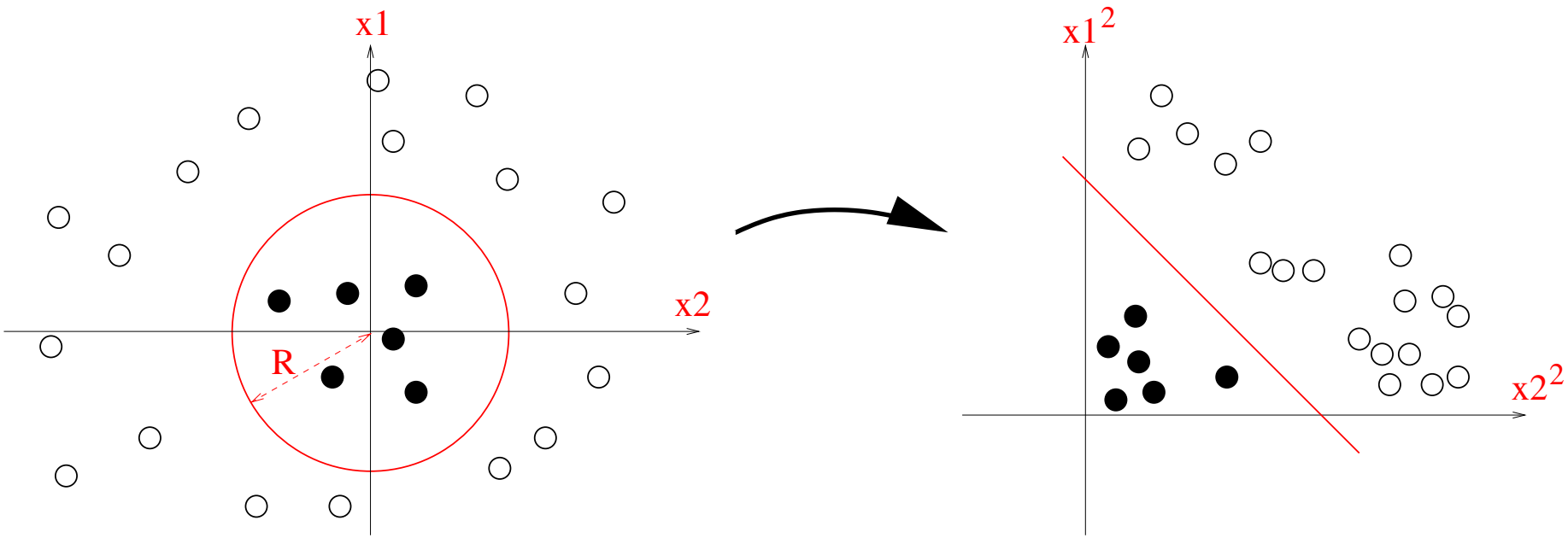
- many **geometrical operations** exist (angle computation, projection on linear subspaces, definition of barycenters....).
- one may learn potentially **rich infinite-dimensional models**.
- **regularization** is natural (see next...)

The principle is **generic** and does not assume anything about the nature of the set $\mathcal{X}$ (vectors, sets, graphs, sequences).

Paradigm 2: Kernel methods

Second purpose: unhappy with the current Euclidean structure?

- lift data to a higher-dimensional space with **nicer properties** (e.g., linear separability, clustering structure).
- then, the **linear** form $f(x) = \langle \varphi(x), f \rangle_{\mathcal{H}}$ in $\mathcal{H}$ may correspond to a **non-linear** model in $\mathcal{X}$.

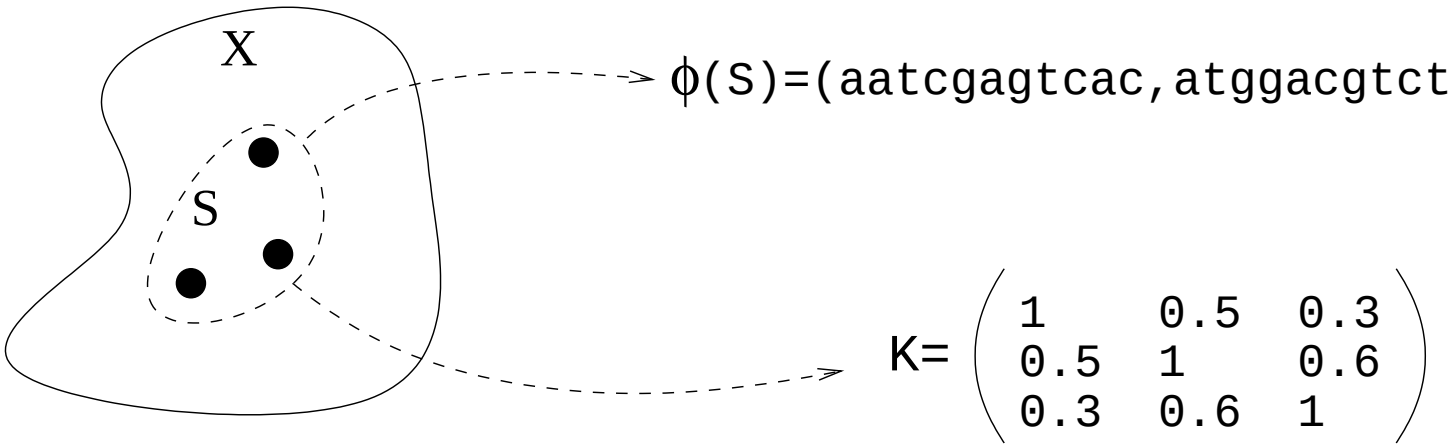


Paradigm 2: Kernel methods

How does it work? representation by pairwise comparisons

- Define a "comparison function": $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.
- Represent a set of n data points $\mathcal{S} = \{x_1, \ldots, x_n\}$ by the $n \times n$ **matrix**:

$$\mathbf{K}_{ij} := K(x_i, x_j).$$

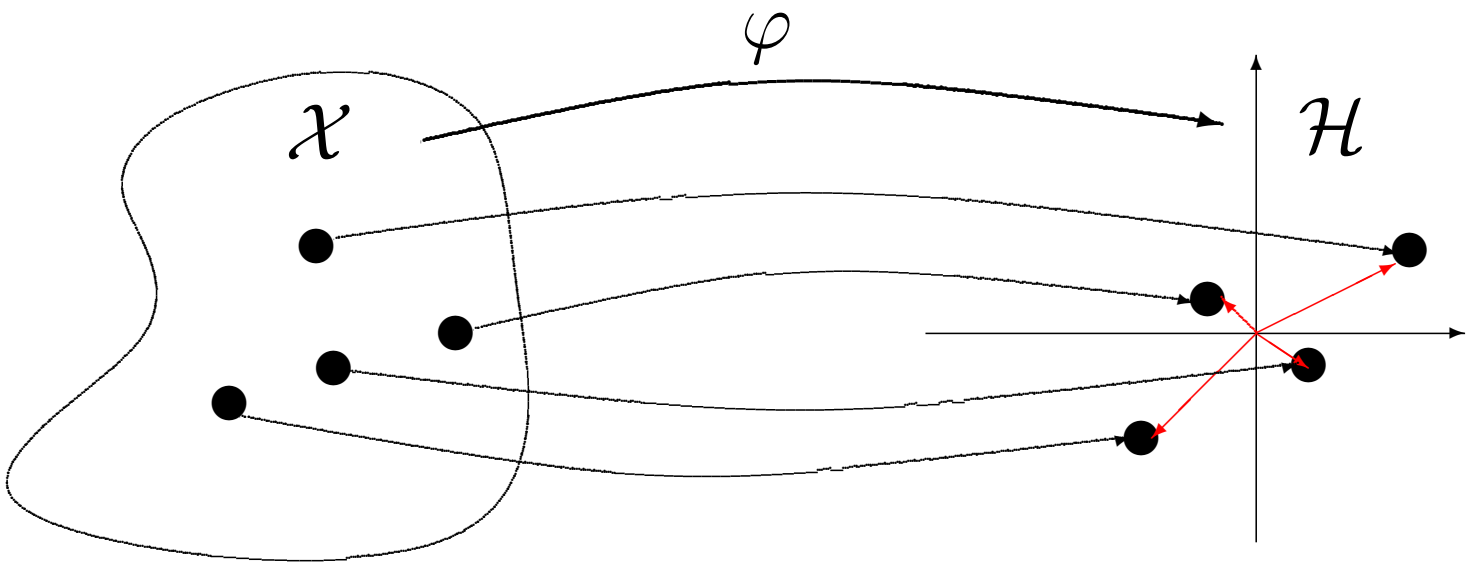


Paradigm 2: Kernel methods

Theorem (Aronszajn, 1950)

$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel if and only if there exists a Hilbert space $\mathcal{H}$ and a mapping $\varphi : \mathcal{X} \to \mathcal{H}$, such that

$$\text{for any } x, x' \text{ in } \mathcal{X}, \qquad K(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}}.$$



Paradigm 2: Kernel methods

Mathematical details

- the only thing we require about K is **symmetry** and **positive definiteness**

$$\forall x_1, \ldots, x_n \in \mathcal{X}, \alpha_1, \ldots, \alpha_n \in \mathbb{R}, \quad \sum_{ij} \alpha_i \alpha_j K(x_i, x_j) \geq 0.$$

- then, there exists a Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$, called the **reproducing kernel Hilbert space (RKHS)** such that

$$\forall f \in \mathcal{H}, x \in \mathcal{X}, \quad f(x) = \langle \varphi(x), f \rangle_{\mathcal{H}},$$

and the mapping $\varphi : \mathcal{X} \to \mathcal{H}$ (from Aronszajn's theorem) satisfies

$$\varphi(x) : y \mapsto K(x, y).$$

Paradigm 2: Kernel methods

Why mapping data in $\mathcal{X}$ to the functional space $\mathcal{H}$?

- it becomes feasible to learn a prediction function $f \in \mathcal{H}$:

$$\min_{f \in \mathcal{H}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \| f \|_{\mathcal{H}}^2}_{\text{regularization}} .$$

  (why? the solution lives in a finite-dimensional hyperplane).

- **non-linear** operations in $\mathcal{X}$ become **inner-products** in $\mathcal{H}$ since

$$\forall f \in \mathcal{H}, x \in \mathcal{X}, \qquad f(x) = \langle \varphi(x), f \rangle_{\mathcal{H}}.$$

- the norm of the RKHS is a **natural regularization function**:

$$|f(x) - f(x')| \leq \|f\|_{\mathcal{H}} \|\varphi(x) - \varphi(x')\|_{\mathcal{H}}.$$

Paradigm 2: Kernel methods

What are the main features of kernel methods?

- builds **well-studied functional spaces** to do machine learning;
- **decoupling** of data representation and learning algorithm;
- typically, **convex optimization problems** in a supervised context;
- **versatility**: applies to vectors, sequences, graphs, sets,. . . ;
- **natural regularization function** to control the learning capacity;

[Shawe-Taylor and Cristianini, 2004, Schölkopf and Smola, 2002, Müller et al., 2001]

Paradigm 2: Kernel methods

What are the main features of kernel methods?

- builds **well-studied functional spaces** to do machine learning;
- **decoupling** of data representation and learning algorithm;
- typically, **convex optimization problems** in a supervised context;
- **versatility**: applies to vectors, sequences, graphs, sets,. . . ;
- **natural regularization function** to control the learning capacity;

But...

- **decoupling** of data representation and learning may not be a good thing, according to recent **supervised** deep learning success.
- requires **kernel design**.
- $O(n^2)$ **scalability problems**.

[Shawe-Taylor and Cristianini, 2004, Schölkopf and Smola, 2002, Müller et al., 2001]

Paradigm 3: The sparsity principle (because of **CoSI**P)

Let us consider again the classical scientific paradigm:

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

[Corfield et al., 2009].

Paradigm 3: The sparsity principle (because of **CoSI**P)

Let us consider again the classical scientific paradigm:

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

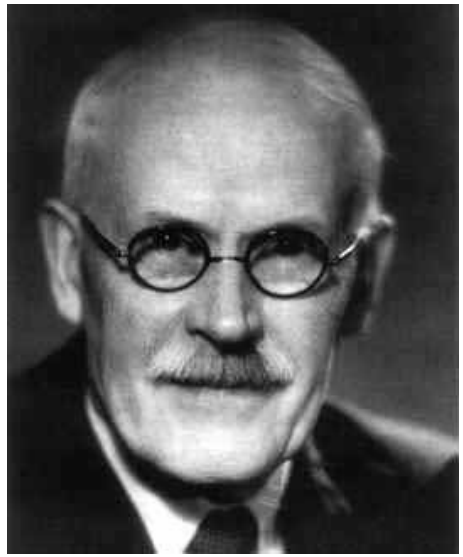
But...

- it is not always possible to distinguish the generalization error of various models based on available data.
- when a complex model A performs slightly better than a simple model B, should we prefer A or B?
- generalization error requires a predictive task: what about unsupervised learning? which measure should we use?
- we are also leaving aside the problem of non i.i.d. train/test data, biased data, testing with counterfactual reasoning...

[Corfield et al., 2009, Bottou et al., 2013, Schölkopf et al., 2012].

Paradigm 3: The sparsity principle (because of **CoSI**P)

(a) Dorothy Wrinch
1894–1980



(b) Harold Jeffreys
1891–1989

The existence of simple laws is, then, apparently, to be regarded as a quality of nature; and accordingly we may infer that it is justifiable to prefer a simple law to a more complex one that fits our observations slightly better.

[Wrinch and Jeffreys, 1921].

Paradigm 3: The sparsity principle (because of **CoSI**P)

Remarks: sparsity is...

- appealing for experimental sciences for **model interpretation**;
- (too-)**well understood** in some mathematical contexts:

$$\min_{w \in \mathbb{R}^p} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L\left(y_i, w^\top x_i\right)}_{\text{empirical risk, data fit}} + \underbrace{\lambda \|w\|_1}_{\text{regularization}} .$$

- extremely powerful for **unsupervised learning** in the context of matrix factorization, and **simple to use**.

[Olshausen and Field, 1996, Chen, Donoho, and Saunders, 1999, Tibshirani, 1996]...

Paradigm 3: The sparsity principle (because of **CoSIP**)

Remarks: sparsity is...

- appealing for experimental sciences for **model interpretation**;
- (too-)**well understood** in some mathematical contexts:

$$\min_{w \in \mathbb{R}^p} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L\left(y_i, w^\top x_i\right)}_{\text{empirical risk, data fit}} + \underbrace{\lambda \|w\|_1}_{\text{regularization}} .$$

- extremely powerful for **unsupervised learning** in the context of matrix factorization, and **simple to use**.

Today's challenges

- Develop sparse and **stable** (and **invariant?**) models.
- Go beyond clustering / low-rank / union of subspaces.

[Olshausen and Field, 1996, Chen, Donoho, and Saunders, 1999, Tibshirani, 1996]...

Some references

On kernel methods

- B. Schölkopf and A. J. Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. 2002.
- J. Shawe-Taylor and N. Cristianini. An introduction to support vector machines and other kernel-based learning methods. 2004.
- 635 slides: `http://members.cbio.mines-paristech.fr/~jvert/svn/kernelcourse/course/2017mva/index.html`

On sparse estimation

- M. Elad. Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing. 2010.
- J. Mairal, F. Bach, and J. Ponce. Sparse Modeling for Image and Vision Processing. 2014. **free online**.

Part II: Convolutional Kernel Networks

Challenges of deep kernel machines

- **Build functional spaces for deep learning**, where we can quantify **invariance and stability** to perturbations, **signal recovery** properties, and the **complexity** of the function class.
- do deep learning with a **geometrical interpretation** (learn collections of linear subspaces, perform projections).
- exploit kernels for **structured objects** (graph, sequences) within deep architectures.
- show that **end-to-end** learning is natural with kernel methods.

Convolutional Kernel Networks

The (happy?) **marriage** of kernel methods and CNNs

1. **a multilayer convolutional kernel for images**: A hierarchy of kernels for local image neighborhoods (aka, receptive fields).
2. **unsupervised scheme for large-scale learning**: the kernel beeing too computationally expensive, the Nyström approximation at each layer yields a new type of unsupervised deep neural network.
3. **end-to-end learning**: learning subspaces in the RKHSs can be achieved with a supervised loss function.

Convolutional Kernel Networks

The (happy?) **marriage** of kernel methods and CNNs

1. **a multilayer convolutional kernel for images**: A hierarchy of kernels for local image neighborhoods (aka, receptive fields).
2. **unsupervised scheme for large-scale learning**: the kernel beeing too computationally expensive, the Nyström approximation at each layer yields a new type of unsupervised deep neural network.
3. **end-to-end learning**: learning subspaces in the RKHSs can be achieved with a supervised loss function.

First proof of concept with unsupervised learning

- J. Mairal, P. Koniusz, Z. Harchaoui and C. Schmid. Convolutional Kernel Networks. NIPS 2014.

Application to image retrieval

- M. Paulin, J. Mairal, M. Douze, Z. Harchaoui, F. Perronnin, and C. Schmid. Convolutional Patch Representations for Image Retrieval: an Unsupervised Approach. IJCV. 2017.

Convolutional Kernel Networks

The (happy?) **marriage** of kernel methods and CNNs

1. **a multilayer convolutional kernel for images**: A hierarchy of kernels for local image neighborhoods (aka, receptive fields).
2. **unsupervised scheme for large-scale learning**: the kernel beeing too computationally expensive, the Nyström approximation at each layer yields a new type of unsupervised deep neural network.
3. **end-to-end learning**: learning subspaces in the RKHSs can be achieved with a supervised loss function.

Conceptually better model, with supervised learning

- J. Mairal. End-to-End Kernel Learning with Supervised Convolutional Kernel Networks. NIPS 2016.

Application to biological sequences

- D. Chen, L. Jacob, and J. Mairal. Predicting Transcription Factor Binding Sites with Convolutional Kernel Networks. preprint BiorXiv. 2017.

Convolutional Kernel Networks

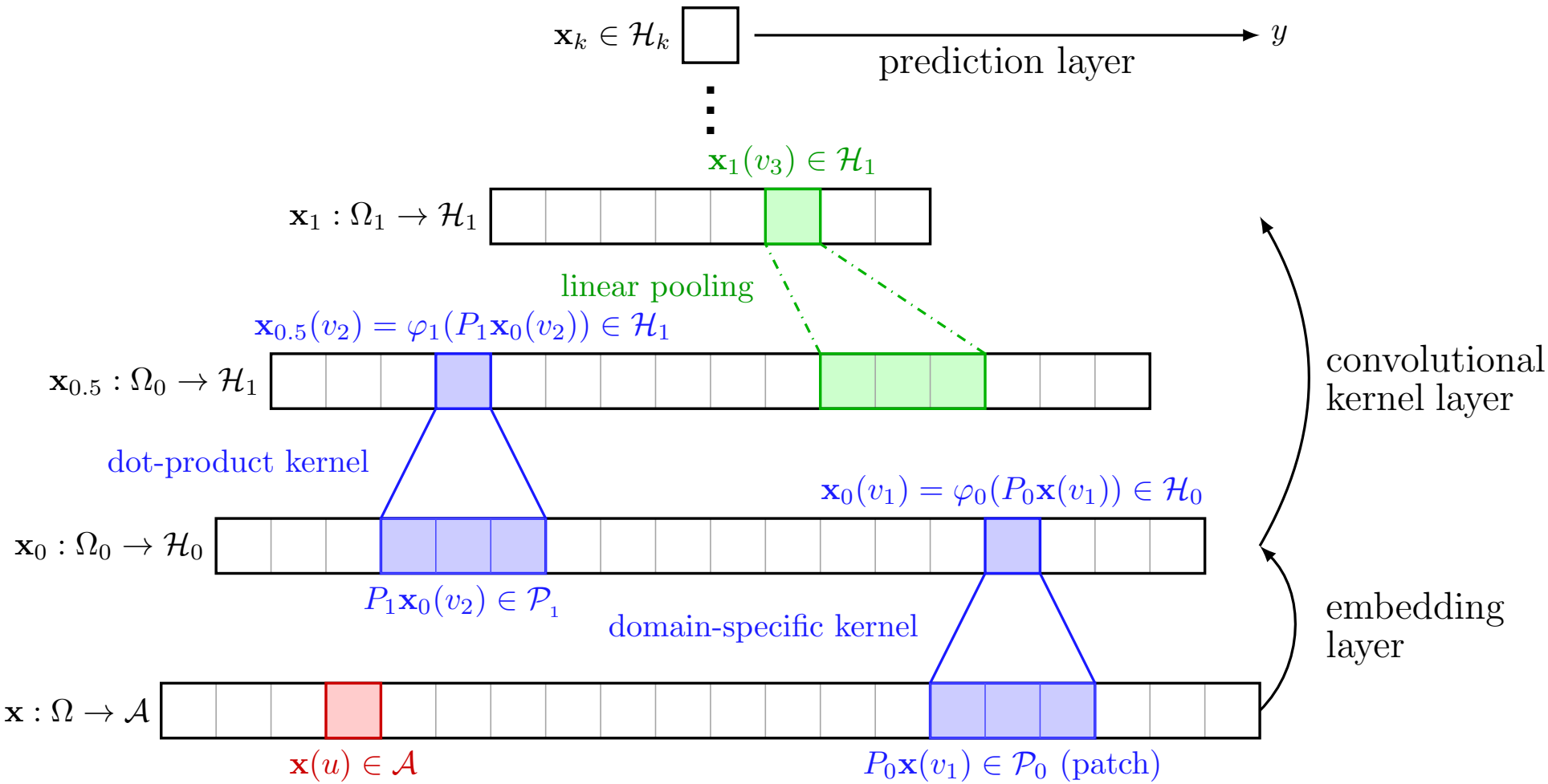


Illustration of multilayer convolutional kernel for 1D discrete signals.

Convolutional Kernel Networks

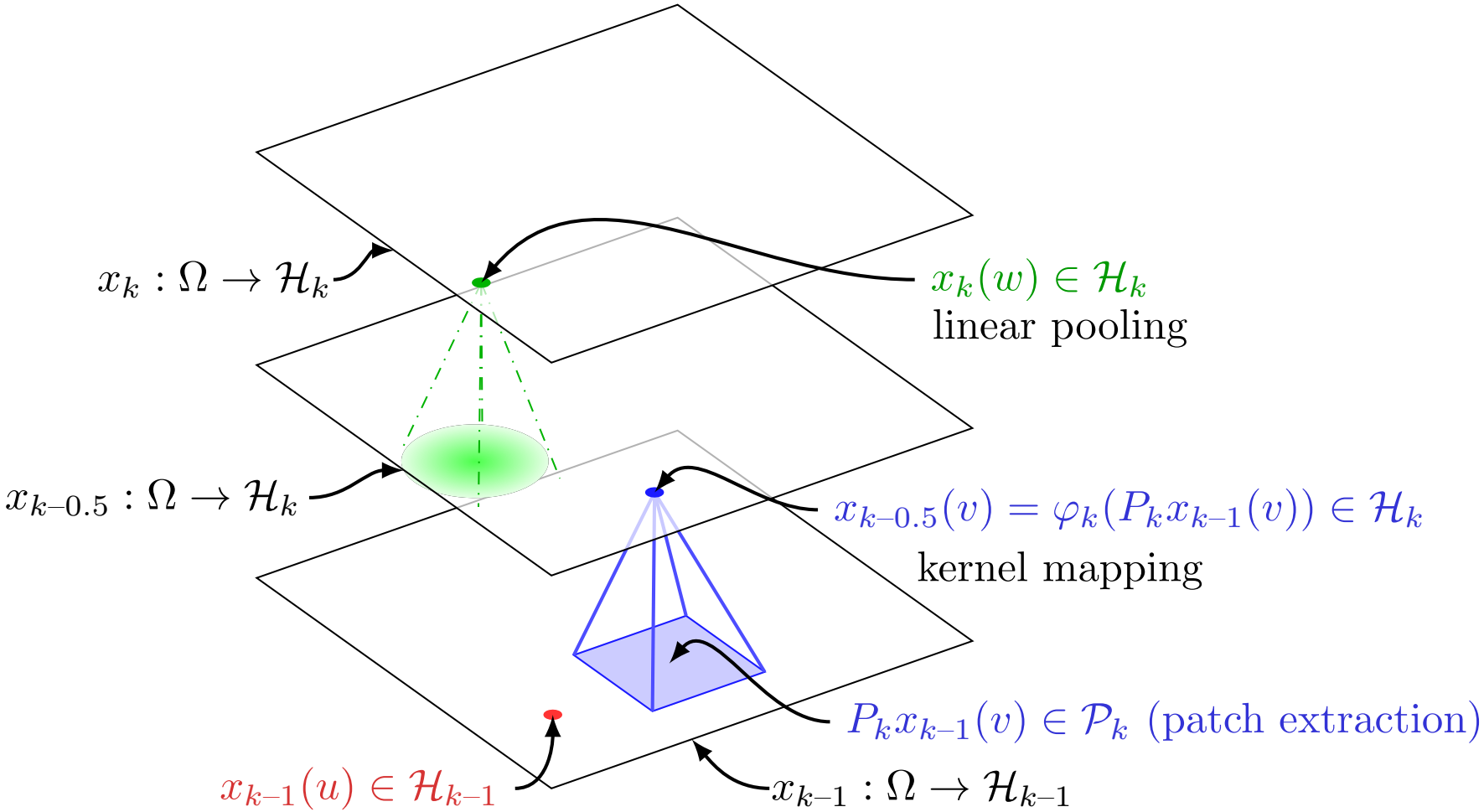
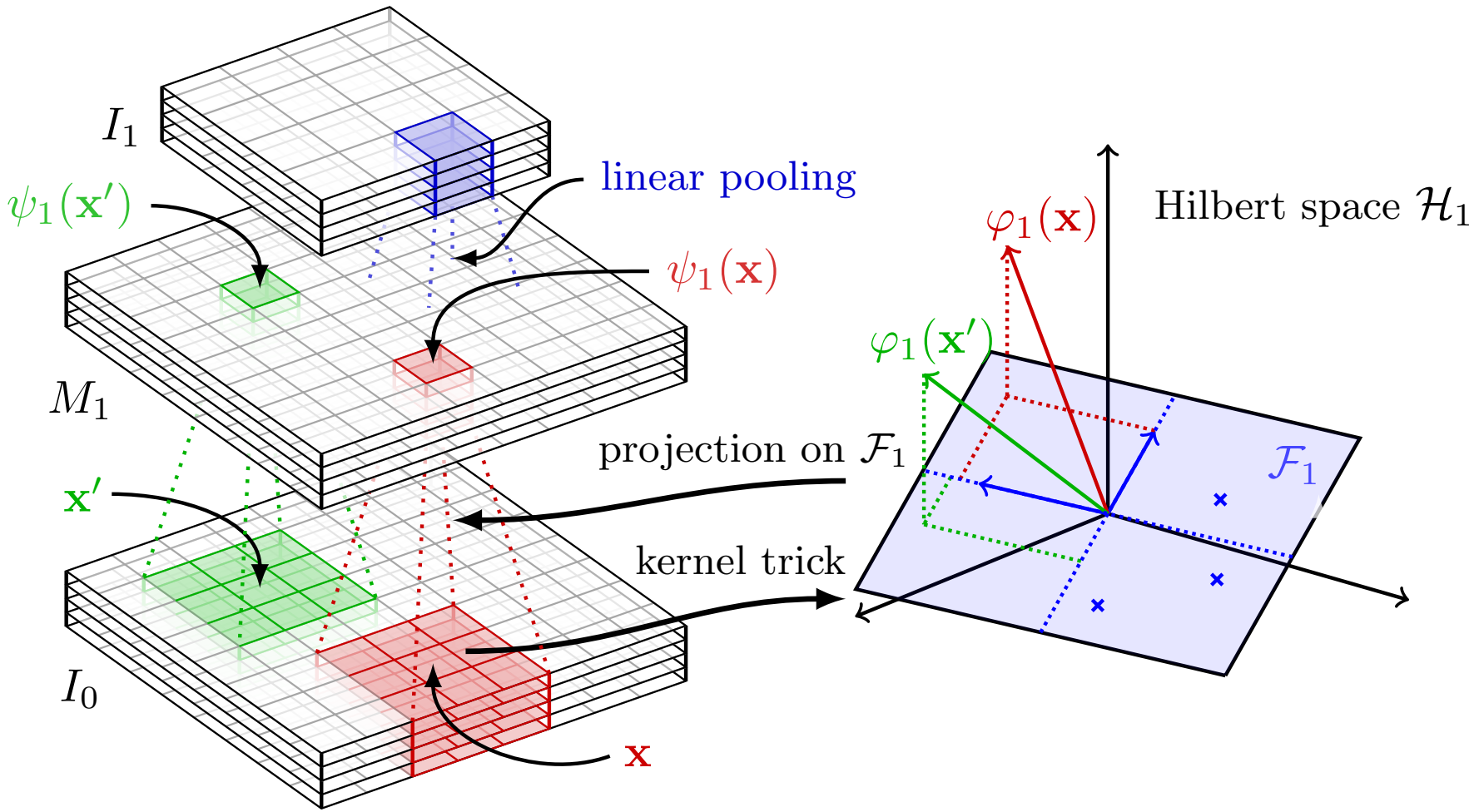


Illustration of multilayer convolutional kernel for 2D continuous signals.

Convolutional Kernel Networks



Learning mechanism of CKNs between layers 0 and 1.

Convolutional Kernel Networks

Main principles

- A multilayer kernel, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).

Convolutional Kernel Networks

Main principles

- A multilayer kernel, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- When going up in the hierarchy, we represent **larger neighborhoods with more invariance**;

Convolutional Kernel Networks

Main principles

- A multilayer kernel, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- When going up in the hierarchy, we represent **larger neighborhoods with more invariance**;
- The first layer may encode **domain-specific knowledge**;

Convolutional Kernel Networks

Main principles

- A multilayer kernel, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- When going up in the hierarchy, we represent **larger neighborhoods with more invariance**;
- The first layer may encode **domain-specific knowledge**;
- We build a sequence of functional spaces and data representations that are **decoupled from learning**...

Convolutional Kernel Networks

Main principles

- A multilayer kernel, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- When going up in the hierarchy, we represent **larger neighborhoods with more invariance**;
- The first layer may encode **domain-specific knowledge**;
- We build a sequence of functional spaces and data representations that are **decoupled from learning**...
- But, we learn **linear subspaces** in RKHSs, where we project data, providing a new type of CNN with a **geometric interpretation**.

Convolutional Kernel Networks

Main principles

- A multilayer kernel, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- When going up in the hierarchy, we represent **larger neighborhoods with more invariance**;
- The first layer may encode **domain-specific knowledge**;
- We build a sequence of functional spaces and data representations that are **decoupled from learning**...
- But, we learn **linear subspaces** in RKHSs, where we project data, providing a new type of CNN with a **geometric interpretation**.
- Learning may be **unsupervised** (reduce approximation error) or **supervised** (via backpropagation).

Basic component: dot-product kernels

A simple link between kernels and neural networks can be obtained by considering dot-product kernels.

A classical old result [Schoenberg, 1942]

Let $\mathcal{X} = \mathbb{S}^{d-1}$ be the unit sphere of $\mathbb{R}^d$. The kernel $K : \mathcal{X}^2 \to \mathbb{R}$

$$K(x, y) = \kappa(\langle x, y \rangle_{\mathbb{R}^d})$$

is positive definite if and only if κ is smooth and its Taylor expansion coefficients are non-negative.

Remark

- the proposition holds if $\mathcal{X}$ is the unit sphere of some Hilbert space and $\langle x, y \rangle_{\mathbb{R}^d}$ is replaced by the corresponding inner-product.

[Smola, Ovari, and Williamson, 2001]...

Basic component: dot-product kernels

linear kernel	$\langle z, z' \rangle$
exponential kernel	$e^{\alpha(\langle z,z' \rangle - 1)}$
inverse polynomial kernel	$\frac{1}{2-\langle z,z' \rangle}$
polynomial kernel of degree p	$(c + \langle z, z' \rangle)^p$
arc-cosine kernel of degree 1	$\frac{1}{\pi}(\sin(\theta) + (\pi - \theta)\cos(\theta))$ with $\theta = \arccos(\langle z, z' \rangle)$
Vovk's kernel of degree 3	$\frac{1}{3}\left(\frac{1-\langle z,z' \rangle^3}{1-\langle z,z' \rangle}\right) = \frac{1}{3}\left(1 + \langle z, z' \rangle + \langle z, z' \rangle^2\right)$

Basic component: dot-product kernels

linear kernel	$\langle z, z' \rangle$
exponential kernel	$e^{\alpha(\langle z,z' \rangle - 1)}$
inverse polynomial kernel	$\frac{1}{2-\langle z,z' \rangle}$
polynomial kernel of degree p	$(c + \langle z, z' \rangle)^p$
arc-cosine kernel of degree 1	$\frac{1}{\pi}\left(\sin(\theta) + (\pi - \theta)\cos(\theta)\right)$ with $\theta = \arccos(\langle z, z' \rangle)$
Vovk's kernel of degree 3	$\frac{1}{3}\left(\frac{1-\langle z,z' \rangle^3}{1-\langle z,z' \rangle}\right) = \frac{1}{3}\left(1 + \langle z, z' \rangle + \langle z, z' \rangle^2\right)$

Remark

if $\|z\| = \|z'\| = 1$, the exponential kernel recovers the Gaussian kernel

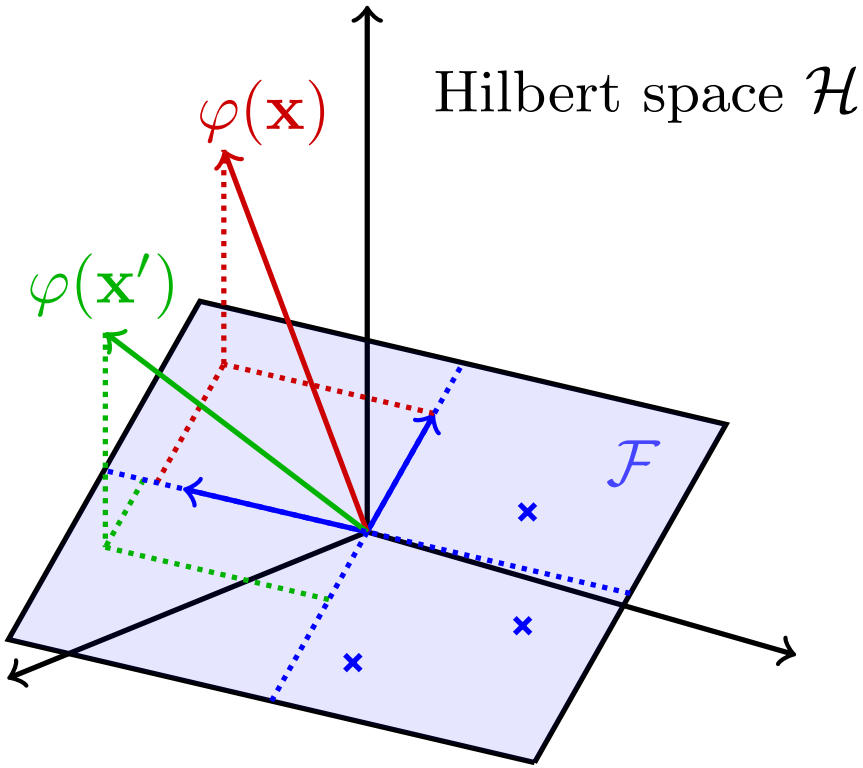
$$\kappa_{\text{exp}}(\langle z, z' \rangle) = e^{\alpha(\langle z,z' \rangle - 1)} = e^{-\frac{\alpha}{2}\|z-z'\|^2},$$

Basic component: dot-product kernels + Nyström

The Nyström method consists of replacing any point $\varphi(x)$ in $\mathcal{H}$, for x in $\mathcal{X}$ by its orthogonal projection onto a finite-dimensional subspace

$$\mathcal{F} = \text{span}(\varphi(z_1), \ldots, \varphi(z_p)),$$

for some anchor points $Z = [z_1, \ldots, z_p]$ in $\mathbb{R}^{d \times p}$



Basic component: dot-product kernels + Nyström

The projection is equivalent to

$$\Pi_{\mathcal{F}}[x] \triangleq \sum_{j=1}^{p} \beta_j^\star \varphi(z_j) \quad \text{with} \quad \beta^\star \in \arg\min_{\beta \in \mathbb{R}^p} \left\| \varphi(x) - \sum_{j=1}^{p} \beta_j \varphi(z_j) \right\|_{\mathcal{H}}^2,$$

Then, it is possible to show that with $K(x,y) = \langle \varphi(x), \varphi(y) \rangle_{\mathcal{H}}$,

$$K(x,y) \approx \langle \Pi_{\mathcal{F}}[x], \Pi_{\mathcal{F}}[y] \rangle_{\mathcal{H}} = \langle \psi(x), \psi(y) \rangle_{\mathbb{R}^p},$$

with

$$\psi(x) = \kappa(Z^\top Z)^{-1/2} \kappa(Z^\top x),$$

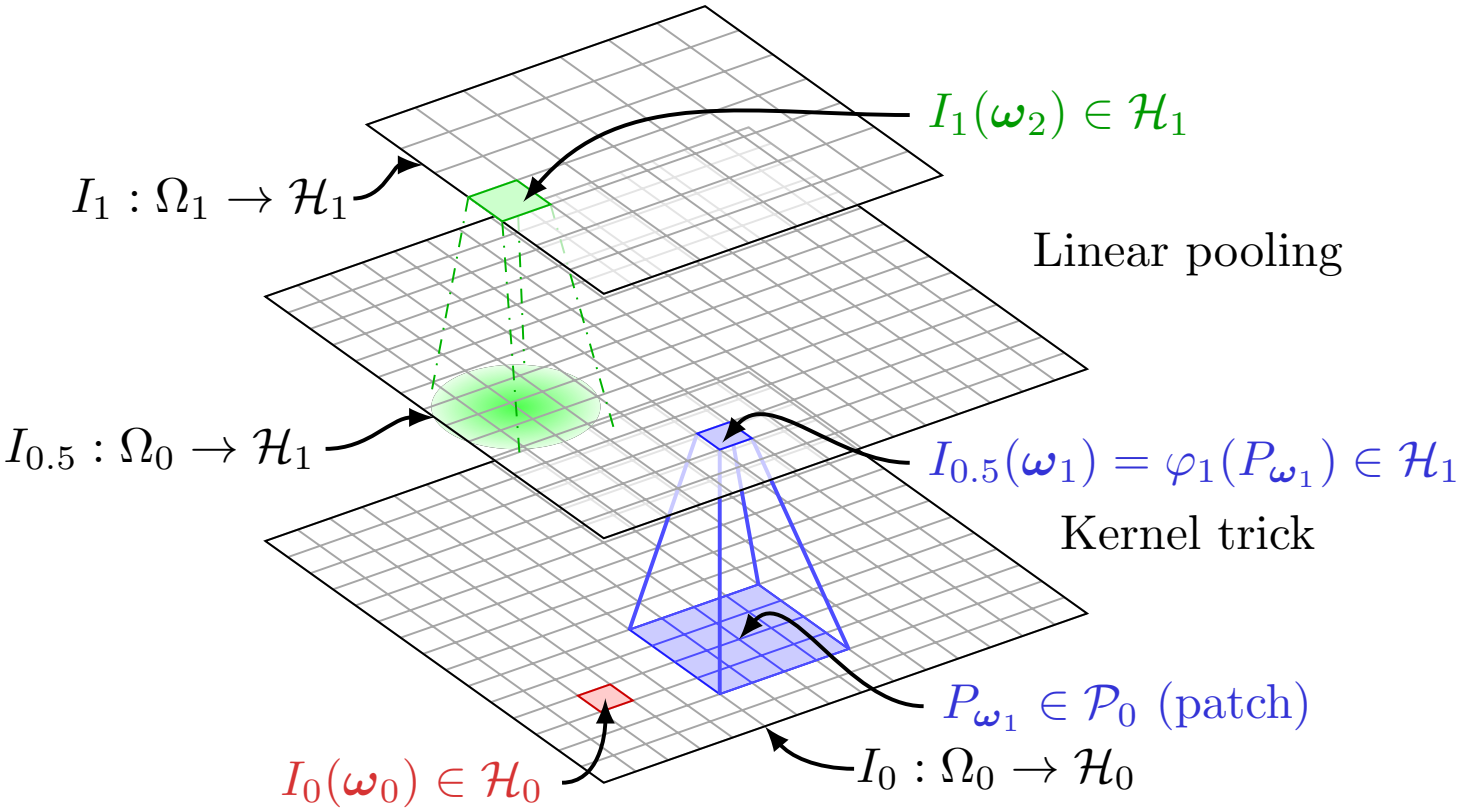
where the function κ is applied pointwise to its arguments. The resulting ψ can be interpreted as a neural network performing (i) linear operation, (ii) pointwise non-linearity, (iii) linear operation.

[Williams and Seeger, 2001, Smola and Schölkopf, 2000, Fine and Scheinberg, 2001].

The multilayer convolutional kernel

Definition: image feature maps

An image feature map is a function $I : \Omega \to \mathcal{H}$, where Ω is a 2D grid representing "coordinates" in the image and $\mathcal{H}$ is a Hilbert space.



The multilayer convolutional kernel

Definition: image feature maps

An image feature map is a function $I : \Omega \to \mathcal{H}$, where Ω is a 2D grid representing "coordinates" in the image and $\mathcal{H}$ is a Hilbert space.

Motivation and examples

- Each point $I(\omega)$ carries information about an image neighborhood, which is motivated by the **local stationarity** of natural images.
- We will construct a sequence of maps $I_0, \ldots, I_k$. Going up in the hierarchy yields **larger receptive fields** with **more invariance**.
- I_0 may simply be the input image, where $\mathcal{H}_0 = \mathbb{R}^3$ for RGB.

The multilayer convolutional kernel

Definition: image feature maps

An image feature map is a function $I : \Omega \to \mathcal{H}$, where Ω is a 2D grid representing "coordinates" in the image and $\mathcal{H}$ is a Hilbert space.

Motivation and examples

- Each point $I(\omega)$ carries information about an image neighborhood, which is motivated by the **local stationarity** of natural images.
- We will construct a sequence of maps $I_0, \ldots, I_k$. Going up in the hierarchy yields **larger receptive fields** with **more invariance**.
- I_0 may simply be the input image, where $\mathcal{H}_0 = \mathbb{R}^3$ for RGB.

How do we go from $I_0 : \Omega_0 \to \mathcal{H}_0$ to $I_1 : \Omega_1 \to \mathcal{H}_1$?

The multilayer convolutional kernel

Definition: image feature maps

An image feature map is a function $I : \Omega \to \mathcal{H}$, where Ω is a 2D grid representing "coordinates" in the image and $\mathcal{H}$ is a Hilbert space.

Motivation and examples

- Each point $I(\omega)$ carries information about an image neighborhood, which is motivated by the **local stationarity** of natural images.
- We will construct a sequence of maps $I_0, \ldots, I_k$. Going up in the hierarchy yields **larger receptive fields** with **more invariance**.
- I_0 may simply be the input image, where $\mathcal{H}_0 = \mathbb{R}^3$ for RGB.

How do we go from $I_0 : \Omega_0 \to \mathcal{H}_0$ to $I_1 : \Omega_1 \to \mathcal{H}_1$?

First, define a p.d. kernel on patches of I_0!

The multilayer convolutional kernel

Going from I_0 to $I_{0.5}$: kernel trick

- Patches of size $e_0 \times e_0$ can be defined as elements of the **Cartesian product** $\mathcal{P}_0 \triangleq \mathcal{H}_0^{e_0 \times e_0}$ endowed with its natural inner-product.
- **Define a p.d. kernel on such patches**: For all x, x' in $\mathcal{P}_0$,

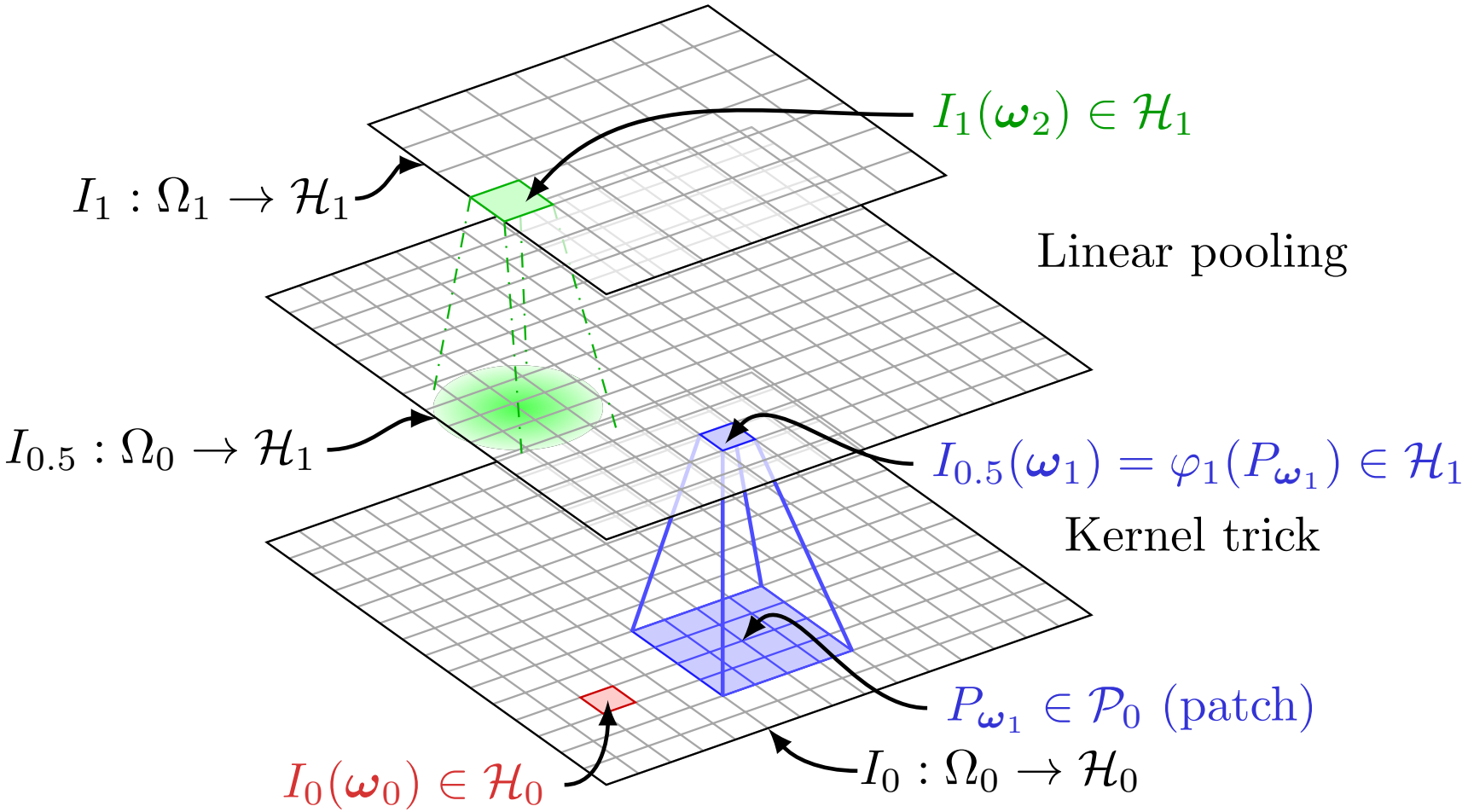
$$K_1(x,x') = \|x\|_{\mathcal{P}_0}\|x'\|_{\mathcal{P}_0}\kappa_1\left(\frac{\langle x, x'\rangle_{\mathcal{P}_0}}{\|x\|_{\mathcal{P}_0}\|x'\|_{\mathcal{P}_0}}\right) \quad \text{if } x, x' \neq 0 \quad \text{and } 0 \text{ otherwise.}$$

Note that for y, y' normalized, we may choose

$$\kappa_1\left(\langle y, y'\rangle_{\mathcal{P}_0}\right) = e^{\alpha_1\left(\langle y, y'\rangle_{\mathcal{P}_0} - 1\right)} = e^{-\frac{\alpha_1}{2}\|y - y'\|^2_{\mathcal{P}_0}}.$$

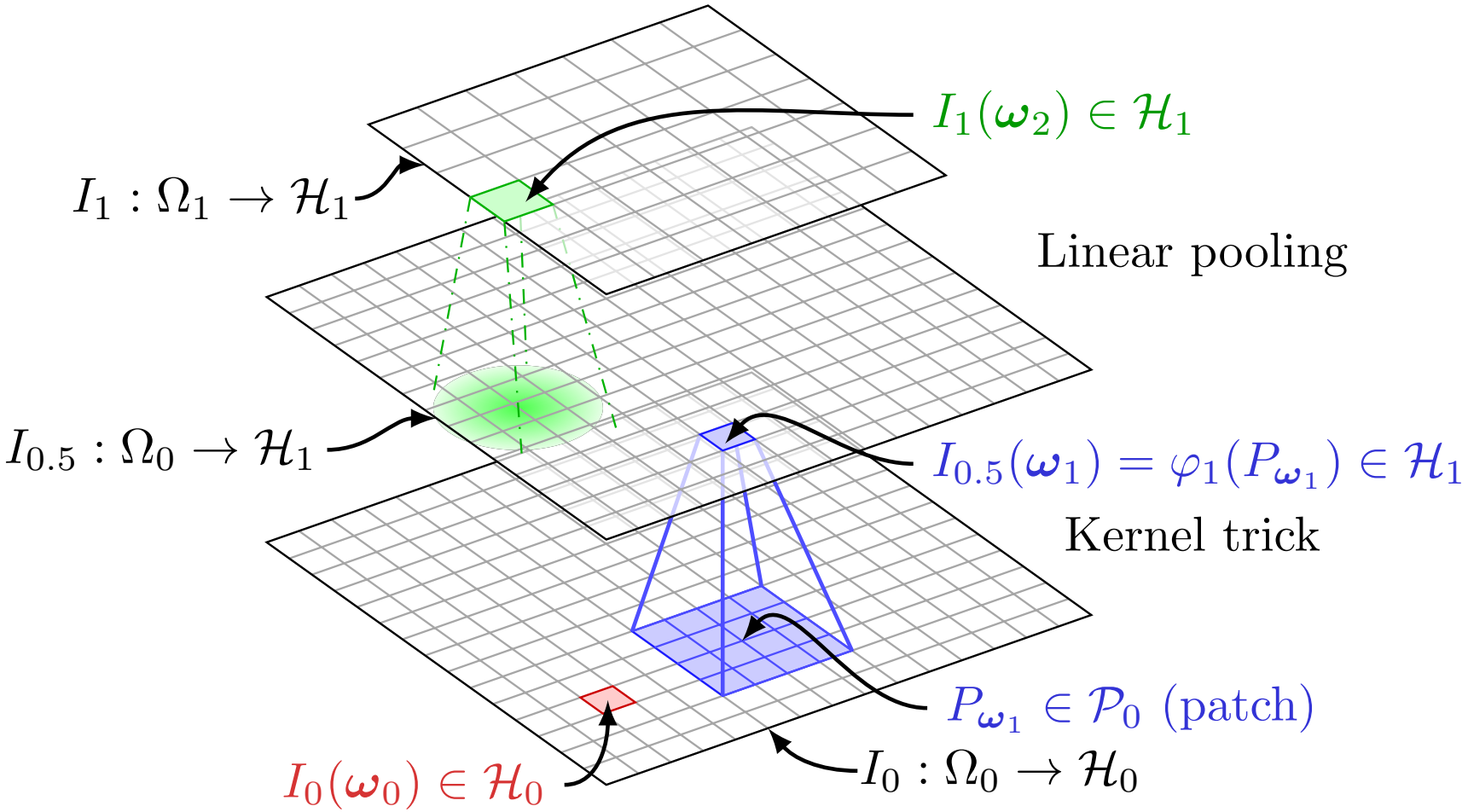
- **We call** $\mathcal{H}_1$ the RKHS and define a **mapping** $\varphi_1 : \mathcal{P}_0 \to \mathcal{H}_1$.
- Then, we may define the map $I_{0.5} : \Omega_0 \to \mathcal{H}_1$ that carries the representations in $\mathcal{H}_1$ of the patches from I_0 at all locations in Ω_0.

The multilayer convolutional kernel



How do we go from $I_{0.5} : \Omega_0 \to \mathcal{H}_1$ to $I_1 : \Omega_1 \to \mathcal{H}_1$?

The multilayer convolutional kernel



How do we go from $I_{0.5} : \Omega_0 \to \mathcal{H}_1$ to $I_1 : \Omega_1 \to \mathcal{H}_1$?

Linear pooling!

The multilayer convolutional kernel

Going from $I_{0.5}$ to I_1: linear pooling

- For all ω in Ω_1:

$$I_1(\omega) = \sum_{\omega' \in \Omega_0} I_{0.5}(\omega') e^{-\beta_1 \|\omega' - \omega\|_2^2}.$$

- The Gaussian weight can be interpreted as an anti-aliasing filter for downsampling the map $I_{0.5}$ to a different resolution.
- Linear pooling is compatible with the kernel interpretation: linear combinations of points in the RKHS are still points in the RKHS.

Finally,

- We may now repeat the process and build $I_0, I_1, \ldots, I_k$.
- and obtain the **multilayer convolutional kernel**

$$K(I_k, I_k') = \sum_{\omega \in \Omega_k} \langle I_k(\omega), I_k'(\omega) \rangle_{\mathcal{H}_k}.$$

The multilayer convolutional kernel

In summary

- The multilayer convolutional kernel builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- Invariance to local translations is achieved through **linear pooling** in the RKHS.
- It remains a **conceptual object** due to its high complexity.
- **Learning and modelling are still decoupled**.

Let us first address the second point (scalability).

Unsupervised learning for convolutional kernel networks

Learn linear subspaces of finite-dimensions where we project the data

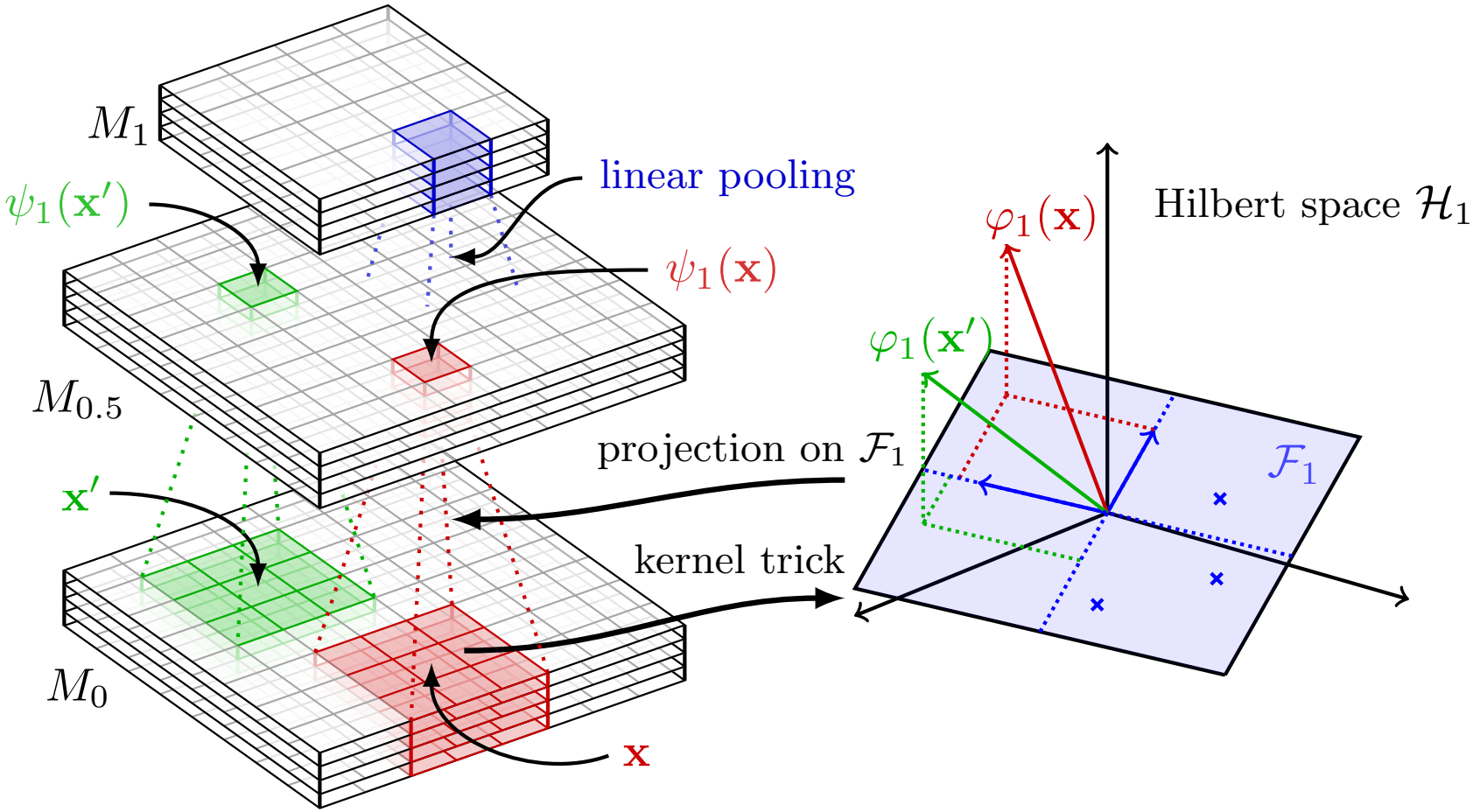


Figure: The convolutional kernel network model between layers 0 and 1.

Unsupervised learning for convolutional kernel networks

Formally, this means using the Nyström approximation

- We now manipulate **finite-dimensional maps** $M_j : \Omega_j \to \mathbb{R}^{p_j}$.
- Every linear subspace is parametrized by anchor points

$$\mathcal{F}_j \triangleq \text{Span}\left(\varphi(z_{j,1}), \ldots, \varphi(z_{j,p_j})\right),$$

  where the $z_{1,j}$'s are in $\mathbb{R}^{p_{j-1}e_{j-1}^2}$ for patches of size $e_{j-1} \times e_{j-1}$.
- The encoding function at layer j is

$$\psi_j(x) \triangleq \|x\|\kappa_j(Z_j^\top Z_j)^{-1/2}\kappa_1\left(Z_j^\top \frac{x}{\|x\|}\right) \text{ if } x \neq 0 \text{ and } 0 \text{ otherwise,}$$

  where $Z_j = [z_{j,1}, \ldots, z_{j,p_j}]$ and $\|.\|$ is the Euclidean norm.
- The interpretation is **convolution** with filters Z_j, **pointwise non-linearity**, 1×1 **convolution**, **contrast normalization**.

Unsupervised learning for convolutional kernel networks

- The pooling operation keeps points in the linear subspace $\mathcal{F}_j$, and pooling $M_{0.5} : \Omega_0 \to \mathbb{R}^{p_1}$ is equivalent to pooling $I_{0.5} : \Omega_0 \to \mathcal{H}_1$.

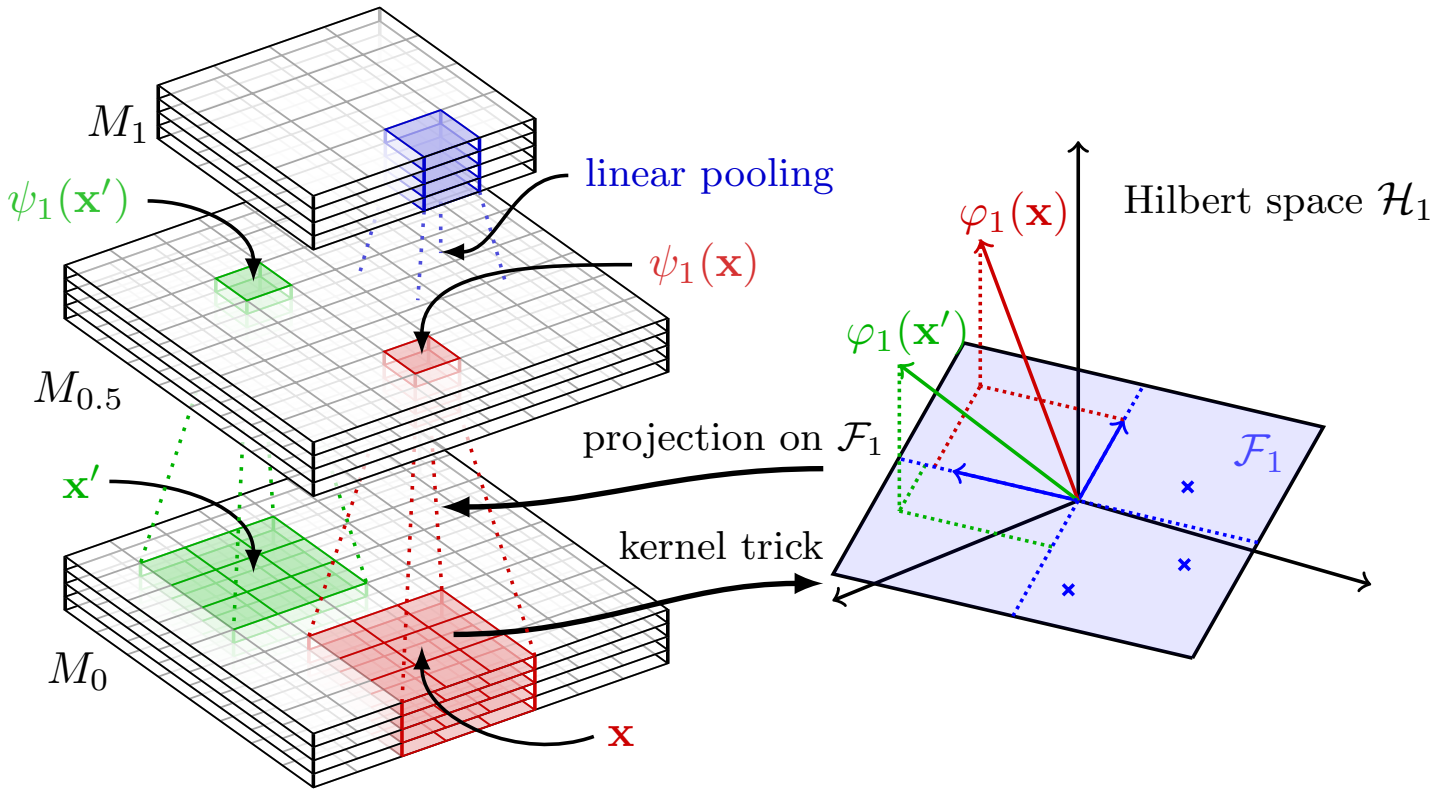


Figure: The convolutional kernel network model between layers 0 and 1.

Unsupervised learning for convolutional kernel networks

How do we learn the filters **with no** supervision?

we learn one layer at a time, starting from the bottom one.

- we **extract a large number**—say $100\,000$ patches from layers $j-1$ computed on an image database and normalize them;
- perform a **spherical K-means algorithm** to learn the filters Z_j;
- **compute the projection matrix** $\kappa_j(Z_j^\top Z_j)^{-1/2}$.

Remarks

- with kernels, we map **patches in infinite dimension**; with the projection, we **manipulate finite-dimensional objects**.
- we obtain an **unsupervised** convolutional net with a **geometric interpretation**, where we perform projections in the RKHSs.

Unsupervised learning for convolutional kernel networks

Remark on input image pre-processing

Unsupervised CKNs are sensitive to pre-processing; we have tested

- RAW RGB input;
- local **centering** of every color channel;
- local **whitening** of each color channel;
- 2D **image gradients**.

(a) RAW RGB

(b) centering

Unsupervised learning for convolutional kernel networks

Remark on input image pre-processing

Unsupervised CKNs are sensitive to pre-processing; we have tested

- RAW RGB input;
- local **centering** of every color channel;
- local **whitening** of each color channel;
- 2D **image gradients**.

(c) RAW RGB

(d) whitening

Unsupervised learning for convolutional kernel networks

Remark on pre-processing with image gradients and 1×1 patches

- Every pixel/patch can be represented as a two dimensional vector

$$x = \rho[\cos(\theta), \sin(\theta)],$$

  where $\rho = \|x\|$ is the gradient intensity and θ is the orientation.
- A natural choice of filters Z would be

$$z_j = [\cos(\theta_j), \sin(\theta_j)] \quad \text{with} \quad \theta_j = 2j\pi/p_0.$$

- Then, the vector $\psi(x) = \|x\|\kappa_1(Z^\top Z)^{-1/2}\kappa_1\left(Z^\top \frac{x}{\|x\|}\right)$, can be interpreted as a **"soft-binning"** of the gradient orientation.
- After pooling, the **representation of this first layer is very close to SIFT/HOG descriptors** [see Bo et al., 2011].

Convolutional kernel networks with supervised learning

How do we learn the filters **with** supervision?

- Given a kernel K and RKHS $\mathcal{H}$, the ERM objective is

$$\min_{f \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\frac{\lambda}{2} \|f\|_{\mathcal{H}}^2}_{\text{regularization}} .$$

- here, we use the parametrized kernel

$$K_{\mathcal{Z}}(I_0, I_0') = \sum_{\omega \in \Omega_k} \langle M_k(\omega), M_k'(\omega) \rangle = \langle M_k, M_k' \rangle_{\mathsf{F}},$$

- and we obtain the simple formulation

$$\min_{W \in \mathbb{R}^{p_k \times |\Omega_k|}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, \langle W, M_k^i \rangle_{\mathsf{F}}) + \frac{\lambda}{2} \|W\|_{\mathsf{F}}^2. \qquad (1)$$

Convolutional kernel networks with supervised learning

How do we learn the filters **with** supervision?

- we **jointly optimize** w.r.t. $\mathcal{Z}$ (set of filters) and W.
- we **alternate** between the optimization of $\mathcal{Z}$ and of W;
- for W, the problem is strongly-convex and can be tackled with recent algorithms that are much faster than SGD;
- for $\mathcal{Z}$, we derive **backpropagation rules** and use classical tricks for learning CNNs (SGD+momentum);

The only tricky part is to differentiate $\kappa_j(Z_j^\top Z_j)^{-1/2}$ w.r.t Z_j, which is a non-standard operation in classical CNNs.

Convolutional kernel networks

In summary

- a multilayer kernel for images, which builds upon similar principles as a convolutional neural net (**multiscale, local stationarity**).
- A new type of convolutional neural network with a geometric interpretation: **orthogonal projections in RKHS**.
- Learning may be unsupervised: **align subspaces with data**.
- Learning may be supervised: **subspace learning in RKHSs**.

Related work on deep kernel machines

Related work

- proof of concept for combining kernels and deep learning [Cho and Saul, 2009];
- hierarchical kernel descriptors [Bo et al., 2011];
- other multilayer models [Bouvrie et al., 2009, Montavon et al., 2011, Anselmi et al., 2015];
- deep Gaussian processes [Damianou and Lawrence, 2013].
- multilayer PCA [Schölkopf et al., 1998].
- old kernels for images [Scholkopf, 1997].
- RBF networks [Broomhead and Lowe, 1988].

Related work on deep kernel machines

Composition of feature spaces

Consider a p.d. kernel $K_1 : \mathcal{X}^2 \to \mathbb{R}$ and its RKHS $\mathcal{H}_1$ with mapping $\varphi_1 : \mathcal{X} \to \mathcal{H}_1$. Consider also a p.d. kernel $K_2 : \mathcal{H}_1^2 \to \mathbb{R}$ and its RKHS $\mathcal{H}_2$ with mapping $\varphi_2 : \mathcal{H}_1 \to \mathcal{H}_2$. Then, $K_3 : \mathcal{X}^2 \to \mathbb{R}$ below is also p.d.

$$K_3(x, x') = K_2(\varphi_1(x), \varphi_1(x')),$$

Related work on deep kernel machines

Composition of feature spaces

Consider a p.d. kernel $K_1 : \mathcal{X}^2 \to \mathbb{R}$ and its RKHS $\mathcal{H}_1$ with mapping $\varphi_1 : \mathcal{X} \to \mathcal{H}_1$. Consider also a p.d. kernel $K_2 : \mathcal{H}_1^2 \to \mathbb{R}$ and its RKHS $\mathcal{H}_2$ with mapping $\varphi_2 : \mathcal{H}_1 \to \mathcal{H}_2$. Then, $K_3 : \mathcal{X}^2 \to \mathbb{R}$ below is also p.d.

$$K_3(x, x') = K_2(\varphi_1(x), \varphi_1(x')),$$

Examples

$$K_3(x, x') = e^{-\frac{1}{2\sigma^2}\|\varphi_1(x) - \varphi_1(x')\|^2_{\mathcal{H}_1}}.$$

$$K_3(x, x') = \langle \varphi_1(x), \varphi_1(x') \rangle^2_{\mathcal{H}_1} = K_1(x, x')^2.$$

Related work on deep kernel machines

Remarks on the composition of feature spaces

- we can iterate the process many times.
- the idea appears early in the literature of kernel methods [see Schölkopf et al., 1998, for a multilayer variant of kernel PCA].

Is this idea sufficient to make kernel methods more powerful?

Related work on deep kernel machines

Remarks on the composition of feature spaces

- we can iterate the process many times.
- the idea appears early in the literature of kernel methods [see Schölkopf et al., 1998, for a multilayer variant of kernel PCA].

Is this idea sufficient to make kernel methods more powerful?

Probably not:

- K_2 is doomed to be a simple kernel (dot-product or RBF kernel).
- K_3 and K_1 operate **on the same type of object**; it is not clear why desining K_3 is easier than designing K_1 directly.

Related work on deep kernel machines

Remarks on the composition of feature spaces

- we can iterate the process many times.
- the idea appears early in the literature of kernel methods [see Schölkopf et al., 1998, for a multilayer variant of kernel PCA].

Is this idea sufficient to make kernel methods more powerful?

Probably not:

- K_2 is doomed to be a simple kernel (dot-product or RBF kernel).
- K_3 and K_1 operate **on the same type of object**; it is not clear why desining K_3 is easier than designing K_1 directly.

CKNs rely on this principle, but exploit the multi-scale and spatial structure of the signal to operate on more and more complex objects.

Related work on deep kernel machines: infinite NN

A large class of kernels on $\mathbb{R}^p$ may be defined as an expectation

$$K(x, y) = \mathbb{E}_w[s(w^\top x)s(w^\top y)],$$

where $s : \mathbb{R} \to \mathbb{R}$ is a nonlinear function. The encoding can be seen as a **one-layer neural network with infinite number of random weights**.

Related work on deep kernel machines: infinite NN

A large class of kernels on $\mathbb{R}^p$ may be defined as an expectation

$$K(x,y) = \mathbb{E}_w[s(w^\top x)s(w^\top y)],$$

where $s : \mathbb{R} \to \mathbb{R}$ is a nonlinear function. The encoding can be seen as a **one-layer neural network with infinite number of random weights**.

Examples

- random Fourier features

$$\kappa(x-y) = \mathbb{E}_{w\sim q(w), b\sim\mathcal{U}[0,2\pi]}\left[\sqrt{2}\cos(w^\top x + b)\sqrt{2}\cos(w^\top y + b)\right]$$

- Gaussian kernel

$$e^{-\frac{1}{2\sigma^2}\|x-y\|_2^2} \propto \mathbb{E}_w\left[e^{\frac{2}{\sigma^2}w^\top x} e^{\frac{2}{\sigma^2}w^\top y}\right] \quad \text{with} \quad w \sim \mathcal{N}(0,(\sigma^2/4)I).$$

Related work on deep kernel machines: infinite NN

Example, arc-cosine kernels

$$K(x,y) \propto \mathbb{E}_w \left[\max\left(w^\top x, 0\right)^\alpha \max\left(w^\top y, 0\right)^\alpha \right] \quad \text{with} \quad w \sim \mathcal{N}(0, I),$$

for x, y on the hyper-sphere $\mathbb{S}^{m-1}$. Interestingly, the non-linearity s are **typical ones from the neural network literature**.

- $s(u) = \max(0, u)$ (rectified linear units) leads to $K_1(x,y) = \sin(\theta) + (\pi - \theta)\cos(\theta)$ **with** $\theta = \cos^{-1}(x^\top y)$;
- $s(u) = \max(0, u)^2$ (squared rectified linear units) leads to $K_2(x,y) = 3\sin(\theta)\cos(\theta) + (\pi - \theta)(1 + 2\cos^2(\theta))$;

Remarks

- infinite neural nets were discovered by Neal, 1994; then revisited many times [Le Roux, 2007, Cho and Saul, 2009].
- the concept does not lead to more powerful kernel methods...

Related work on deep kernel machines: infinite NN

Mea culpa

The first version of CKN [Mairal et al., 2014] relied on the infinite NN point of view. That was a bad design choice.

- unlike Nyström, the kernel approximation does not provide a point in the RKHS, which is **problematic for multilayer constructions**.
- unsupervised learning led to a **costly non-convex stochastic optimization problem** (vs. simple K-means).
- the quality of the results where far from that of Nyström.

Image classification

Experiments were conducted on classical **"deep learning" datasets**, on CPUs with no model averaging and no data augmentation.

Dataset	♯ classes	im. size	n_{train}	n_{test}
CIFAR-10	10	32×32	$50\,000$	$10\,000$
SVHN	10	32×32	$604\,388$	$26\,032$

	Stoch P. [29]	MaxOut [9]	NiN [17]	DSN [15]	Gen P. [14]	SCKN (Ours)
CIFAR-10	15.13	11.68	10.41	9.69	**7.62**	10.20
SVHN	2.80	2.47	2.35	1.92	**1.69**	2.04

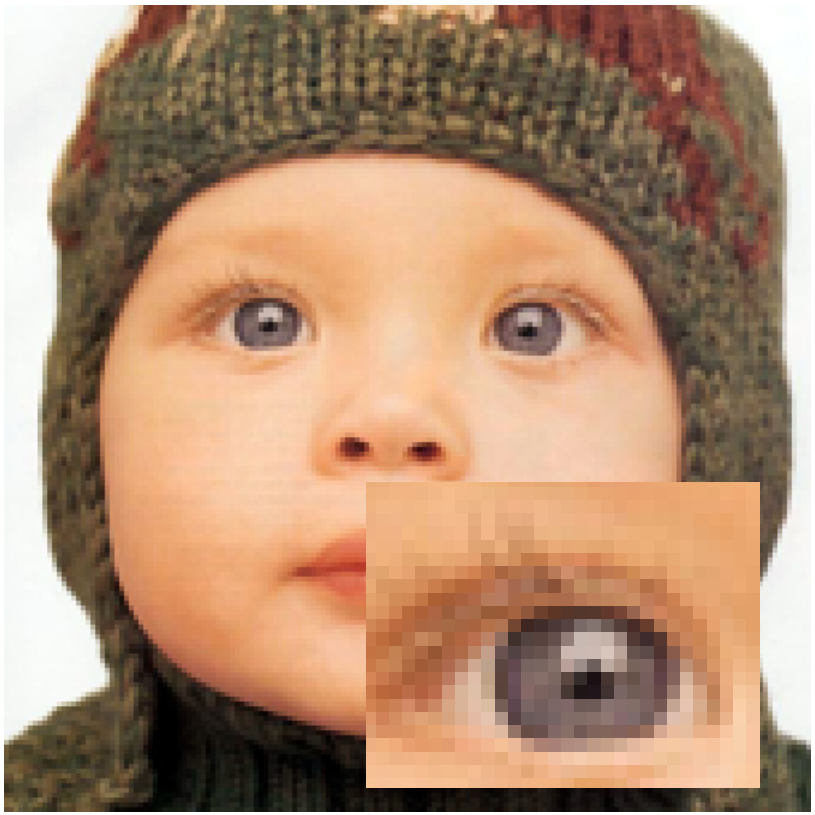
Figure: Figure from the NIPS'16 paper. Error rates in percents.

Remarks on CIFAR-10

- 10% is the standard "good" result for single model with no data augmentation.
- the best **unsupervised** architecture has two layers, is wide (1024-16384 filters), and achieves 14.2%;

Image super-resolution

The task is to predict a high-resolution y image from low-resolution one x. This may be formulated as a **multivariate regression problem**.

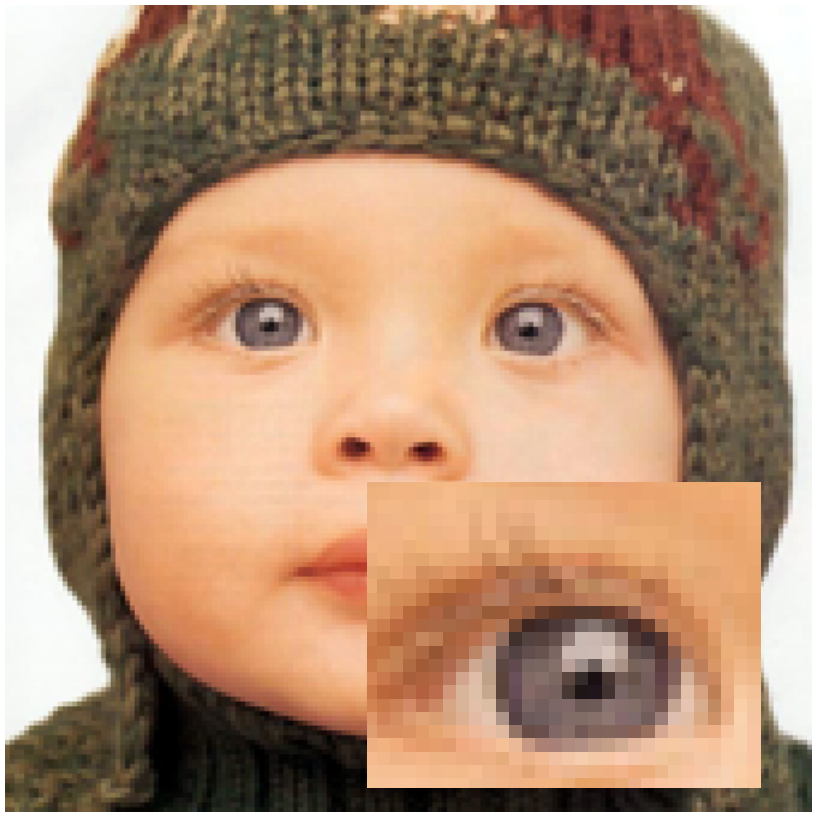


(a) Low-resolution y

(b) High-resolution x

Image super-resolution

The task is to predict a high-resolution y image from low-resolution one x. This may be formulated as a **multivariate regression problem**.



(c) Low-resolution y

(d) Bicubic interpolation

Image super-resolution

Fact.	Dataset	Bicubic	SC	CNN	CSCN	SCKN
x2	Set5	33.66	35.78	36.66	36.93	**37.07**
	Set14	30.23	31.80	32.45	32.56	**32.76**
	Kodim	30.84	32.19	32.80	32.94	**33.21**
x3	Set5	30.39	31.90	32.75	**33.10**	33.08
	Set14	27.54	28.67	29.29	29.41	**29.50**
	Kodim	28.43	29.21	29.64	29.76	**29.88**

Table: Reconstruction accuracy for super-resolution in PSNR (the higher, the better). All CNN approaches are without data augmentation at test time.

Remarks

- CNN is a "vanilla CNN" [Dong et al., 2016];
- Very recent work does better with very deep CNNs and residual learning [Kim et al., 2016];
- CSCN combines ideas from sparse coding and CNNs;

[Zeyde et al., 2010, Dong et al., 2016, Wang et al., 2015, Kim et al., 2016].

Image super-resolution

Bicubic | Sparse coding | CNN | SCKN (Ours)

Figure: Results for x3 upscaling.

Image super-resolution

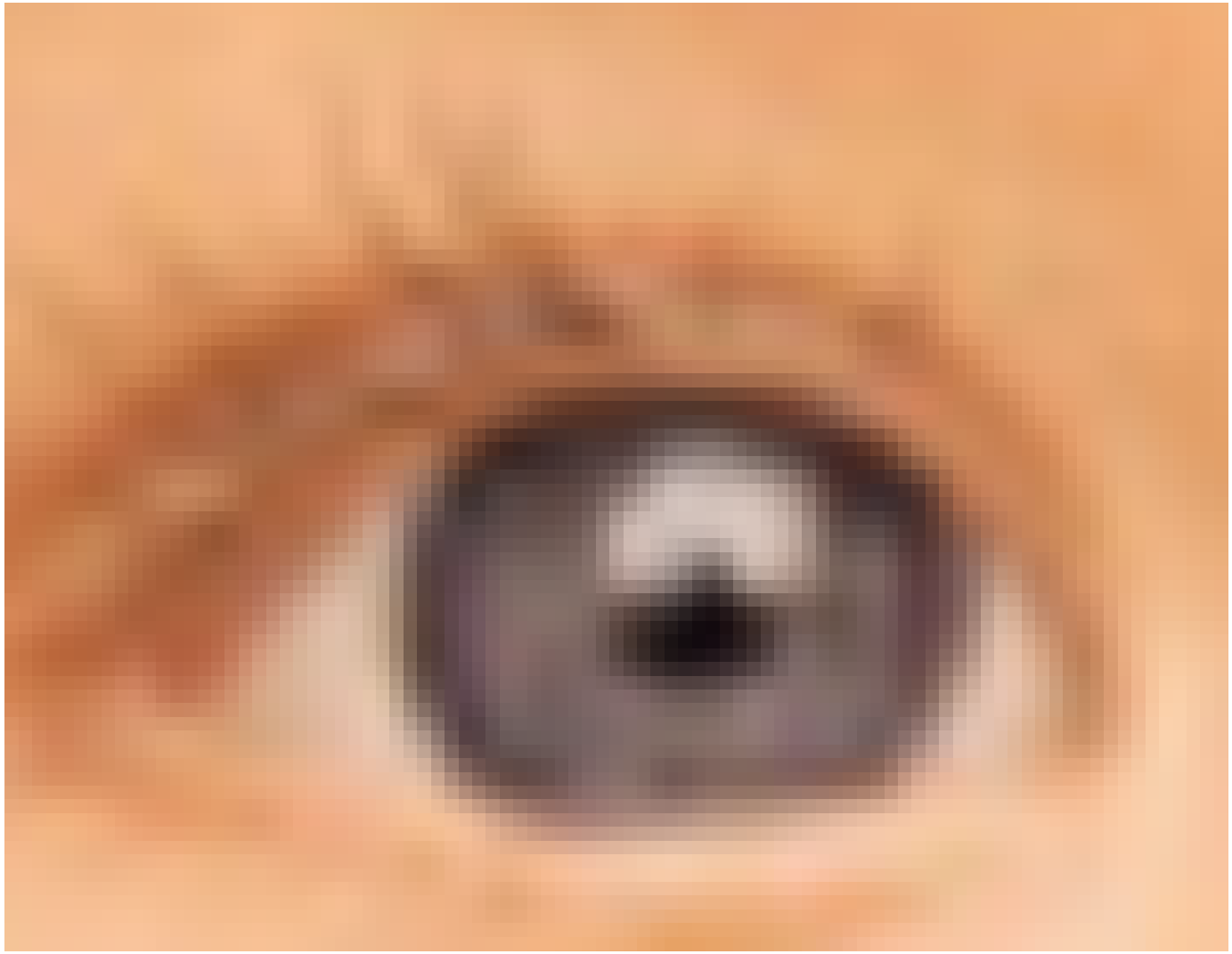


Figure: Bicubic

Image super-resolution

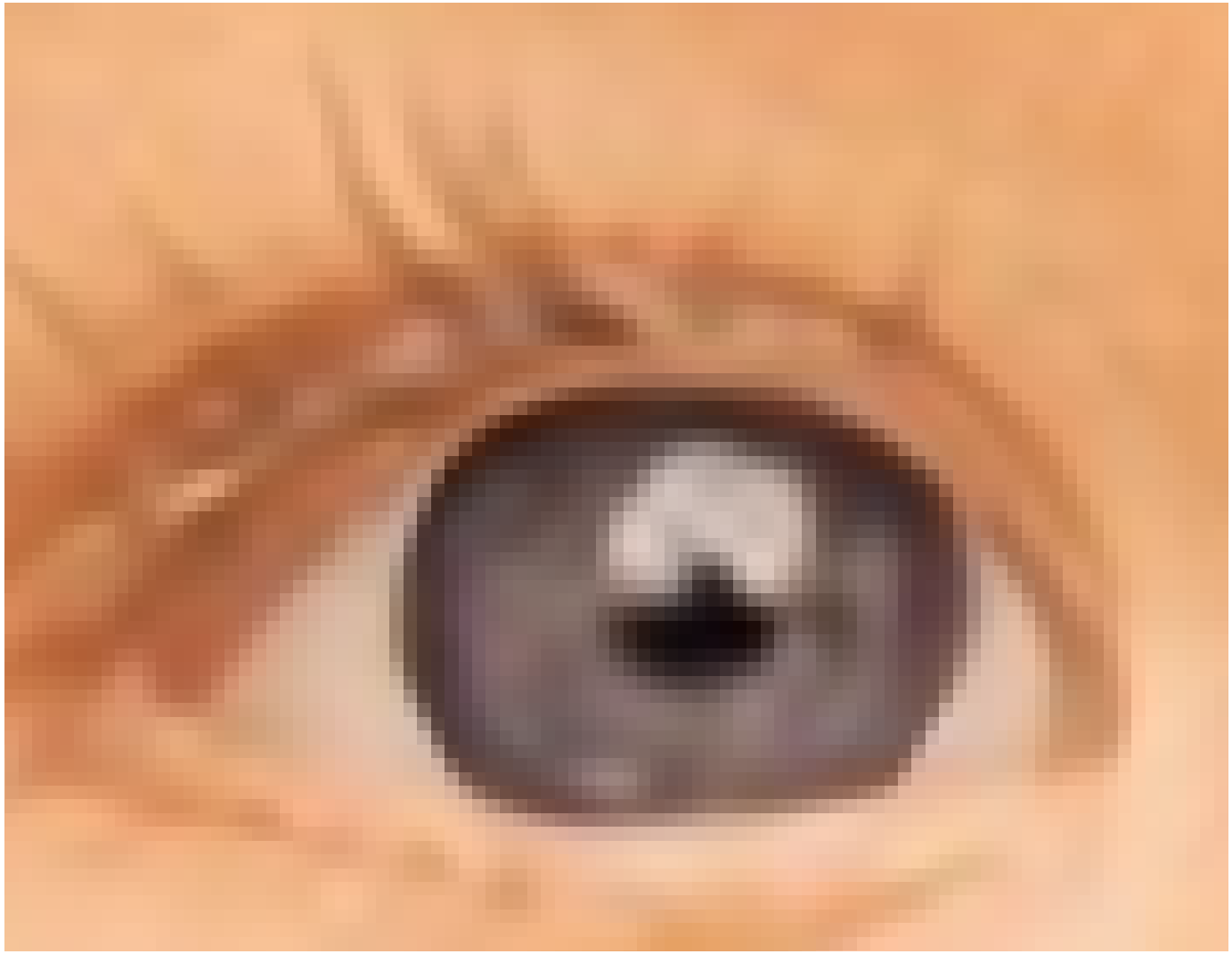
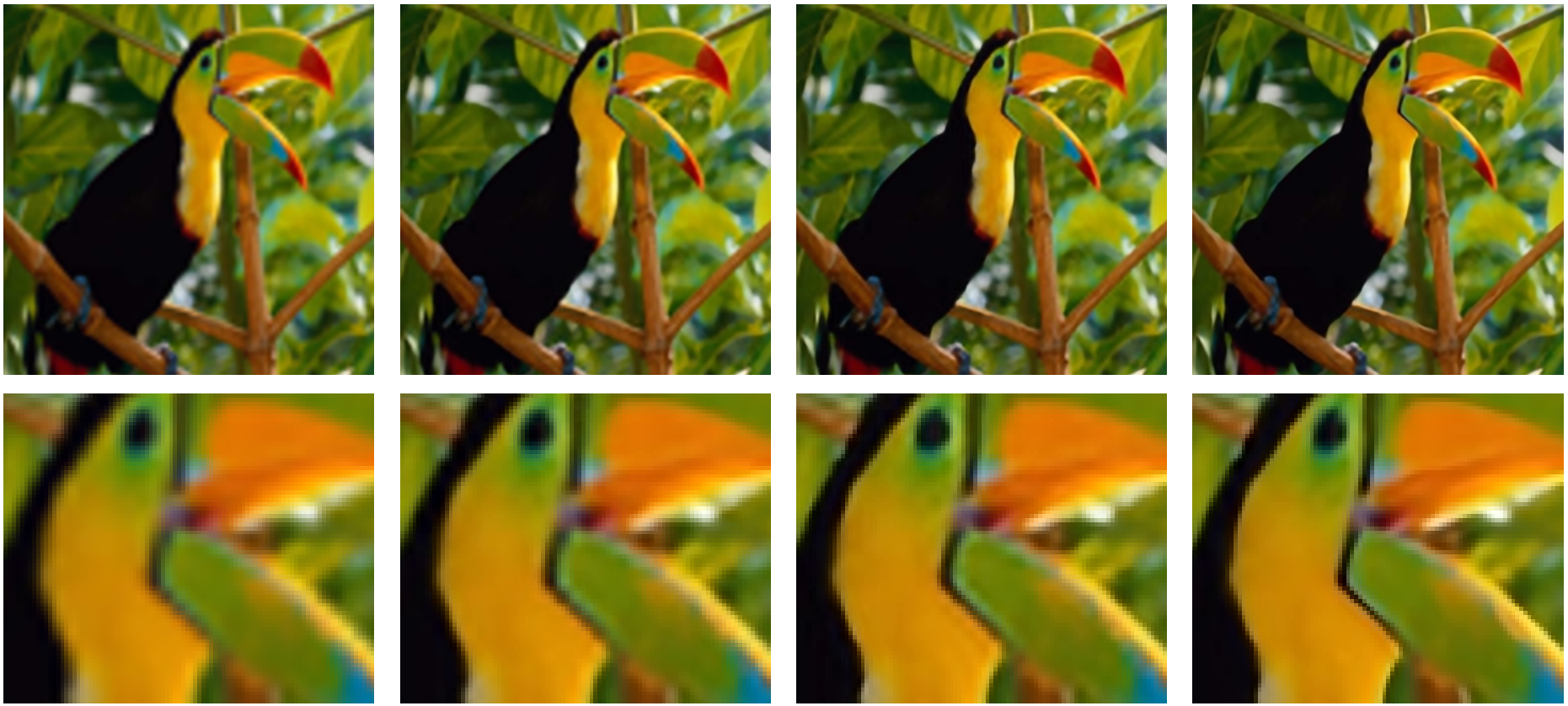


Figure: SCKN

Image super-resolution



Bicubic | Sparse coding | CNN | SCKN (Ours)

Figure: Results for x3 upscaling.

Image super-resolution

Figure: Bicubic

Image super-resolution

Figure: SCKN

Image super-resolution

Bicubic | Sparse coding | CNN | SCKN (Ours)

Figure: Results for x3 upscaling.

Image super-resolution

Figure: Bicubic

Image super-resolution

Figure: SCKN

Image super-resolution

Bicubic CNN SCKN (Ours)

Figure: Results for x3 upscaling.

Image super-resolution

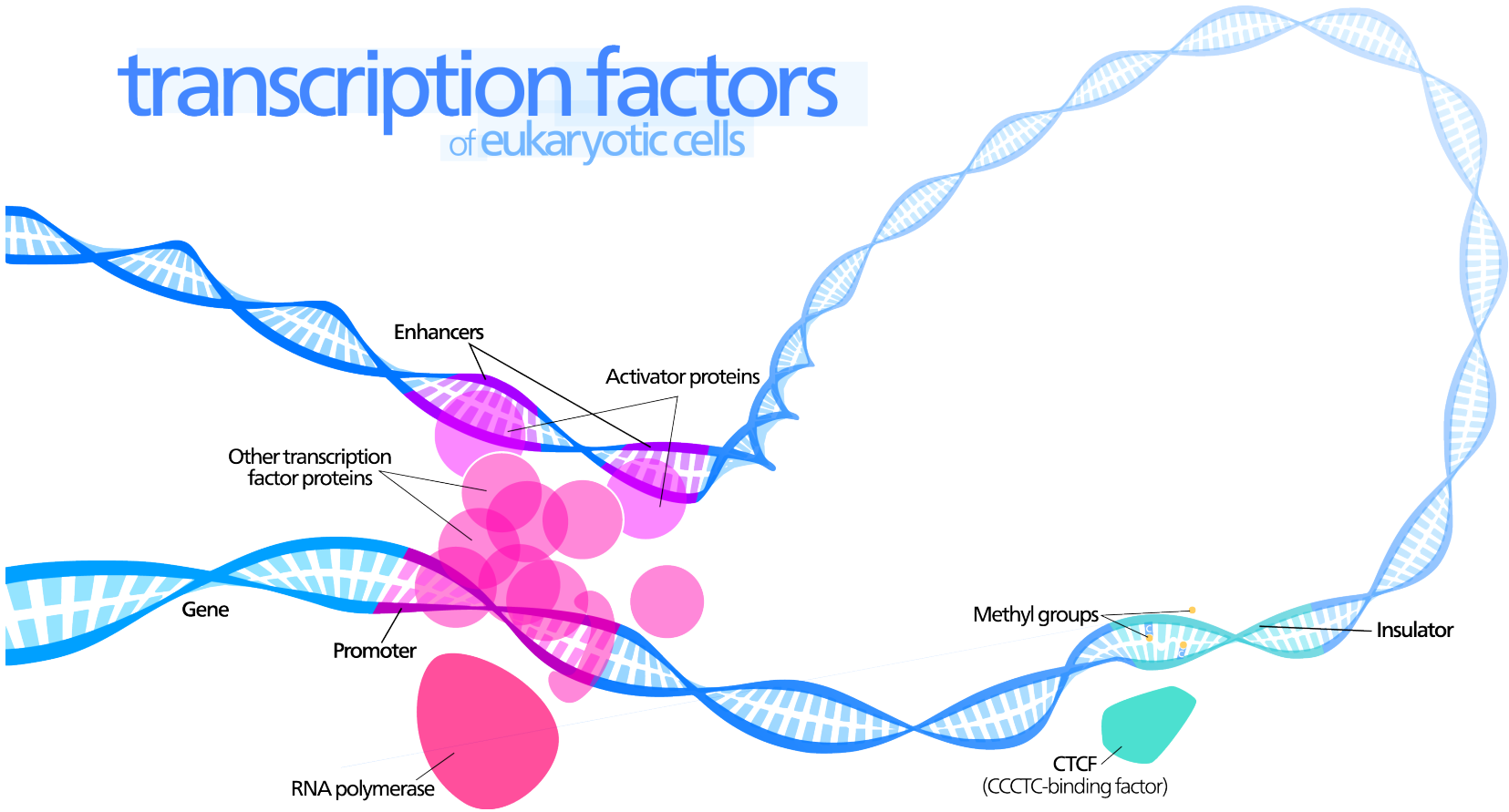
Figure: Bicubic

Image super-resolution

Figure: SCKN

Application to biological sequences

- D. Chen, L. Jacob, and J. Mairal. Predicting Transcription Factor Binding Sites with Convolutional Kernel Networks. BiorXiv. 2017.

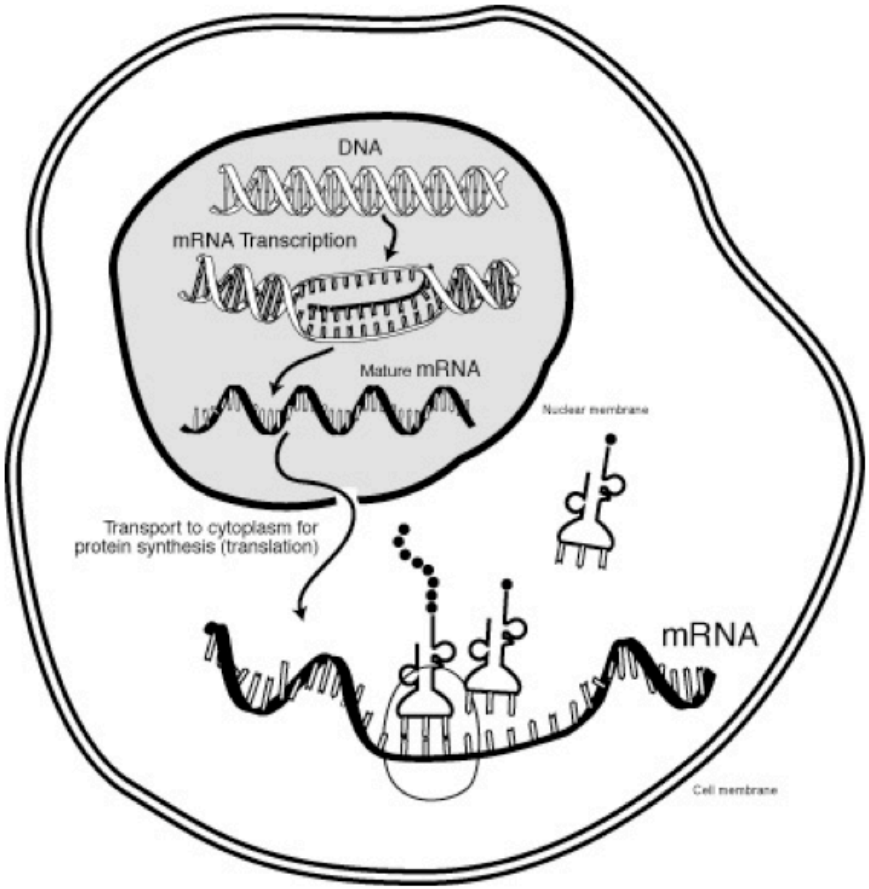


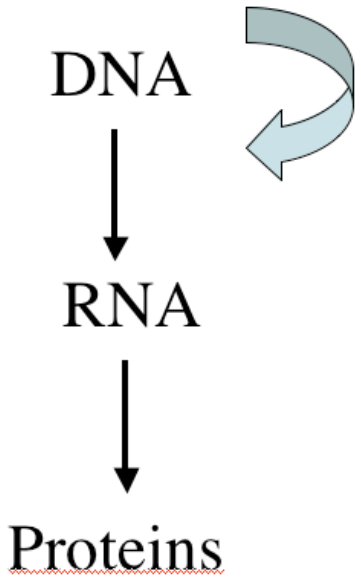
Source wikipedia.

Application to biological sequences

Transcription factors (TFs)

are proteins that bind to particular locations of the genome and regulate the rate of transcription from DNA to mRNA.





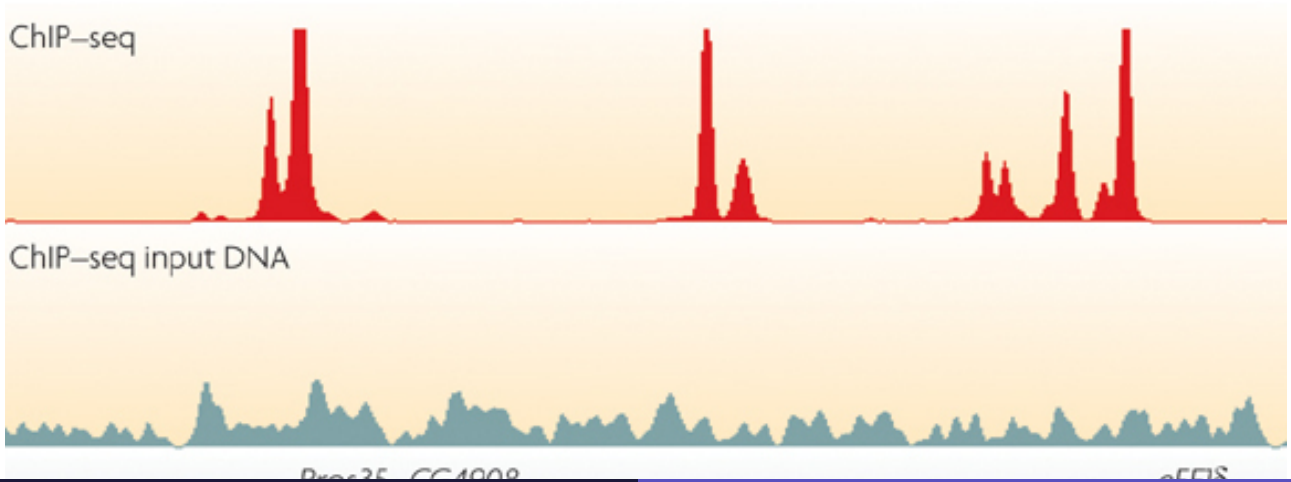
Application to biological sequences

There are more than 2 000 TFs types in humans. Studying the nature of binding sites is of utmost importance for

- clinical diagnostics;
- drug targets;
- synthetic biology;
- understanding mechanisms of evolution...

CHIP-seq technology

For a particular TFs, it is possible to extract short DNA sequences (about 100 bases) that contain a binding site.



Application to biological sequences

Data from ENCODE

composed of the peaks (sequences of size 101) of 506 experiments. For each experiment, between 1000s and 50 000s of sequences are produced.

Question

Can we design a model that predicts the position of TF binding sites, and provide an interpretation of their DNA patterns?

Ideally, the machine learning task is that of genome-wide **detection**. However, following earlier work, we consider a **classification** task.

- A dataset contains n pairs (x_i, y_i), where x_i is a DNA sequence, and y_i is label in $\{-1, +1\}$ (TF binding site or not).
- negative examples are generated by dinucleotide shuffling.

Application to biological sequences

Data from ENCODE

composed of the peaks (sequences of size 101) of 506 experiments. For each experiment, between 1000s and 50 000s of sequences are produced.

Question

Can we design a model that predicts the position of TF binding sites, and provide an interpretation of their DNA patterns?

Ideally, the machine learning task is that of genome-wide **detection**. However, following earlier work, we consider a **classification** task.

- A dataset contains n pairs (x_i, y_i), where x_i is a DNA sequence, and y_i is label in $\{-1, +1\}$ (TF binding site or not).
- negative examples are generated by dinucleotide shuffling.

All of this is a **simple proxy** for the "true" detection task.

Application to biological sequences

But then, we end up with a classical supervised learning formulation.

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(f)}_{\text{regularization}}$$

Challenges

- achieve **good prediction** while being **interpretable**.
- learn **without user interaction** (no manual parameter tuning). (remember we need to process 500 datasets).
- exploit together datasets from the same TF (**multitask learning?**).

Convolutional neural networks for biological sequences

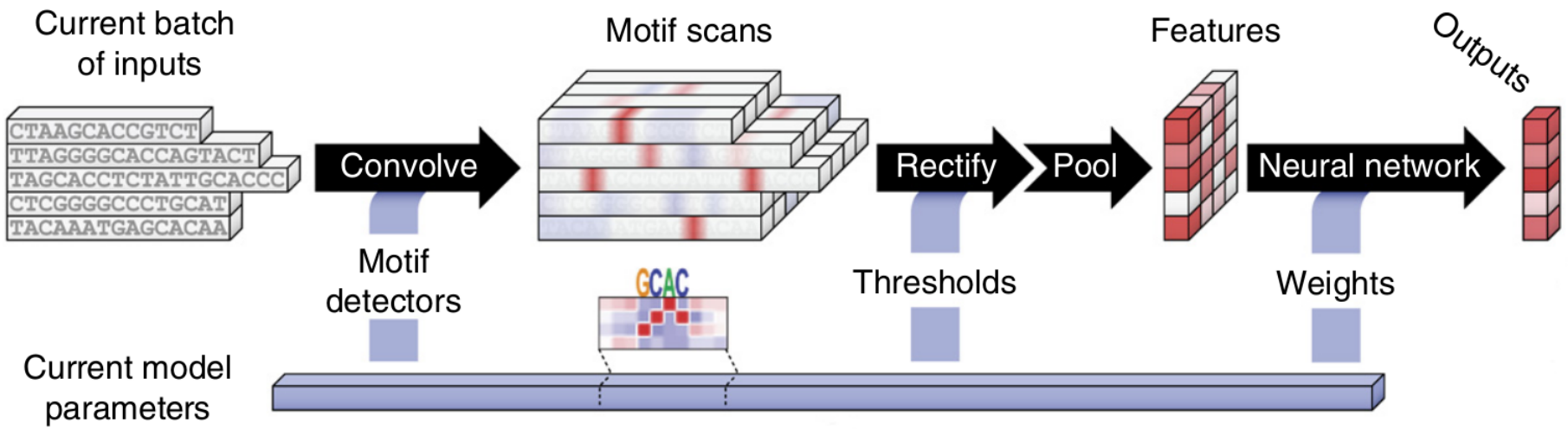


Figure: DeepBind model architecture from Alipanahi et al. [2015]

- DeepBind: a "one-layer" convolutional neural network architecture, **state of the art** model for TF binding prediction problem.
- an **embedding** layer is used to encode sequence x from $\mathcal{X}$ to $\mathbb{R}^{4\times m}$ where m is the sequence length.

Representation of sequence motifs

Sequence logo

A sequence logo is a representation of the relative frequency, at each position, of each letter, in a collection of aligned sequences.

Figure: An example of sequence logo for the LexA-binding motif

Convolutional neural networks for biological sequences

Figure: Filter visualizations for DeepBind [Angermueller et al., 2016]

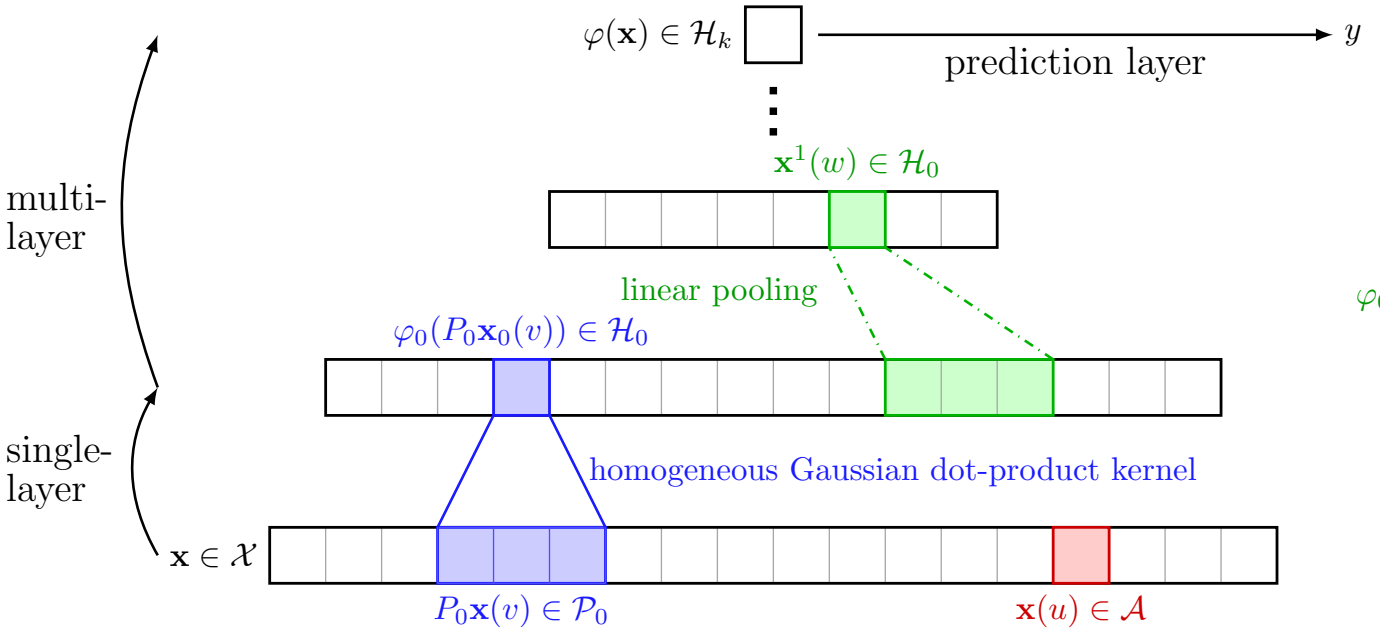
- The filters in the convolutional layer need to be interpreted.
- sequence logos can be generated by alignment of filter to the top activated subsequences from a set of test sequences.

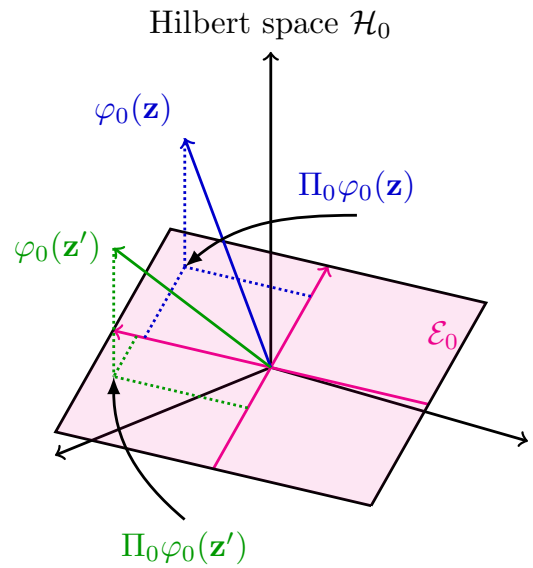
Convolutional neural networks for biological sequences

Remarks

- CNNs are hard to regularize and require Dropout and weight decays.
- initialization requires also parameters.
- To remove the need of manual hyper-parameter tuning, DeepBind uses random search to **find a set of parameters per dataset**. This requires to learn ≈ 100 models per dataset.
- **Can we interpret the filters without using test sequences?**

Convolutional kernel networks for biological sequences





Also use invariance to reverse complementation of the DNA sequence.

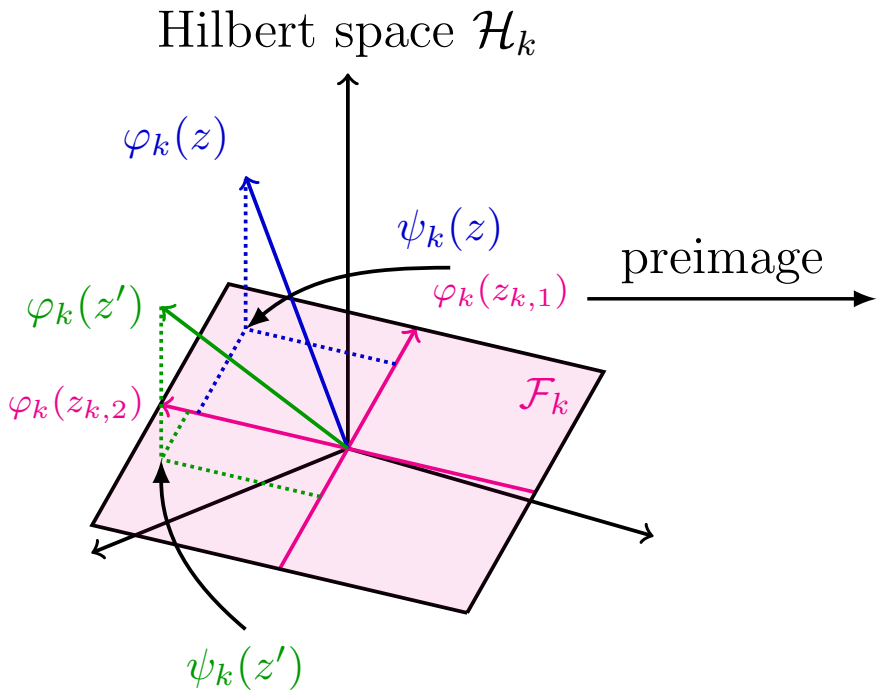
$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, \max(f(x_i), f(x_i^c))}_{\text{empirical risk, data fit}} + \underbrace{\lambda\Omega(f)}_{\text{regularization}}$$

Convolutional kernel networks for biological sequences

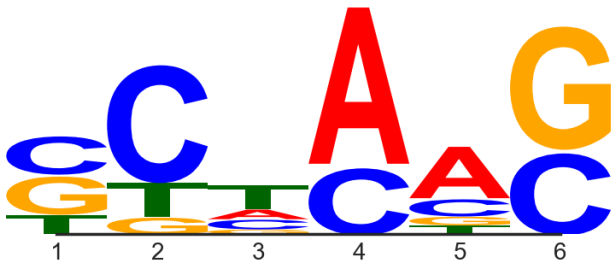
Remarks

- same set of parameters for all 507 datasets.
- initialization via unsupervised learning (no parameters).
- interpretation of the motifs via pre-image problem.

$$\min_{\mathbf{y} \in \mathcal{M}} \|\varphi_k(P_k \mathbf{y}_{k-1}) - \varphi_k(z_{k,i})\|^2_{\mathcal{H}_k},$$



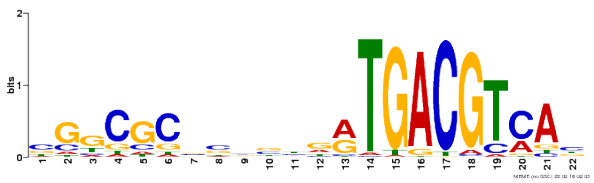
preimage



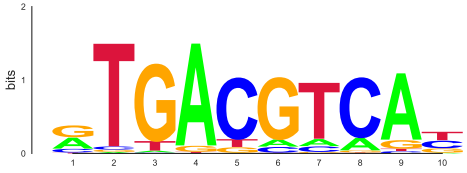
A	0	0	0.2	0.7	0.6	0
C	0.4	0.7	0.2	0.3	0.2	0.4
G	0.4	0.1	0.1	0	0.1	0.6
T	0.2	0.2	0.5	0	0.1	0

motif associated with $\varphi_k(z_{k,1})$

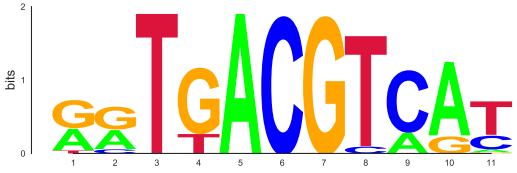
Motifs in convolutional kernel networks



(a) Factorbook

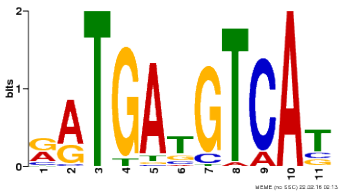


(b) DeepBind

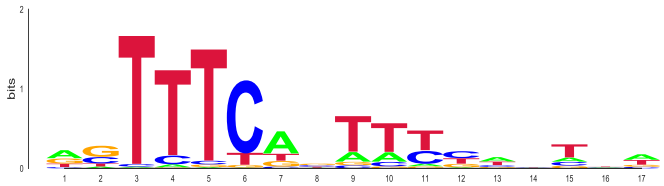


(c) CKN

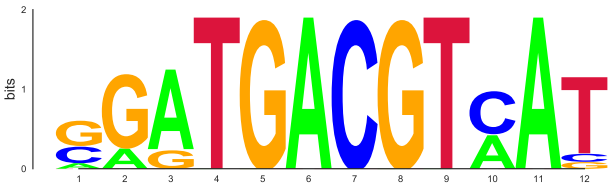
Figure: ATF1 K562

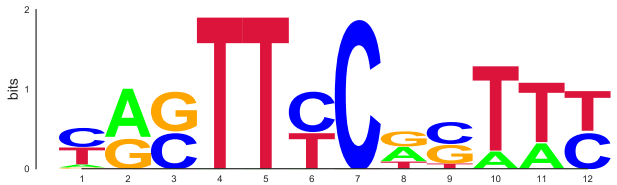


(a) Factorbook



(b) DeepBind





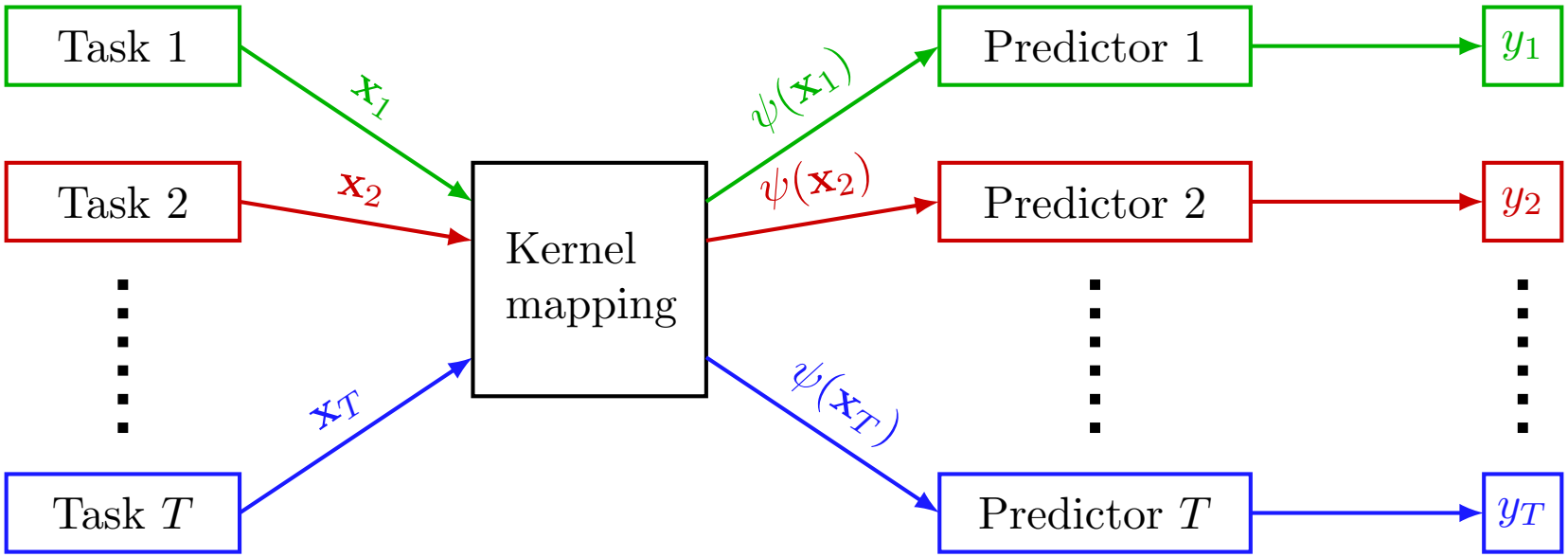
(c) CKN

Figure: ATF2 GM12878

Convolutional kernel networks for biological sequences

With multitask learning

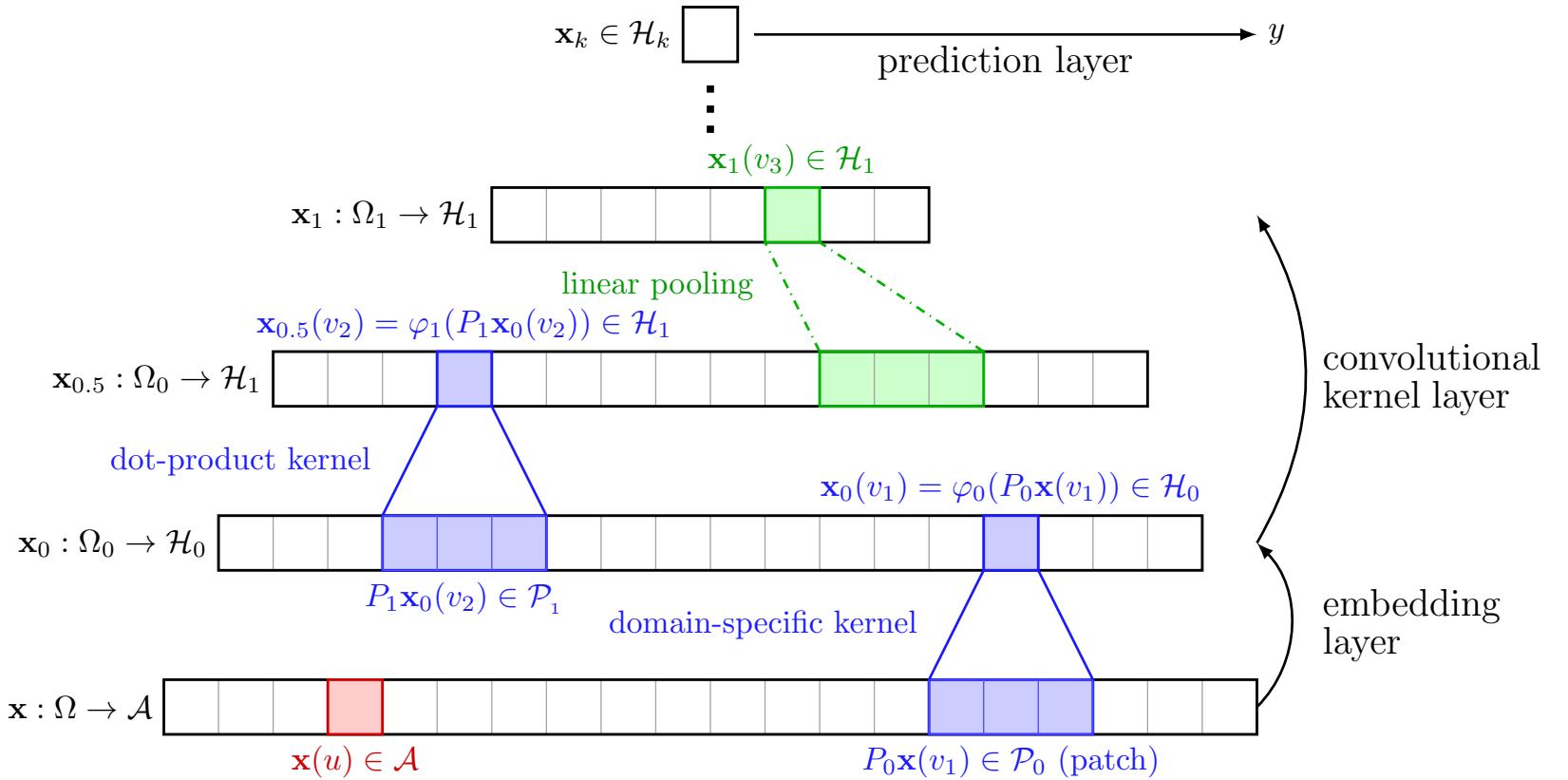
$$\min_{w_1,\ldots,w_T \in \mathbb{R}^{p_k}, \mathbf{Z}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n_t}\sum_{i=1}^{n_t} L(y_{i,t}, \langle w_t, \psi(x_{i,t})\rangle) + \frac{\lambda}{2}\|w_t\|^2,$$



What about classical kernel methods?

Classical kernels for biological sequences [see Ben-Hur et al., 2008], typically encode **biological phenomenons** (insertions, deletions, ...).

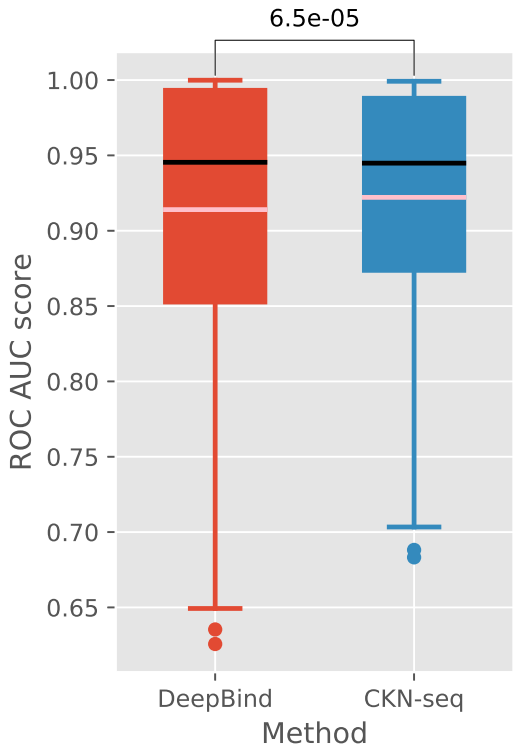
They are less fashionable these days due to their lack of stability, but can we still use some of them in the embedding layer?

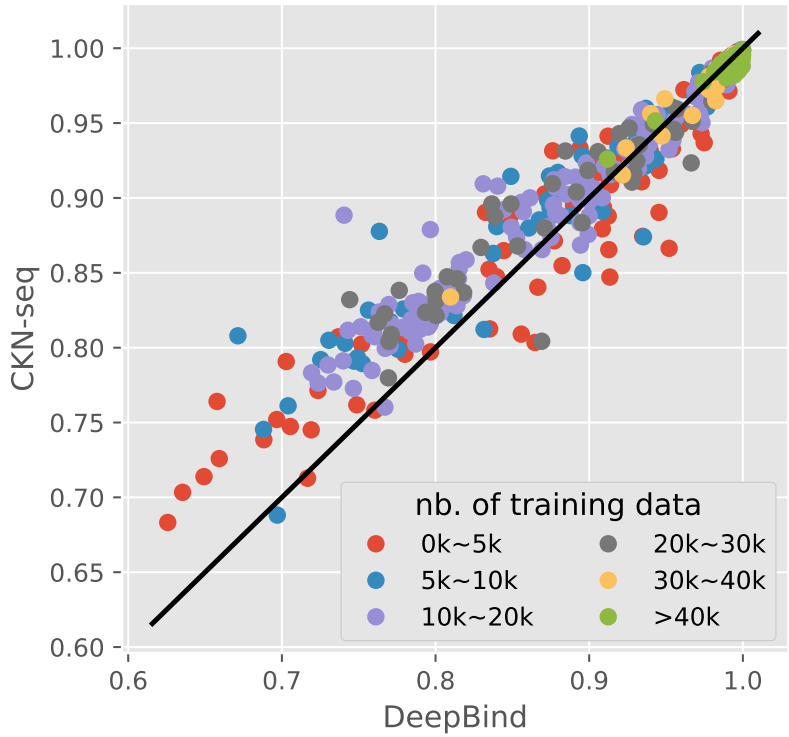


CKN-seq vs DeepBind

	DeepBind	CKN-seq
Training time (min)	72.0 ± 1.0	2.9 ± 2.3

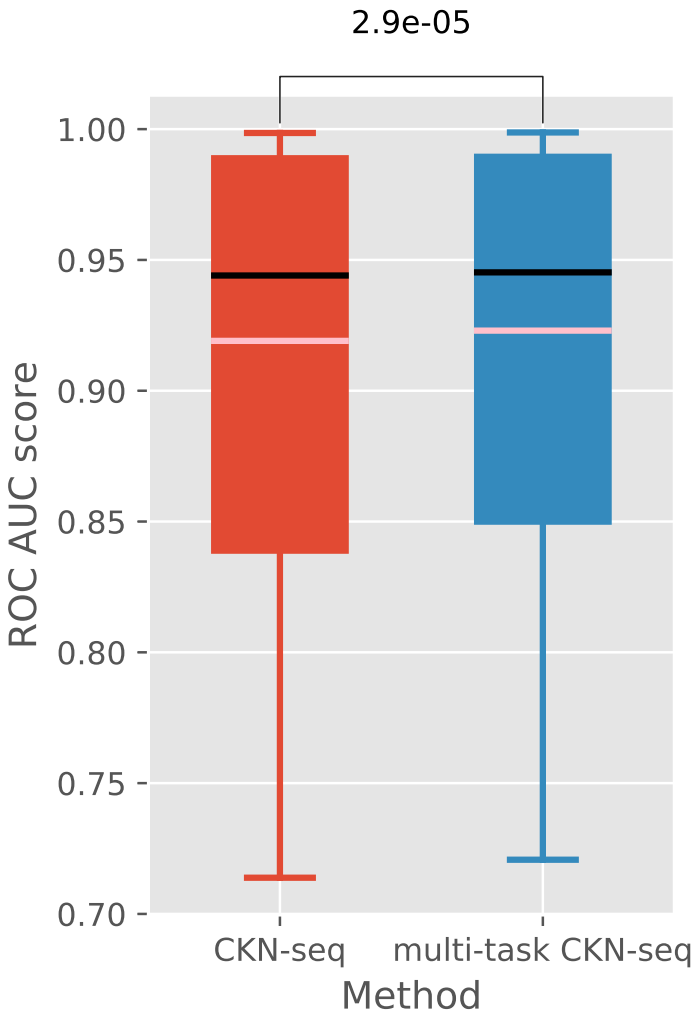
Table: Average training time on 40 different experiments

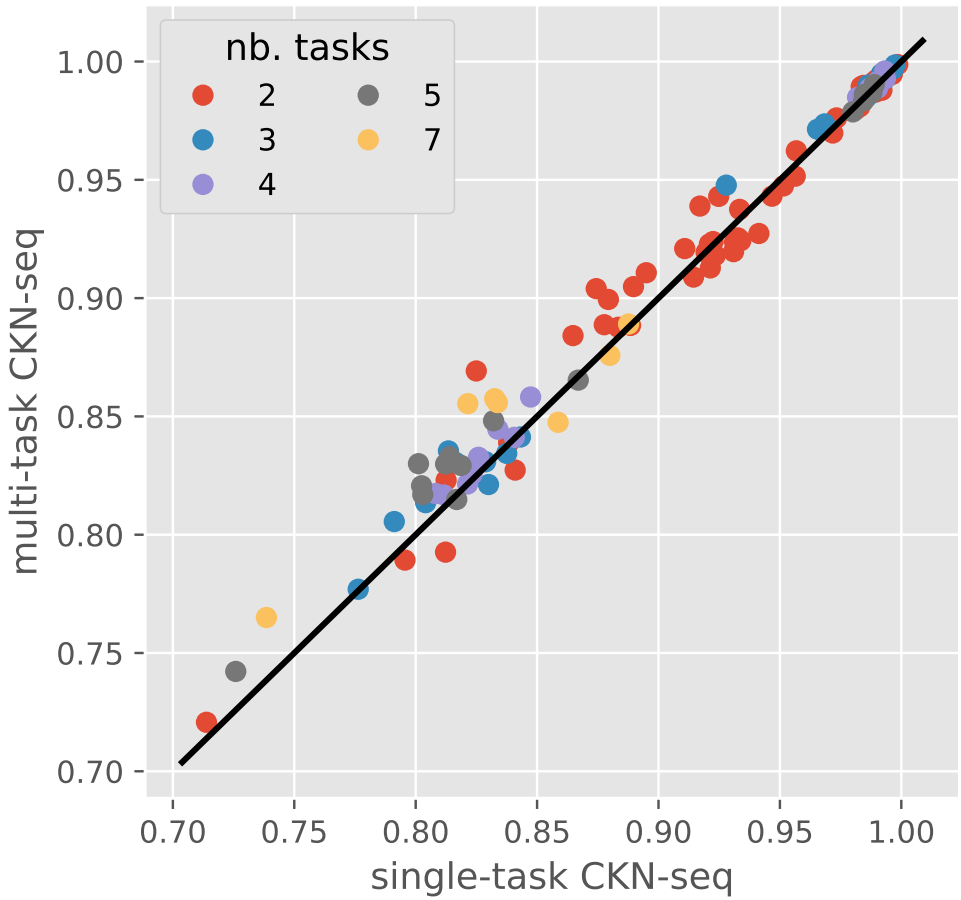




Benefits of multitask learning

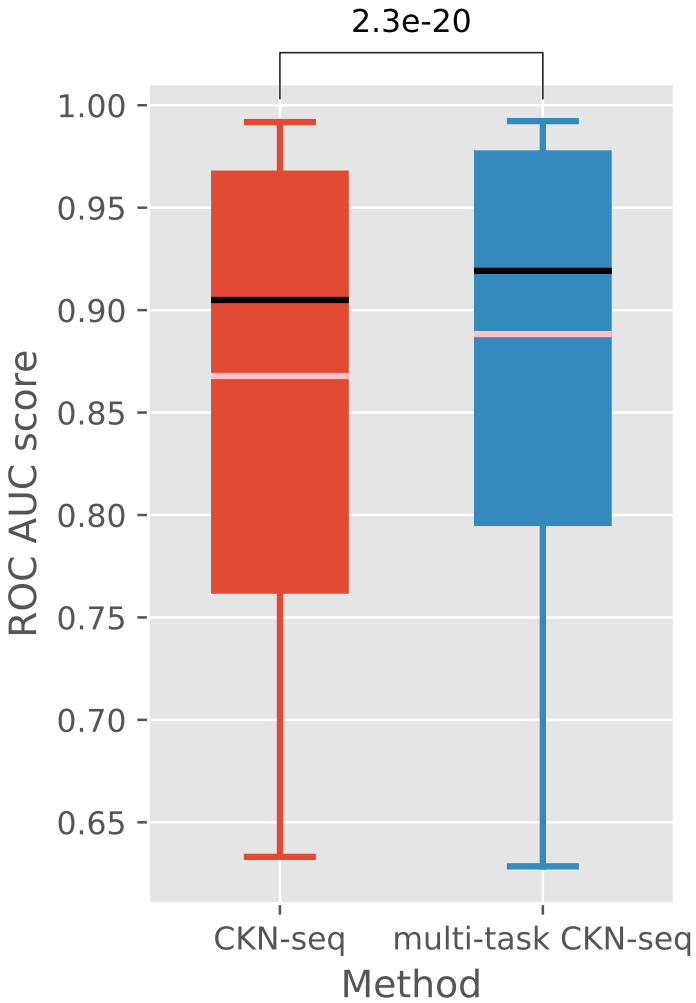
With all data available

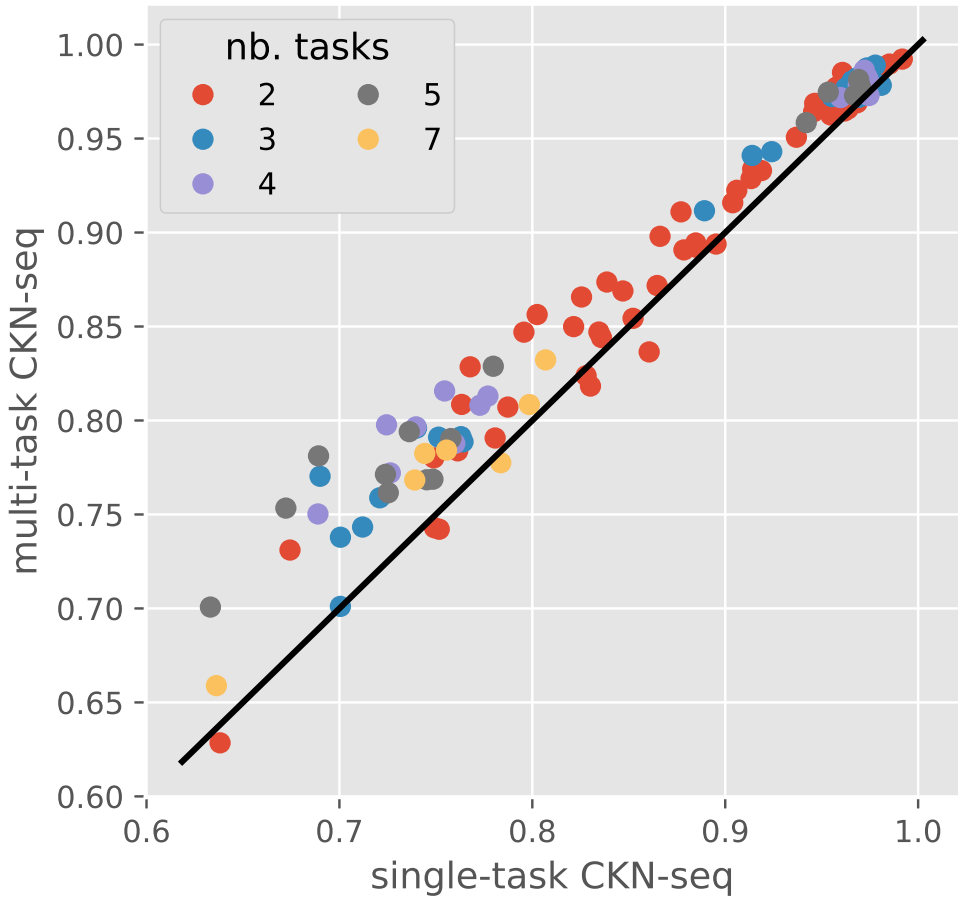




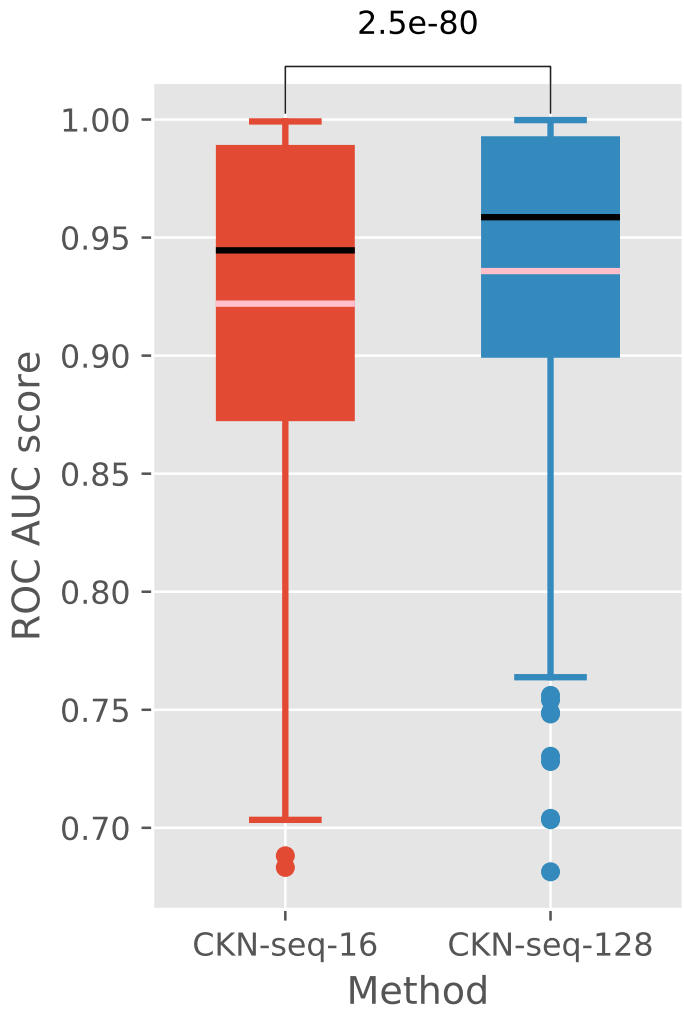
Benefits of multitask learning

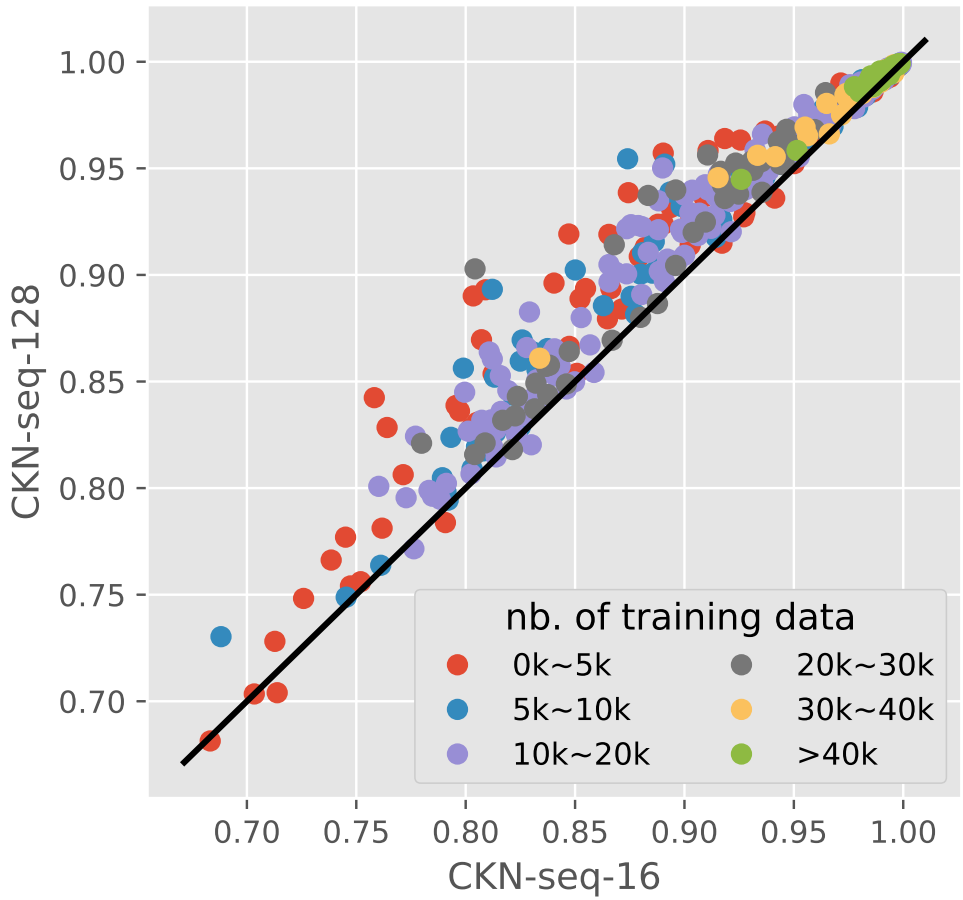
When using fewer training data ($n = 1000$).



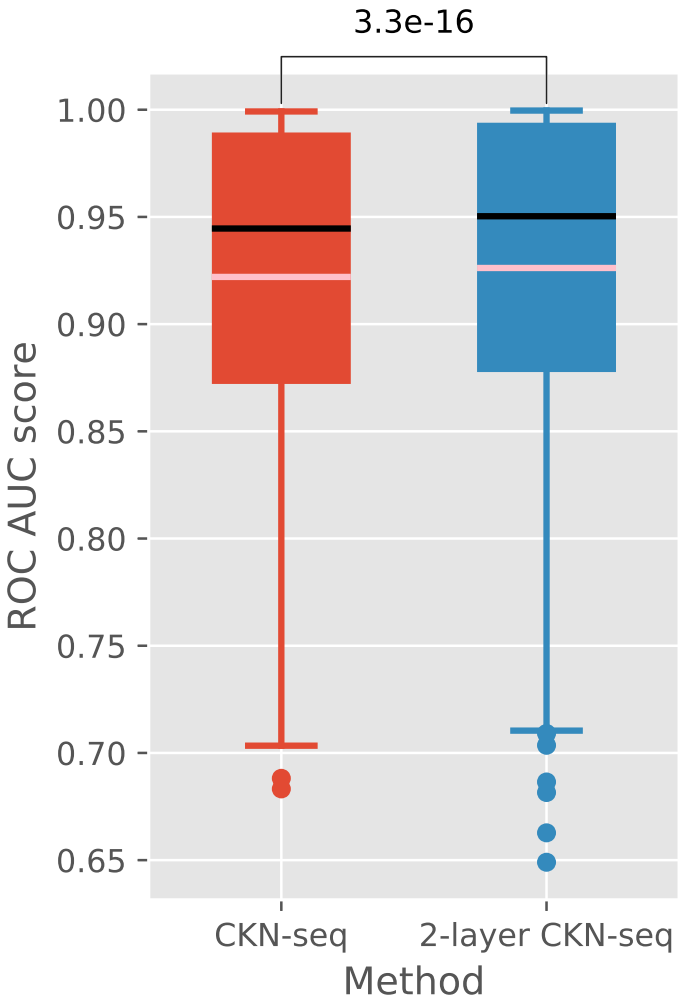


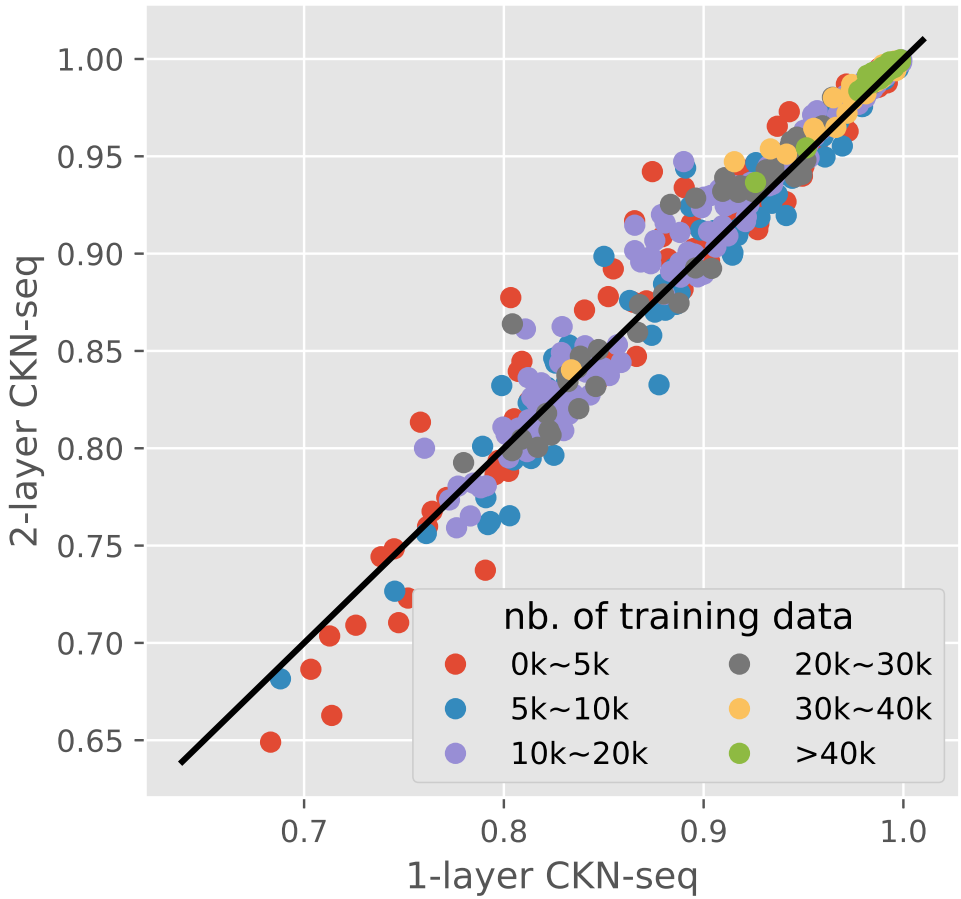
Influence of the number of filters





Multilayer CKN-seq

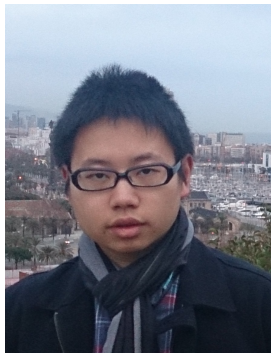




Conclusion of this part

On-going work, but code is available to play with these models.

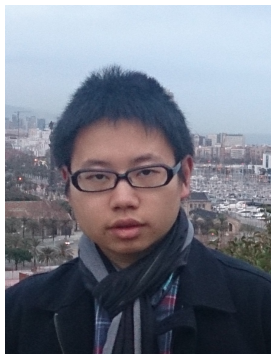
- first version of unsupervised CKN, slow, Matlab.
  `http://ckn.gforge.inria.fr/` (do not use this one).
- second version of CKN, unsupervised or supervised. TensorFlow.
  `https://gitlab.inria.fr/thoth/ckn` by Ghislain Durif.
- CKN for biological sequences. PyTorch.
  `https://gitlab.inria.fr/dchen/CKN-seq` by Dexiong Chen.



Conclusion of this part

On-going work, but code is available to play with these models.

- first version of unsupervised CKN, slow, Matlab.
  `http://ckn.gforge.inria.fr/` (do not use this one).
- second version of CKN, unsupervised or supervised. TensorFlow.
  `https://gitlab.inria.fr/thoth/ckn` by Ghislain Durif.
- CKN for biological sequences. PyTorch.
  `https://gitlab.inria.fr/dchen/CKN-seq` by Dexiong Chen.



We have seen the practice. **What about the theory?**

Part III: Invariance, Stability, and Complexity of Deep Convolutional Representations

Understanding deep convolutional representations

Questions

- Are they **stable to deformations?**
- How can we achieve **invariance to transformation groups?**
- Do they **preserve signal information?**
- How can we measure **model complexity?**

- A. Bietti and J. Mairal. Group Invariance, Stability to Deformations, and Complexity of Deep Convolutional Representations. arXiv:1706.03078. 2017.
- A. Bietti and J. Mairal. Invariance and Stability of Deep Convolutional Representations. NIPS. 2017.

Construct a functional space for deep learning

Main ideas

1. use the kernel construction of CKNs;
2. notice that the functional space contains some CNNs;
3. derive theoretical results for CKNs and CNNs.

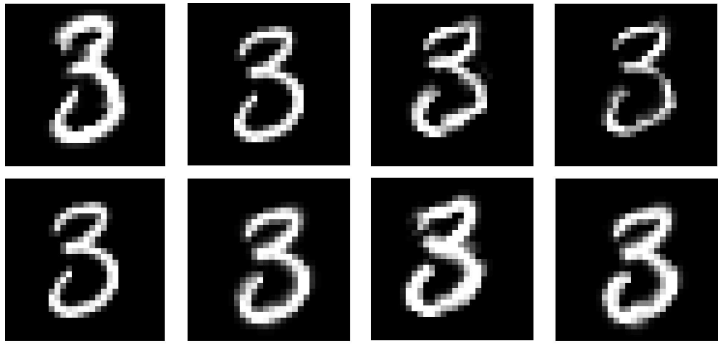
Why? Separate learning from representation: $f(x) = \langle f, \Phi(x) \rangle$

- $\Phi(x)$: CNN **architecture** (stability, invariance, signal preservation)
- f: CNN **model**, learning, generalization through $\|f\|$

$$|f(x) - f(x')| \leq \|f\| \cdot \|\Phi(x) - \Phi(x')\|.$$

- $\|f\|$ **controls both stability and generalization**!
  $\rightarrow$ discriminating small deformations requires large $\|f\|$
  $\rightarrow$ learning stable functions is "easier"

Property 1: Stability to deformations



- Go beyond simple translation invariance;
- Small local deformations do not change content of images ("label");
- Formally studied for wavelet-based scattering transform [Mallat, 2012, Bruna and Mallat, 2013];
- Can we do the same for CKNs and CNNs?

Property 2: Group invariance

- Convolutions and pooling provides translation invariance
- Can we encode more general **transformation groups** in the architecture? (e.g. rotations, roto-translations, rigid motion)
- How does this relate to stability?
- Related work: [Cohen and Welling, 2016, Mallat, 2012, Sifre and Mallat, 2013, Raj et al., 2016]

Property 3: Signal preservation

- Do deep convolutional representations **preserve signal information**?
- Can x be recovered from $\Phi(x)$?
- At odds with invariance and stability?

Property 4: Model complexity and generalization

- How do we measure **model complexity** of CKNs and CNNs?
- Can we get meaningful bounds on generalization error?
- Summary of results:
  - Some CNNs are contained in the RKHS of CKNs.
  - we may control the RKHS norm of a generic CNN
  - The choice of activation function is important.
  - Same norm also controls stability ("stable functions generalize better")
- Related work: [Zhang et al., 2017]

Property 4: Model complexity and generalization

- How do we measure **model complexity** of CKNs and CNNs?
- Can we get meaningful bounds on generalization error?
- Summary of results:
  - Some CNNs are contained in the RKHS of CKNs.
  - we may control the RKHS norm of a generic CNN
  - The choice of activation function is important.
  - Same norm also controls stability ("stable functions generalize better")
- Related work: [Zhang et al., 2017]

Spoiler: classical CNNs should be regularized with products of spectral norms [Bartlett et al., 2017]?

Property 4: Model complexity and generalization

- How do we measure **model complexity** of CKNs and CNNs?
- Can we get meaningful bounds on generalization error?
- Summary of results:
  - Some CNNs are contained in the RKHS of CKNs.
  - we may control the RKHS norm of a generic CNN
  - The choice of activation function is important.
  - Same norm also controls stability ("stable functions generalize better")
- Related work: [Zhang et al., 2017]

Spoiler: classical CNNs should be regularized with products of spectral norms [Bartlett et al., 2017]?

CKNs should be regularized with the ℓ_2-norm of the last layer.

A generic deep convolutional representation

We adopt a formalism for **continuous signals**.

- $x_0 : \Omega \to \mathcal{H}_0$: initial (**continuous**) signal
  - $u \in \Omega = \mathbb{R}^d$: location ($d = 2$ for images)
  - $x_0(u) \in \mathcal{H}_0$: value ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images)

A generic deep convolutional representation

We adopt a formalism for **continuous signals**.

- $x_0 : \Omega \to \mathcal{H}_0$: initial (**continuous**) signal
  - $u \in \Omega = \mathbb{R}^d$: location ($d = 2$ for images)
  - $x_0(u) \in \mathcal{H}_0$: value ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images)
- $x_k : \Omega \to \mathcal{H}_k$: *feature map* at layer k

$$P_k x_{k-1}$$

  - P_k: **patch extraction** operator, extract small patch of feature map x_{k-1} around each point u

A generic deep convolutional representation

We adopt a formalism for **continuous signals**.

- $x_0 : \Omega \to \mathcal{H}_0$: initial (**continuous**) signal
  - $u \in \Omega = \mathbb{R}^d$: location ($d = 2$ for images)
  - $x_0(u) \in \mathcal{H}_0$: value ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images)
- $x_k : \Omega \to \mathcal{H}_k$: *feature map* at layer k

$$M_k P_k x_{k-1}$$

  - P_k: **patch extraction** operator, extract small patch of feature map x_{k-1} around each point u
  - M_k: **non-linear mapping** operator, maps each patch to a new point with a **pointwise** non-linear function $\varphi_k(\cdot)$

A generic deep convolutional representation

We adopt a formalism for **continuous signals**.

- $x_0 : \Omega \to \mathcal{H}_0$: initial (**continuous**) signal
  - $u \in \Omega = \mathbb{R}^d$: location ($d = 2$ for images)
  - $x_0(u) \in \mathcal{H}_0$: value ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images)
- $x_k : \Omega \to \mathcal{H}_k$: *feature map* at layer k

$$x_k = A_k M_k P_k x_{k-1}$$

- P_k: **patch extraction** operator, extract small patch of feature map x_{k-1} around each point u
- M_k: **non-linear mapping** operator, maps each patch to a new point with a **pointwise** non-linear function $\varphi_k(\cdot)$
- A_k: (linear, Gaussian) **pooling** operator at scale σ_k

A generic deep convolutional representation

We adopt a formalism for **continuous signals**.

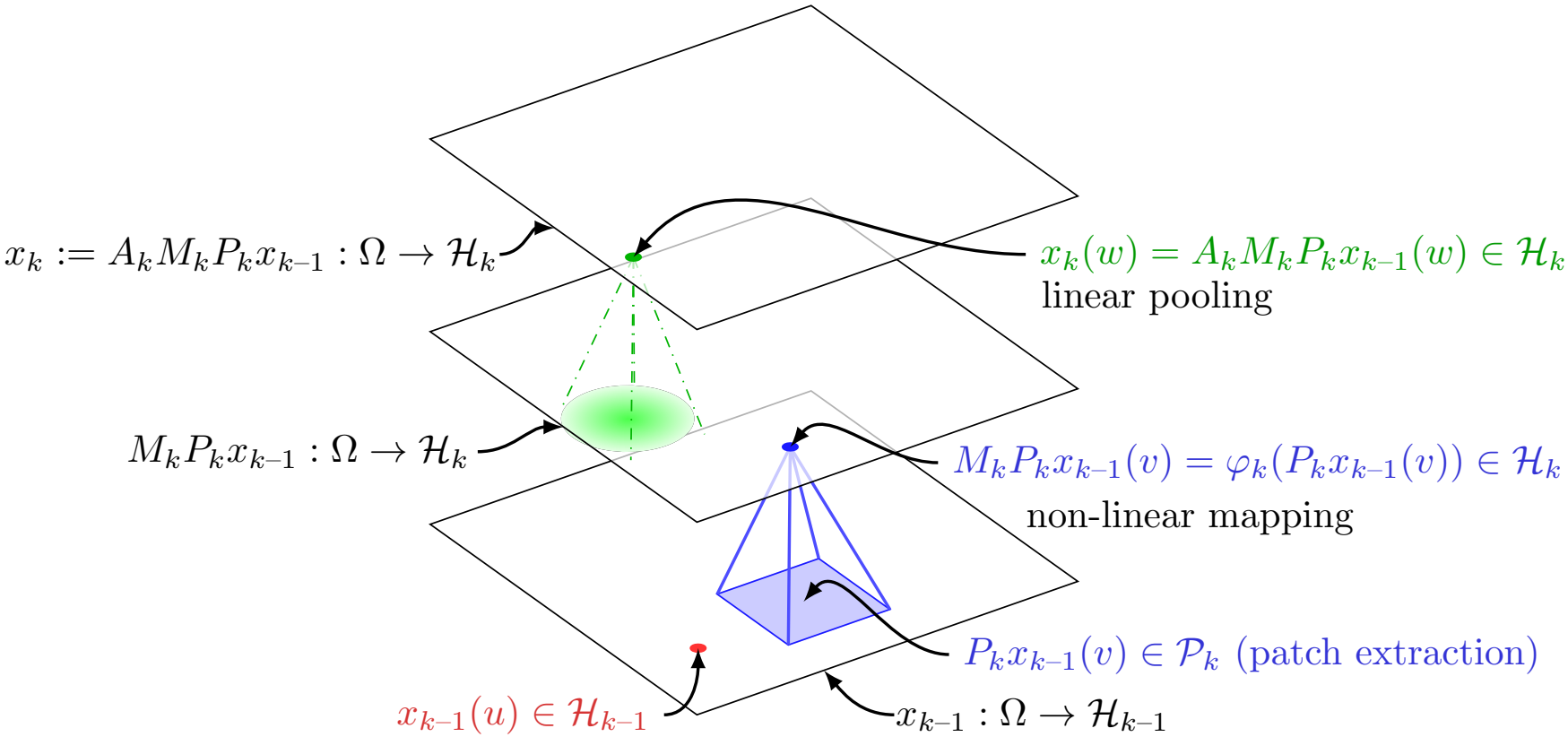
- $x_0 : \Omega \to \mathcal{H}_0$: initial (**continuous**) signal
  - $u \in \Omega = \mathbb{R}^d$: location ($d = 2$ for images)
  - $x_0(u) \in \mathcal{H}_0$: value ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images)
- $x_k : \Omega \to \mathcal{H}_k$: *feature map* at layer k

$$x_k = A_k M_k P_k x_{k-1}$$

  - P_k: **patch extraction** operator, extract small patch of feature map x_{k-1} around each point u
  - M_k: **non-linear mapping** operator, maps each patch to a new point with a **pointwise** non-linear function $\varphi_k(\cdot)$
  - A_k: (linear, Gaussian) **pooling** operator at scale σ_k

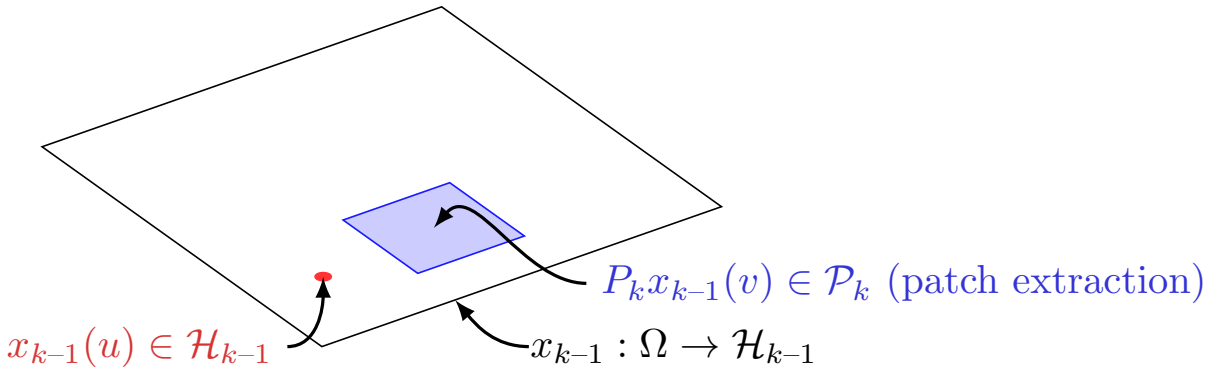
In this part, a signal x is always in $L^2(\Omega, \mathcal{H})$ for some Hilbert space $\mathcal{H}$; in other words $\|x\|_{L^2}^2 = \int_\Omega \|x(u)\|_{\mathcal{H}}^2 du < \infty$.

A generic deep convolutional representation



Patch extraction operator P_k

$$P_k x_{k-1}(u) := (v \mapsto x_{k-1}(u+v))_{v \in S_k} \in \mathcal{P}_k$$



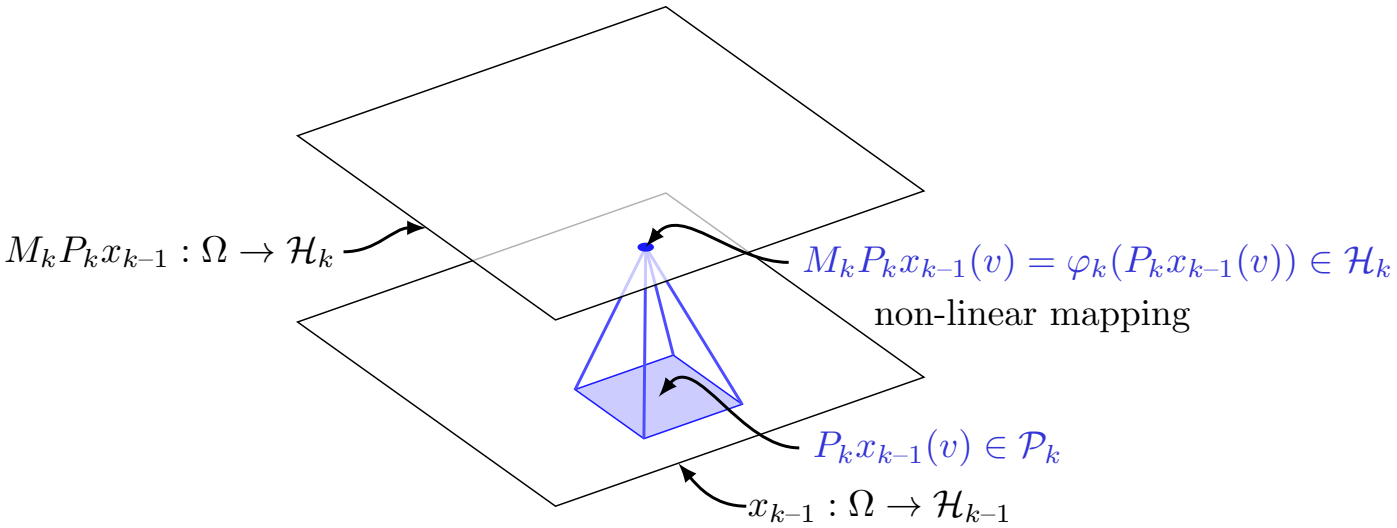
Patch extraction operator P_k

$$P_k x_{k-1}(u) := (v \mapsto x_{k-1}(u+v))_{v \in S_k} \in \mathcal{P}_k$$

- S_k: patch shape, e.g. box
- $\mathcal{P}_k = \mathcal{H}_{k-1}^{S_k}$
- P_k is **linear**, and **preserves the norm**: $\|P_k x_{k-1}\| = \|x_{k-1}\|$

Non-linear mapping operator M_k

$$M_k P_k x_{k-1}(u) := \varphi_k(P_k x_{k-1}(u)) \in \mathcal{H}_k$$



Non-linear mapping operator M_k

$$M_k P_k x_{k-1}(u) := \varphi_k(P_k x_{k-1}(u)) \in \mathcal{H}_k$$

- $\varphi_k : \mathcal{P}_k \to \mathcal{H}_k$ pointwise non-linearity on patches (kernel map)
- We assume **non-expansivity**: for $z, z' \in \mathcal{P}_k$

$$\|\varphi_k(z)\| \leq \|z\| \quad \text{and} \quad \|\varphi_k(z) - \varphi_k(z')\| \leq \|z - z'\|$$

- M_k then satisfies, for $x, x' \in L^2(\Omega, \mathcal{P}_k)$

$$\|M_k x\| \leq \|x\| \quad \text{and} \quad \|M_k x - M_k x'\| \leq \|x - x'\|$$

Non-linear mapping operator M_k

$$M_k P_k x_{k-1}(u) := \varphi_k(P_k x_{k-1}(u)) \in \mathcal{H}_k$$

- $\varphi_k : \mathcal{P}_k \to \mathcal{H}_k$ pointwise non-linearity on patches
- We assume: for $z, z' \in \mathcal{P}_k$

$$\|\varphi_k(z)\| \leq \rho_k \|z\| \quad \text{and} \quad \|\varphi_k(z) - \varphi_k(z')\| \leq \rho_k \|z - z'\|$$

- M_k then satisfies, for $x, x' \in L^2(\Omega, \mathcal{P}_k)$

$$\|M_k x\| \leq \rho_k \|x\| \quad \text{and} \quad \|M_k x - M_k x'\| \leq \rho_k \|x - x'\|$$

- (can think instead: $\varphi_k(z) = \text{ReLU}(W_k z)$, ρ_k-**Lipschitz** with $\rho_k = \|W_k\|$)

φ_k from kernels

- Kernel mapping of **homogeneous dot-product kernels**:

$$K_k(z, z') = \|z\|\|z'\|\kappa_k\left(\frac{\langle z, z'\rangle}{\|z\|\|z'\|}\right) \quad \text{with} \quad \kappa_k(1) = 1.$$

- Commonly used for hierarchical kernels
- $\|\varphi_k(z)\| = K_k(z, z)^{1/2} = \|z\|$
- $\|\varphi_k(z) - \varphi_k(z')\| \leq \|z - z'\|$ if $\kappa_k'(1) \leq 1$
- $\Longrightarrow$ non-expansive

φ_k from kernels

- Kernel mapping of **homogeneous dot-product kernels**:

$$K_k(z,z') = \|z\|\|z'\|\kappa_k\left(\frac{\langle z,z'\rangle}{\|z\|\|z'\|}\right) \quad \text{with} \quad \kappa_k(1) = 1.$$

- Commonly used for hierarchical kernels
- $\|\varphi_k(z)\| = K_k(z,z)^{1/2} = \|z\|$
- $\|\varphi_k(z) - \varphi_k(z')\| \leq \|z - z'\|$ if $\kappa_k'(1) \leq 1$
- $\Longrightarrow$ non-expansive
- Examples:
  - $\kappa_{\text{exp}}(\langle z,z'\rangle) = e^{\langle z,z'\rangle - 1}$ (Gaussian kernel on the sphere)
  - $\kappa_{\text{inv-poly}}(\langle z,z'\rangle) = \frac{1}{2-\langle z,z'\rangle}$

φ_k from kernels: CKNs approximation

φ_k from kernels: CKNs approximation

Convolutional Kernel Networks approximation:

- Approximate $\varphi_k(z)$ by **projection** on $span(\varphi_k(z_1), \ldots, \varphi_k(z_p))$ (Nystrom)
- Leads to **tractable**, p-dimensional representation $\psi_k(z)$

φ_k from kernels: CKNs approximation

Convolutional Kernel Networks approximation:

- Approximate $\varphi_k(z)$ by **projection** on $span(\varphi_k(z_1), \ldots, \varphi_k(z_p))$ (Nystrom)
- Leads to **tractable**, p-dimensional representation $\psi_k(z)$
- Norm is preserved, and projection is non-expansive:

$$\begin{aligned}\|\psi_k(z) - \psi_k(z')\| &= \|\Pi_k \varphi_k(z) - \Pi_k \varphi_k(z')\| \\ &\leq \|\varphi_k(z) - \varphi_k(z')\| \leq \|z - z'\|\end{aligned}$$

φ_k from kernels: CKNs approximation

Convolutional Kernel Networks approximation:

- Approximate $\varphi_k(z)$ by **projection** on $span(\varphi_k(z_1), \ldots, \varphi_k(z_p))$ (Nystrom)
- Leads to **tractable**, p-dimensional representation $\psi_k(z)$
- Norm is preserved, and projection is non-expansive:

$$\begin{aligned}\|\psi_k(z) - \psi_k(z')\| &= \|\Pi_k \varphi_k(z) - \Pi_k \varphi_k(z')\| \\ &\leq \|\varphi_k(z) - \varphi_k(z')\| \leq \|z - z'\|\end{aligned}$$

- Non-expansive $\Longrightarrow$ **robust to additive perturbations**!

φ_k from kernels: CKNs approximation

Convolutional Kernel Networks approximation:

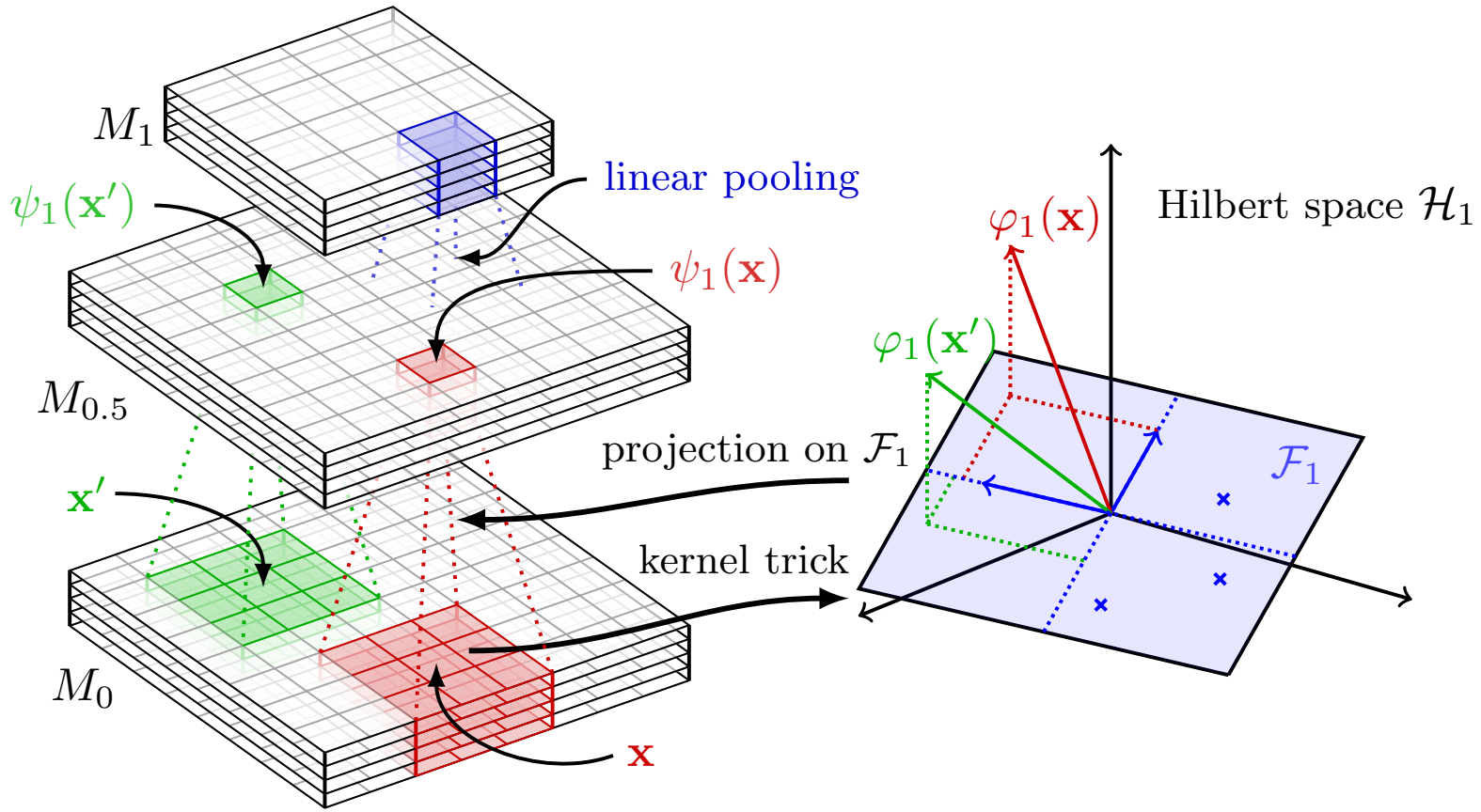
- Approximate $\varphi_k(z)$ by **projection** on $span(\varphi_k(z_1), \ldots, \varphi_k(z_p))$ (Nystrom)
- Leads to **tractable**, p-dimensional representation $\psi_k(z)$
- Norm is preserved, and projection is non-expansive:

$$\begin{aligned}\|\psi_k(z) - \psi_k(z')\| &= \|\Pi_k \varphi_k(z) - \Pi_k \varphi_k(z')\| \\ &\leq \|\varphi_k(z) - \varphi_k(z')\| \leq \|z - z'\|\end{aligned}$$

- Non-expansive $\Longrightarrow$ **robust to additive perturbations**!
- Anchor points $z_1, \ldots, z_p$ ($\approx$ filters) can be **learned from data** (K-means or backprop)

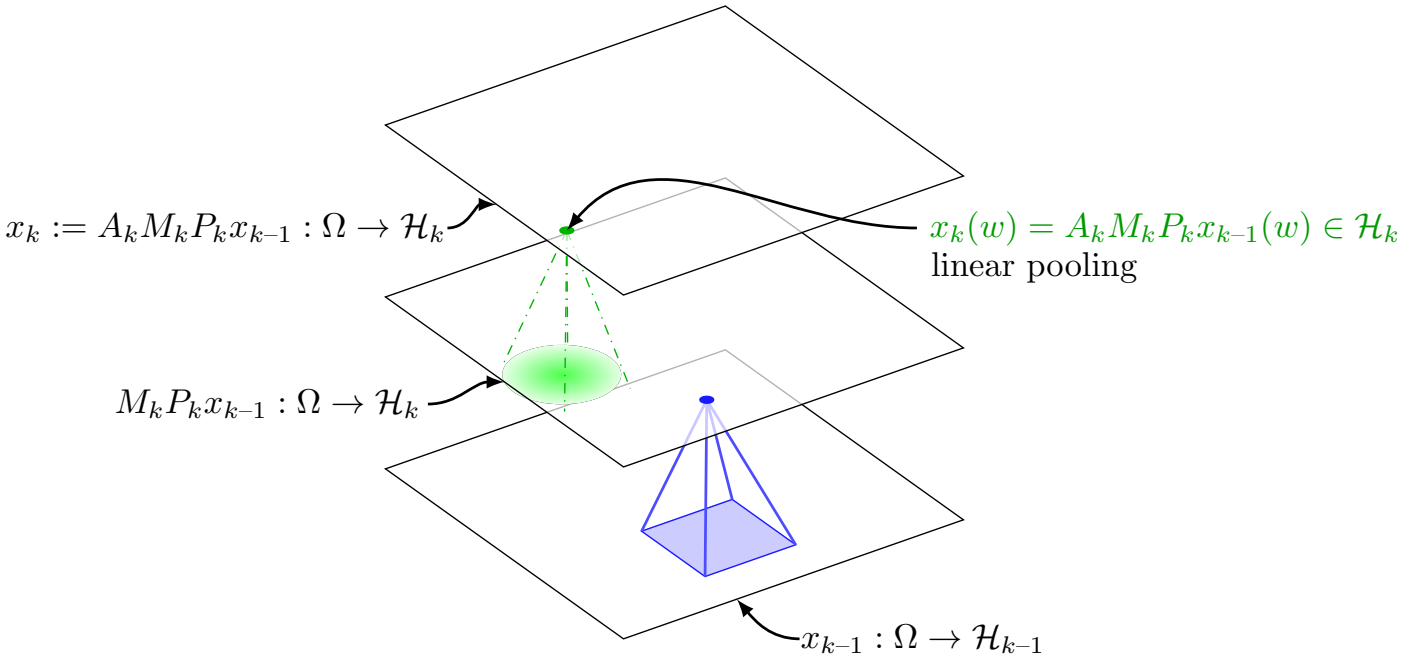
φ_k from kernels: CKNs approximation

Convolutional Kernel Networks approximation:



Pooling operator A_k

$$x_k(u) = A_k M_k P_k x_{k-1}(u) = \int_{\mathbb{R}^d} h_{\sigma_k}(u-v) M_k P_k x_{k-1}(v) dv \in \mathcal{H}_k$$

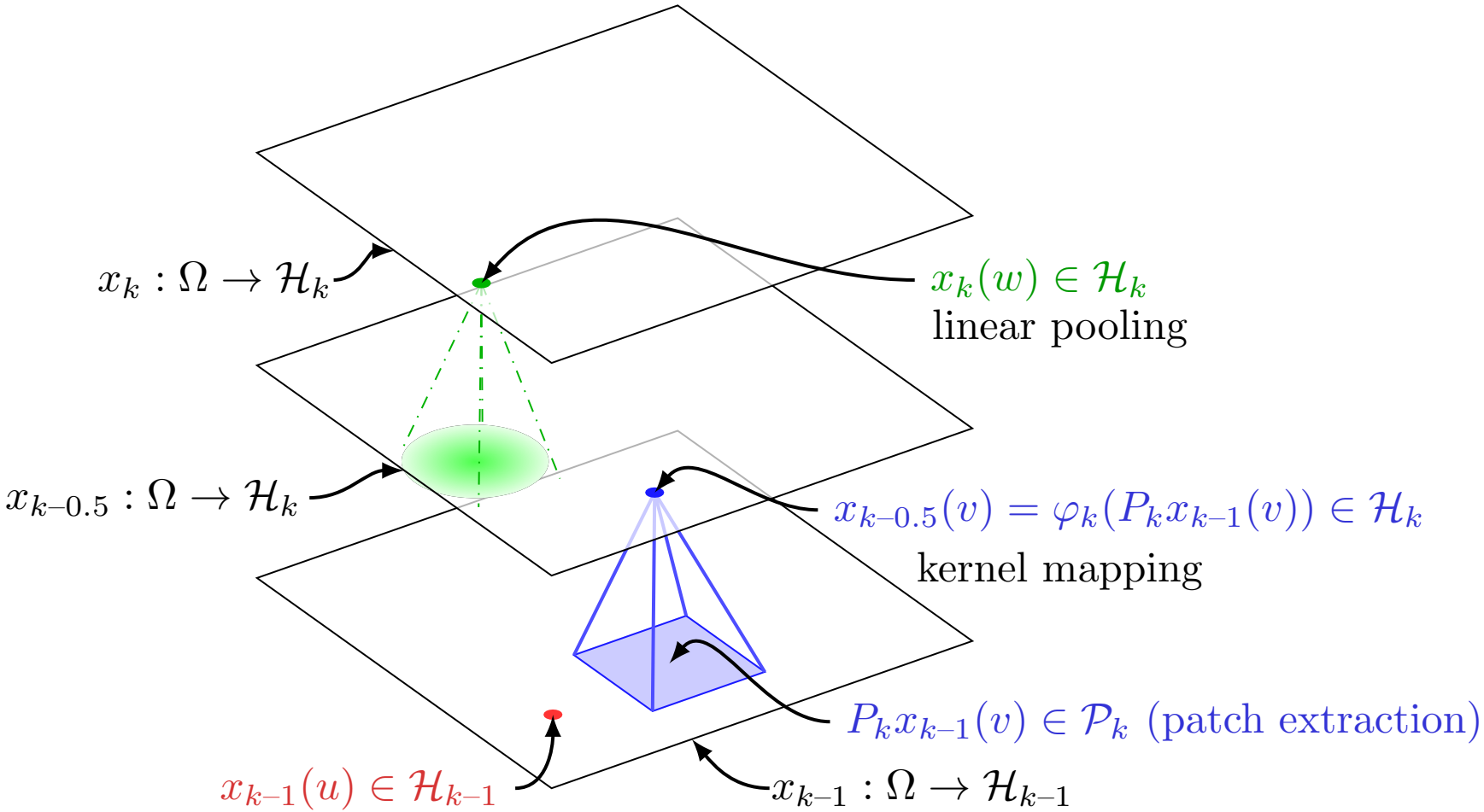


Pooling operator A_k

$$x_k(u) = A_k M_k P_k x_{k-1}(u) = \int_{\mathbb{R}^d} h_{\sigma_k}(u-v) M_k P_k x_{k-1}(v) dv \in \mathcal{H}_k$$

- h_{σ_k}: pooling filter at scale σ_k
- $h_{\sigma_k}(u) := \sigma_k^{-d} h(u/\sigma_k)$ with $h(u)$ **Gaussian**
- **linear, non-expansive operator**: $\|A_k\| \leq 1$

Recap: P_k, M_k, A_k



Multilayer construction

$$x_n \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \in L^2(\Omega, \mathcal{H}_n)$$

- S_k, σ_k grow exponentially in practice (i.e. fixed with subsampling)

Multilayer construction

$$x_n \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \in L^2(\Omega, \mathcal{H}_n)$$

- S_k, σ_k grow exponentially in practice (i.e. fixed with subsampling)
- x_0 is typically a **discrete** signal aquired with physical device
  - Natural assumption: $x_0 = A_0 x$, with x the original continuous signal, A_0 local integrator (**anti-aliasing**)

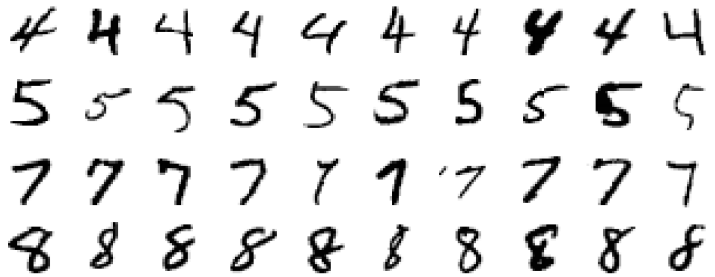
Multilayer construction

$$x_n \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \in L^2(\Omega, \mathcal{H}_n)$$

- S_k, σ_k grow exponentially in practice (i.e. fixed with subsampling)
- x_0 is typically a **discrete** signal aquired with physical device
  - Natural assumption: $x_0 = A_0 x$, with x the original continuous signal, A_0 local integrator (**anti-aliasing**)
- **Prediction layer**: e.g. linear
  - $f(x_0) = \langle w, x_n \rangle$
  - "linear kernel" $\mathcal{K}(x_0, x_0') = \langle x_n, x_n' \rangle = \int_\Omega \langle x_n(u), x_n'(u) \rangle du$

Stability to deformations: definitions

- $\tau : \Omega \to \Omega$: C^1-diffeomorphism
- $L_\tau x(u) = x(u - \tau(u))$: action operator
- Much richer group of transformations than translations



Stability to deformations: definitions

- Representation $\Phi(\cdot)$ is **stable** [Mallat, 2012] if:

$$\|\Phi(L_\tau x) - \Phi(x)\| \leq (C_1 \|\nabla \tau\|_\infty + C_2 \|\tau\|_\infty)\|x\|$$

- $\|\nabla \tau\|_\infty = \sup_u \|\nabla \tau(u)\|$ controls deformation
- $\|\tau\|_\infty = \sup_u |\tau(u)|$ controls translation
- $C_2 \to 0$: translation invariance

Warmup: translation invariance

- Representation:

$$\Phi_n(x) \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x.$$

- Translation: $L_c x(u) = x(u - c)$

Warmup: translation invariance

- Representation:

$$\Phi_n(x) \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x.$$

- Translation: $L_c x(u) = x(u-c)$
- *Equivariance* - all operators commute with L_c: $\Box L_c = L_c \Box$

$$\begin{aligned} \|\Phi(L_c x) - \Phi(x)\| &= \|L_c \Phi(x) - \Phi(x)\| \\ &\leq \|L_c A_n - A_n\| \cdot \|x\| \end{aligned}$$

Warmup: translation invariance

- Representation:

$$\Phi_n(x) \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x.$$

- Translation: $L_c x(u) = x(u - c)$
- *Equivariance* - all operators commute with L_c: $\Box L_c = L_c \Box$

$$\begin{aligned} \|\Phi(L_c x) - \Phi(x)\| &= \|L_c \Phi(x) - \Phi(x)\| \\ &\leq \|L_c A_n - A_n\| \cdot \|x\| \end{aligned}$$

- Mallat [2012]: $\|L_\tau A_n - A_n\| \leq \frac{C_2}{\sigma_n} \|\tau\|_\infty$

Warmup: translation invariance

- Representation:

$$\Phi_n(x) \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x.$$

- Translation: $L_c x(u) = x(u - c)$
- *Equivariance* - all operators commute with L_c: $\Box L_c = L_c \Box$

$$\begin{aligned} \|\Phi(L_c x) - \Phi(x)\| &= \|L_c \Phi(x) - \Phi(x)\| \\ &\leq \|L_c A_n - A_n\| \cdot \|x\| \end{aligned}$$

- Mallat [2012]: $\|L_c A_n - A_n\| \leq \frac{C_2}{\sigma_n} c$

Stability to deformations

- Representation:

$$\Phi_n(x) \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x.$$

- Patch extraction P_k and pooling A_k **do not commute** with L_τ!

Stability to deformations

- Representation:

$$\Phi_n(x) \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x.$$

- Patch extraction P_k and pooling A_k **do not commute** with L_τ!
- Mallat [2012]: $\|A_k L_\tau - L_\tau A_k\| \leq C_1 \|\nabla \tau\|_\infty$

Stability to deformations

- Representation:

$$\Phi_n(x) \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x.$$

- Patch extraction P_k and pooling A_k **do not commute** with L_τ!
- Mallat [2012]: $\|[A_k, L_\tau]\| \leq C_1 \|\nabla \tau\|_\infty$

Stability to deformations

- Representation:

$$\Phi_n(x) \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x.$$

- Patch extraction P_k and pooling A_k **do not commute** with L_τ!
- Mallat [2012]: $\|[A_k, L_\tau]\| \leq C_1 \|\nabla \tau\|_\infty$
- But: $[P_k, L_\tau]$ is **unstable** at high frequencies!

Stability to deformations

- Representation:

$$\Phi_n(x) \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x.$$

- Patch extraction P_k and pooling A_k **do not commute** with L_τ!
- Mallat [2012]: $\|[A_k, L_\tau]\| \leq C_1 \|\nabla \tau\|_\infty$
- But: $[P_k, L_\tau]$ is **unstable** at high frequencies!
- Adapt to **current layer resolution**, patch size controlled by σ_{k-1}:

$$\|[P_k A_{k-1}, L_\tau]\| \leq C_1 \|\nabla \tau\|_\infty \qquad \sup_{u \in S_k} |u| \leq \kappa \sigma_{k-1}$$

Stability to deformations

- Representation:

$$\Phi_n(x) \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x.$$

- Patch extraction P_k and pooling A_k **do not commute** with L_τ!
- Mallat [2012]: $\|[A_k, L_\tau]\| \leq C_1 \|\nabla \tau\|_\infty$
- But: $[P_k, L_\tau]$ is **unstable** at high frequencies!
- Adapt to **current layer resolution**, patch size controlled by σ_{k-1}:

$$\|[P_k A_{k-1}, L_\tau]\| \leq C_1 \|\nabla \tau\|_\infty \qquad \sup_{u \in S_k} |u| \leq \kappa \sigma_{k-1}$$

- C_1 grows as $\kappa^{d+1} \implies$ more stable with **small patches** (e.g., 3x3, VGG et al.)

Stability to deformations: final result

- Representation:

$$\Phi_n(x) \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x.$$

- **Result**: if $\|\nabla\tau\|_\infty \leq 1/2$,

$$\|\Phi_n(L_\tau x) - \Phi_n(x)\| \leq \left(C_1 \left(1+n\right) \|\nabla\tau\|_\infty + \frac{C_2}{\sigma_n} \|\tau\|_\infty \right) \|x\|$$

Stability to deformations: final result

- Representation:

$$\Phi_n(x) \triangleq A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 A_0 x.$$

- **Result**: if $\|\nabla \tau\|_\infty \leq 1/2$,

$$\|\Phi_n(L_\tau x) - \Phi_n(x)\| \leq \prod_k \rho_k \left(C_1 (1+n) \|\nabla \tau\|_\infty + \frac{C_2}{\sigma_n} \|\tau\|_\infty \right) \|x\|$$

- (for generic CNNs, multiply by $\prod_k \rho_k = \prod_k \|W_k\|$)

Controlling stability

How is stability controlled?

- full kernels: $\|f\|_{\mathcal{H}_\mathcal{K}}$ (regularizer)
- CKN: $\|W\|_2$, ℓ_2 norm of last layer (regularizer)
- CNN: $\|W\|_2 \cdot \prod_k \rho_k$ (luck...? SGD magic? Parseval nets?)

Beyond the translation group

- Global invariance to other groups? (rotations, reflections, roto-translations, ...)
- Group action $L_g x(u) = x(g^{-1}u)$
- **Equivariance** in inner layers + **(global) pooling** in last layer
- Similar construction to [Cohen and Welling, 2016]

G-equivariant layer construction

- Feature maps $x(u)$ defined on $u \in G$ (G: locally compact group)
- **Patch extraction**:

$$Px(u) = (x(uv))_{v \in S}$$

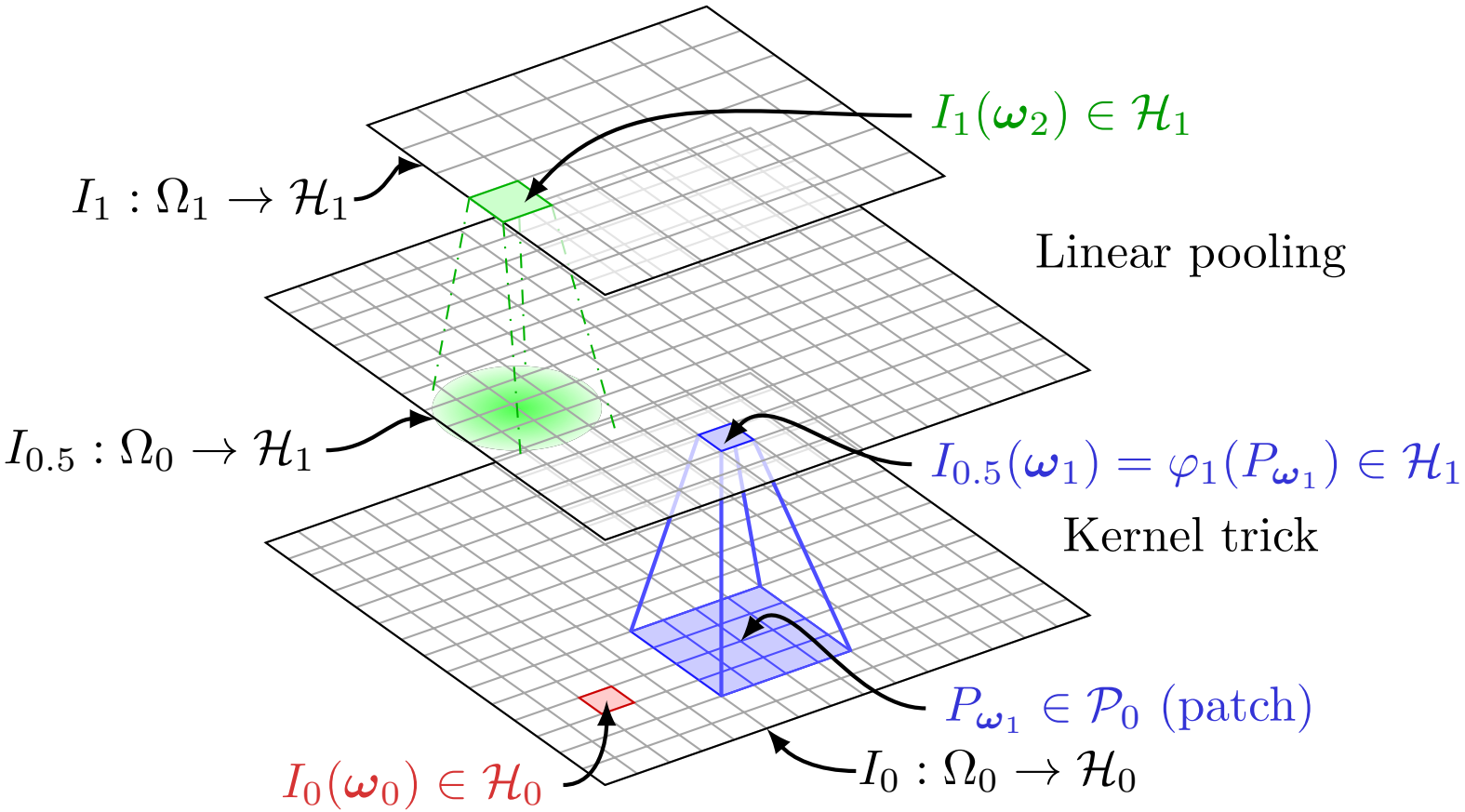
- **Non-linear mapping**: equivariant because pointwise!
- **Pooling** (μ: left-invariant Haar measure):

$$Ax(u) = \int_G x(uv)h(v)d\mu(v) = \int_G x(v)h(u^{-1}v)d\mu(v)$$

Group invariance and stability

- Stability analysis should work on "compact Lie groups" [similar to Mallat, 2012], e.g., rotations only
- For more complex groups (e.g., roto-translations):
  - Stability only w.r.t. subgroup (translations) is enough?
  - Inner layers: only pool on translation group
  - Last layer: global pooling on rotations
  - Cohen and Welling [2016]: rotation pooling in inner layers hurts performance on Rotated MNIST

Discretization and signal preservation



Discretization and signal preservation

- $\bar{x}_k$: subsampling factor s_k after pooling with scale $\sigma_k \approx s_k$:

$$\bar{x}_k[n] = A_k M_k P_k \bar{x}_{k-1}[n s_k]$$

Discretization and signal preservation

- $\bar{x}_k$: subsampling factor s_k after pooling with scale $\sigma_k \approx s_k$:

$$\bar{x}_k[n] = A_k M_k P_k \bar{x}_{k-1}[n s_k]$$

- Claim: We can recover $\bar{x}_{k-1}$ from $\bar{x}_k$ if **subsampling** $s_k \leq$ **patch size**

Discretization and signal preservation

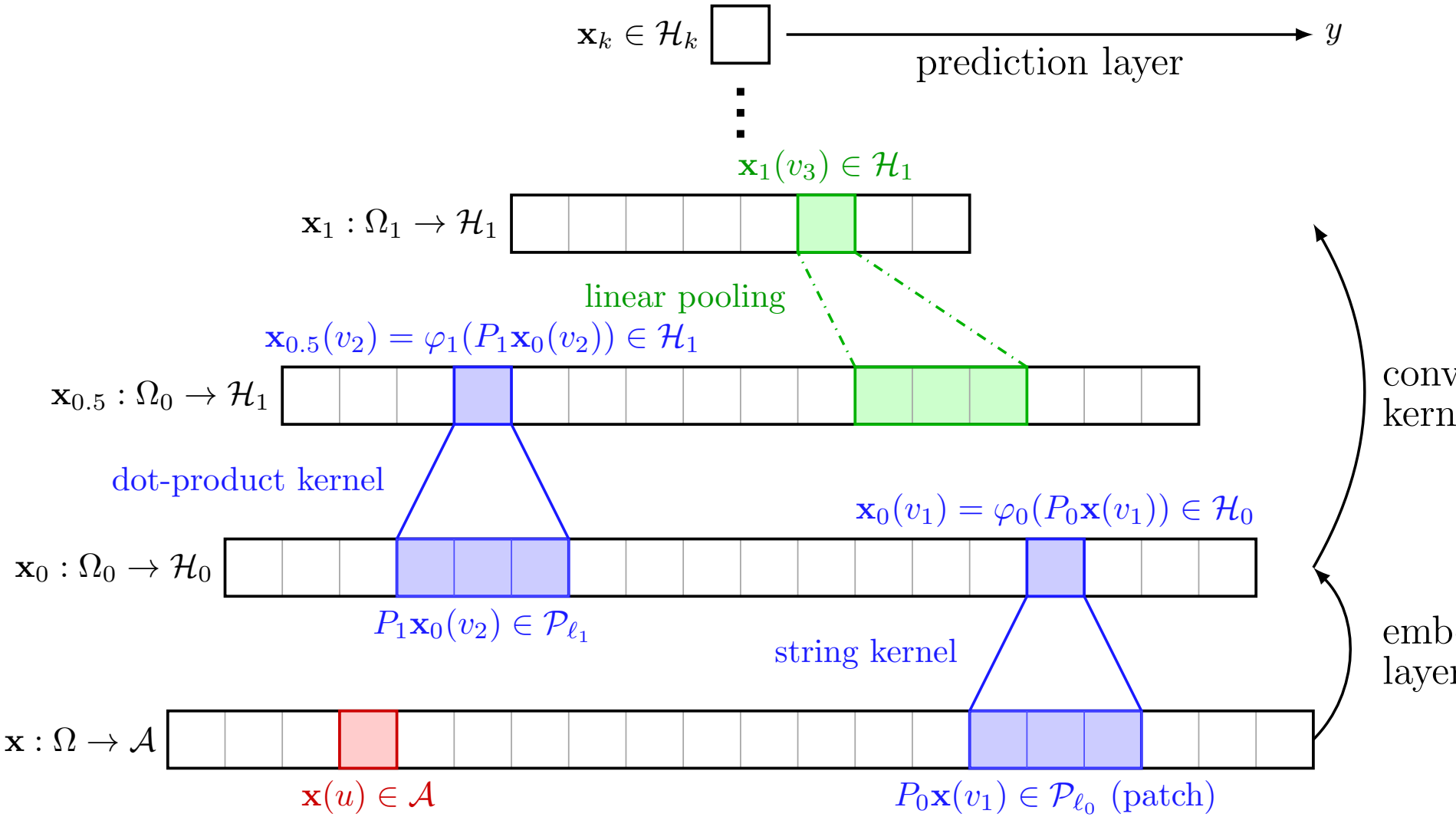
- $\bar{x}_k$: subsampling factor s_k after pooling with scale $\sigma_k \approx s_k$:

$$\bar{x}_k[n] = A_k M_k P_k \bar{x}_{k-1}[n s_k]$$

- Claim: We can recover $\bar{x}_{k-1}$ from $\bar{x}_k$ if **subsampling** $s_k \leq$ **patch size**
- How? Kernels! Recover patches with **linear functions** (contained in RKHS)

$$\langle f_w, M_k P_k x(u) \rangle = f_w(P_k x(u)) = \langle w, P_k x(u) \rangle$$

Signal recovery: example in 1D



From kernel representation to CNNs?

- Functions in the RKHS $\mathcal{H}_k$ of **patch kernels** K_k?
- CNNs in the RKHS $\mathcal{H}_\mathcal{K}$ of the **full kernel** $\mathcal{K}(x, x') = \langle \Phi(x), \Phi(x') \rangle$?
- RKHS norm $\|f\|_{\mathcal{H}_\mathcal{K}}$ for a typical CNN:
  - Stability
  - Generalization

RKHS of patch kernels K_k

$$K_k(z, z') = \|z\|\|z'\|\kappa_k\left(\frac{\langle z, z'\rangle}{\|z\|\|z'\|}\right), \qquad \kappa_k(u) = \sum_{j=0}^{\infty} b_j u^j$$

- RKHS contains **homogeneous functions**:

$$f : z \mapsto \|z\|\sigma(\langle g, z\rangle / \|z\|)$$

Homogeneous version of [**?**Zhang et al., 2017]

RKHS of patch kernels K_k

$$K_k(z, z') = \|z\|\|z'\|\kappa_k\left(\frac{\langle z, z'\rangle}{\|z\|\|z'\|}\right), \qquad \kappa_k(u) = \sum_{j=0}^{\infty} b_j u^j$$

- RKHS contains **homogeneous functions**:

$$f : z \mapsto \|z\|\sigma(\langle g, z\rangle/\|z\|)$$

- **Smooth activations**: $\sigma(u) = \sum_{j=0}^{\infty} a_j u^j$
- Norm: $\|f\|_{\mathcal{H}_k}^2 \leq C_\sigma^2(\|g\|^2)$

Homogeneous version of [?Zhang et al., 2017]

RKHS of patch kernels K_k

$$K_k(z, z') = \|z\|\|z'\|\kappa_k\left(\frac{\langle z, z'\rangle}{\|z\|\|z'\|}\right), \qquad \kappa_k(u) = \sum_{j=0}^{\infty} b_j u^j$$

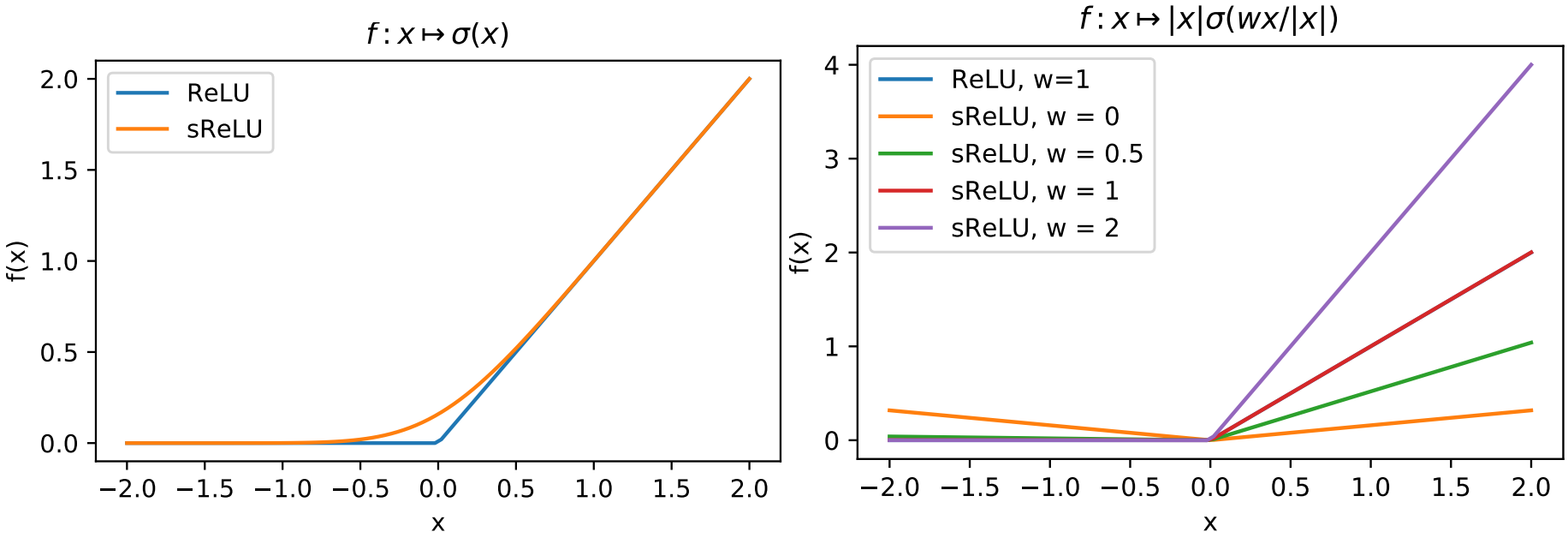
- RKHS contains **homogeneous functions**:

$$f : z \mapsto \|z\|\sigma(\langle g, z\rangle/\|z\|)$$

- **Smooth activations**: $\sigma(u) = \sum_{j=0}^{\infty} a_j u^j$
- Norm: $\|f\|_{\mathcal{H}_k}^2 \leq C_\sigma^2(\|g\|^2)$
- Examples:
  - $\sigma(u) = u$ (linear): $C_\sigma^2(\lambda^2) = O(\lambda^2)$
  - $\sigma(u) = u^p$ (polynomial): $C_\sigma^2(\lambda^2) = O(\lambda^{2p})$
  - $\sigma \approx$ sin, sigmoid, smooth ReLU: $C_\sigma^2(\lambda^2) = O(e^{c\lambda^2})$

Homogeneous version of [?Zhang et al., 2017]

RKHS of patch kernels K_k



Constructing a CNN in the RKHS $\mathcal{H}_K$

- Consider a CNN with filters $w_k^{ij}(u), u \in S_k$
- "Homogeneous" activations σ
- The CNN can be constructed hierarchically in $\mathcal{H}_K$ (define one function $f_k^i \in \mathcal{H}_k$ for each feature map)
- Norm:

$$\|f_\sigma\|^2 \leq \|w_{n+1}\|_2^2 C_\sigma^2(\|w_n\|_2^2 C_\sigma^2(\|w_{n-1}\|_2^2 C_\sigma^2(\dots)))$$

Constructing a CNN in the RKHS $\mathcal{H}_\mathcal{K}$

- Consider a CNN with filters $w_k^{ij}(u), u \in S_k$
- "Homogeneous" activations σ
- The CNN can be constructed hierarchically in $\mathcal{H}_\mathcal{K}$ (define one function $f_k^i \in \mathcal{H}_k$ for each feature map)
- Norm (linear layers):

$$\|f_\sigma\|^2 \leq \|w_{n+1}\|_2^2 \cdot \|w_n\|_2^2 \cdot \|w_{n-1}\|_2^2 \ldots \|w_1\|_2^2$$

- Linear layers: product of spectral norms

Link with generalization

- Simple bound on Rademacher complexity for linear/kernel methods:

$$\mathcal{F}_B = \{f \in \mathcal{H}_\mathcal{K}, \|f\| \leq B\} \implies \text{Rad}_n(\mathcal{F}_B) \leq O\left(\frac{BR}{\sqrt{n}}\right)$$

- Leads to margin bound $O(\|\hat{f}_n\|R/\sqrt{n})$ for a learned CNN $\hat{f}_n$ (margin $= 1/\|\hat{f}_n\|$)

Link with generalization

- Simple bound on Rademacher complexity for linear/kernel methods:

$$\mathcal{F}_B = \{f \in \mathcal{H}_\mathcal{K}, \|f\| \leq B\} \implies \text{Rad}_n(\mathcal{F}_B) \leq O\left(\frac{BR}{\sqrt{n}}\right)$$

- Leads to margin bound $O(\|\hat{f}_n\|R/\sqrt{n})$ for a learned CNN $\hat{f}_n$ (margin $= 1/\|\hat{f}_n\|$)
- For linear activations ($\|f\| \leq \|w_{n+1}\| \cdots \|w_1\|$), matches Rademacher complexity lower bound of Bartlett et al. [2017]
- Their bound has additional factors:

$$R_\mathcal{A} := \left(\prod_{i=1}^{L} \rho_i \|A_i\|_\sigma\right) \left(\sum_{i=1}^{L} \frac{\|A_i - M_i\|_1^{2/3}}{\|A_i\|_\sigma^{2/3}}\right)^{3/2}$$

Deep convolutional representations: conclusions

Study of generic properties

- Deformation stability with small patches, adapted to resolution
- Signal preservation when subsampling $\leq$ patch size
- Group invariance by changing patch extraction and pooling

Deep convolutional representations: conclusions

Study of generic properties

- Deformation stability with small patches, adapted to resolution
- Signal preservation when subsampling $\leq$ patch size
- Group invariance by changing patch extraction and pooling

Applies to learned models

- RKHS norm as a measure of model complexity
- Useful generalization bounds for CNNs
- Same quantity controls stability and generalization:
  - "higher capacity" (small margin) is needed to discriminate small deformations
  - Learning is "easier" on deformation manifold? ("manifold assumption")
  - Open: how do SGD and friends control capacity in generic CNNs?

Stability to deformations: proof idea

- Generic bound with **commutators** $[A, B] = AB - BA$:

$$\|\Phi_n(L_\tau x) - \Phi_n(x)\|$$
$$\leq \left(\sum_{k=1}^{n} \|[P_k A_{k-1}, L_\tau]\| + \|[A_n, L_\tau]\| + \|L_\tau A_n - A_n\| \right) \|x\|.$$

- Use small patch assumption to bound:

$$\|[P_k A_{k-1}, L_\tau]\| \leq \sup_{c \in S_k} \|[L_c A_{k-1}, L_\tau]\| \leq C_1 \|\nabla \tau\|_\infty$$

- From [Mallat, 2012]:

$$\|L_\tau A_\sigma - A_\sigma\| \leq \frac{C_2}{\sigma} \|\tau\|_\infty.$$

Stability to deformations: takeaways

- Small patches adapted to resolution are important for stability
- Translation invariance comes from
  - Last pooling layer
  - Exact *equivariance* in inner layers ("commute with translations")
- Intermediate pooling is for antialiasing/stable downsampling (strided convolutions enough in practice?)
- Why not just skip intermediate layers..? Loss of signal information! (See discretization below...)
- How is stability controlled?
  - full kernels: $\|f\|_{\mathcal{H}}$ (regularizer)
  - CKN: $\|W\|_2$, ℓ_2 norm of last layer (regularizer)
  - CNN: $\|W\|_2 \cdot \prod_k \rho_k$ (luck...? SGD magic? Parseval nets?)

Signal recovery with kernels

Idea:

- "Invert" kernel mapping with **linear functions** to reconstruct patches (non-overlapping)
- Recover full higher resolution (pooled) signal before downsampling
- Deconvolve to recover signal before pooling

Signal recovery with kernels

Idea:

- "Invert" kernel mapping with **linear functions** to reconstruct patches (non-overlapping)
- Recover full higher resolution (pooled) signal before downsampling
- Deconvolve to recover signal before pooling

Linear functions?

- $f_w \in \mathcal{H}_k$ s.t. $f_w(z) = \langle f_w, \varphi_k(z) \rangle_{\mathcal{H}_k} = \langle w, z \rangle_{\mathcal{P}_k}$ for a patch z
- Consider w in a basis of $\mathcal{H}_{k-1}$ for each patch location to recover signal
- Contained in RKHS of most dot-product kernels considered!

Signal recovery: takeaways

- Kernels allow recovery of the signal (up to pooling deconvolutions), when **subsampling $\leq$ patch size**
- $\Phi(x)$ contains all signal information, $f(x) = \langle f, \Phi(x) \rangle$ may focus on what's relevant to the task
- Harder to obtain for CNNs or kernel approximations, but can do well when data-dependent?
- High frequencies are hard to recover if we want translation invariance (vs. full "horizontal" multi-resolution approach like scattering): $A_n \dots A_0 x \approx A_n x$

RKHS of patch kernels K_k

$$K_k(z, z') = \|z\|\|z'\|\kappa_k\left(\frac{\langle z, z'\rangle}{\|z\|\|z'\|}\right)$$

- Expansion $\kappa_k(u) = \sum_{j=0}^{\infty} b_j u^j$
- If
  - $\sigma(u) := \sum_{j=0}^{\infty} a_j u^j$ (activation)
  - $C_\sigma^2(\|w\|^2) := \sum_{j=0}^{\infty}(a_j^2/b_j)\|w\|^{2j} < +\infty$
- Then

  $$f : z \mapsto \|z\|\sigma(\langle g, z\rangle/\|z\|)$$

  is in $\mathcal{H}_k$ with $\|f\|_{\mathcal{H}_k}^2 \leq C_\sigma^2(\|w\|^2)$.
- Homogeneous version of [?Zhang et al., 2017]

RKHS of patch kernels K_k

$$K_k(z, z') = \|z\|\|z'\|\kappa_k\left(\frac{\langle z, z'\rangle}{\|z\|\|z'\|}\right)$$

- Expansion $\kappa_k(u) = \sum_{j=0}^{\infty} b_j u^j$
- If
  - $\sigma(u) := \sum_{j=0}^{\infty} a_j u^j$ (activation)
  - $C_\sigma^2(\|w\|^2) := \sum_{j=0}^{\infty}(a_j^2/b_j)\|w\|^{2j} < +\infty$
- Then

  $$f : z \mapsto \|z\|\sigma(\langle g, z\rangle/\|z\|)$$

  is in $\mathcal{H}_k$ with $\|f\|_{\mathcal{H}_k}^2 \leq C_\sigma^2(\|w\|^2)$.
- Homogeneous version of [?Zhang et al., 2017]
- Linear functions contained when $b_1 > 0$

RKHS of full kernel $\mathcal{K}$

Theorem [e.g., **?**]

- If $\Phi : \mathcal{X} \to H$ (e.g., $\mathcal{X} = L^2(\Omega, \mathcal{H}^0), H = L^2(\Omega, \mathcal{H}_n)$)
- The RKHS of $\mathcal{K}(x, x') = \langle \Phi(x), \Phi(x') \rangle_H$ is

$$\mathcal{H}_{\mathcal{K}} := \{ f_w \ ; \ w \in H \} \quad \text{s.t.} \quad f_w : z \mapsto \langle w, \Phi(z) \rangle_H ,$$

$$\|f_w\|^2_{\mathcal{H}_{\mathcal{K}}} := \inf_{w' \in H} \{ \|w'\|^2_H \quad \text{s.t.} \quad f_w = f_{w'} \} \leq \|w\|^2_H$$

Goal: construct a $w \in L^2(\Omega, \mathcal{H}_n)$ hierarchically to obtain a CNN

Constructing a CNN in the RKHS

CNN:

- Filters $w_k^{ij} \in L^2(S_k, \mathbb{R})$
- Feature maps $z_k^i = A_k \tilde{z}_k^i \in L^2(\Omega, \mathbb{R})$ $(z_0 = x_0)$:

$$\tilde{z}_k^i(u) = \sigma\big(\langle w_k^i, P_k z_{k-1}(u) \rangle\big)$$

Constructing a CNN in the RKHS

CNN:

- Filters $w_k^{ij} \in L^2(S_k, \mathbb{R})$
- Feature maps $z_k^i = A_k \tilde{z}_k^i \in L^2(\Omega, \mathbb{R})$ $(z_0 = x_0)$:

$$\tilde{z}_k^i(u) = \sigma\big(\langle w_k^i, P_k z_{k-1}(u) \rangle\big)$$

RKHS construction:

- f_k^i in $\mathcal{H}_k$ and g_k^i in $\mathcal{P}_k$

$$g_k^i(v) = \sum_{j=1}^{p_{k-1}} w_k^{ij}(v) f_{k-1}^j \quad \text{where} \quad w_k^i(v) = (w_k^{ij}(v))_{j=1,\dots,p_{k-1}}$$

$$f_k^i(z) = \|z\| \sigma(\langle g_k^i, z \rangle / \|z\|) \quad \text{for } z \in \mathcal{P}_k.$$

Constructing a CNN in the RKHS

CNN:

- Filters $w_k^{ij} \in L^2(S_k, \mathbb{R})$
- Feature maps $z_k^i = A_k \tilde{z}_k^i \in L^2(\Omega, \mathbb{R})$ $(z_0 = x_0)$:

$$\tilde{z}_k^i(u) = n_k(u)\sigma\big(\langle w_k^i, P_k z_{k-1}(u)\rangle / n_k(u)\big)$$

RKHS construction:

- f_k^i in $\mathcal{H}_k$ and g_k^i in $\mathcal{P}_k$

$$g_k^i(v) = \sum_{j=1}^{p_{k-1}} w_k^{ij}(v) f_{k-1}^j \quad \text{where} \quad w_k^i(v) = (w_k^{ij}(v))_{j=1,\dots,p_{k-1}}$$

$$f_k^i(z) = \|z\|\sigma(\langle g_k^i, z\rangle / \|z\|) \quad \text{for } z \in \mathcal{P}_k.$$

Constructing a CNN in the RKHS

CNN:

- Linear prediction layer: $w_{n+1}^j \in L^2(\Omega, \mathbb{R})$
- $f_\sigma(x_0) = \langle w_{n+1}, z_n \rangle$

Constructing a CNN in the RKHS

CNN:

- Linear prediction layer: $w_{n+1}^j \in L^2(\Omega, \mathbb{R})$
- $f_\sigma(x_0) = \langle w_{n+1}, z_n \rangle$

RKHS construction:

- $g_\sigma \in L^2(\Omega, \mathcal{H}_n)$

$$g_\sigma(u) = \sum_{j=1}^{p_n} w_{n+1}^j(u) f_n^j \qquad \text{for all } u \in \Omega,$$

Constructing a CNN in the RKHS

CNN:

- Linear prediction layer: $w_{n+1}^j \in L^2(\Omega, \mathbb{R})$
- $f_\sigma(x_0) = \langle w_{n+1}, z_n \rangle$

RKHS construction:

- $g_\sigma \in L^2(\Omega, \mathcal{H}_n)$

$$g_\sigma(u) = \sum_{j=1}^{p_n} w_{n+1}^j(u) f_n^j \quad \text{for all } u \in \Omega,$$

We have: $\langle g_\sigma, \Phi(x_0) \rangle = f_\sigma(x_0) \implies f_\sigma \in \mathcal{H}_K$

Norm of the CNN

Simple recursive bound

$$\|f_\sigma\|^2 \leq p_n \sum_{i=1}^{p_n} \|w_{n+1}^i\|_2^2 B_{n,i},$$

with

$$B_{1,i} = C_\sigma^2(\|w_1^i\|_2^2)$$

$$B_{k,i} = C_\sigma^2 \left(p_{k-1} \sum_{j=1}^{p_{k-1}} \|w_k^{ij}\|_2^2 B_{k-1,j} \right).$$

Norm of the CNN

Spectral norm bound (not in paper...)

$$\|f_\sigma\|^2 \leq \|w_{n+1}\|_2^2 C_\sigma^2(\|w_n\|_2^2 C_\sigma^2(\|w_{n-1}\|_2^2 C_\sigma^2(\ldots))),$$

where $\|w_k\|_2^2 = \int_{S_k} \|w_k(u)\|_2^2 du$ and $\|w_k(u)\|_2$ is the spectral norm of the matrix $(w_k^{ij}(u))_{ij}$.

- With 1x1 patches (fully-connected) and no activations (linear), $C_\sigma^2(\lambda) = \lambda$, we get **product of spectral norms**
  - similar Rademacher complexity lower bound of [Bartlett et al., 2017]

References I

Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular systems biology*, 12(7): 878, 2016.

Fabio Anselmi, Lorenzo Rosasco, Cheston Tan, and Tomaso Poggio. Deep convolutional networks are hierarchical kernel machines. *arXiv preprint arXiv:1508.01084*, 2015.

Peter Bartlett, Dylan J Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1706.08498*, 2017.

Asa Ben-Hur, Cheng Soon Ong, Sören Sonnenburg, Bernhard Schölkopf, and Gunnar Rätsch. Support vector machines and kernels for computational biology. *PLoS computational biology*, 4(10):e1000173, 2008.

References II

L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *The Journal of Machine Learning Research*, 14 (1):3207–3260, 2013.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

J. V. Bouvrie, L. Rosasco, and T. Poggio. On invariance in hierarchical models. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document, 1988.

References III

Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE Transactions on pattern analysis and machine intelligence (PAMI)*, 35 (8):1872–1886, 2013.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61, 1999.

Y. Cho and L. K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*, 2016.

David Corfield, Bernhard Schölkopf, and Vladimir Vapnik. Falsificationism and statistical learning theory: Comparing the popper and vapnik-chervonenkis dimensions. *Journal for General Philosophy of Science*, 40(1):51–58, 2009.

References IV

A. Damianou and N. Lawrence. Deep Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013.

C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 38(2):295–307, 2016.

Shai Fine and Katya Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research (JMLR)*, 2:243–264, 2001.

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

References V

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *P. IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

GrÃŠgoire Montavon, Mikio L Braun, and Klaus-Robert MÃÅ§ller. Kernel analysis of deep networks. *Journal of Machine Learning Research*, 12(Sep): 2563–2581, 2011.

K-R Müller, Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernhard Scholkopf. An introduction to kernel-based learning algorithms. *IEEE transactions on neural networks*, 12(2):181–201, 2001.

References VI

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381: 607–609, 1996.

Anant Raj, Abhishek Kumar, Youssef Mroueh, P Thomas Fletcher, and Bernhard Scholkopf. Local group invariant representations via orbit embeddings. *preprint arXiv:1612.01988*, 2016.

Saburou Saitoh. *Integral transforms, reproducing kernels and their applications*, volume 369. CRC Press, 1997.

I. Schoenberg. Positive definite functions on spheres. *Duke Math. J.*, 1942.

B. Scholkopf. *Support Vector Learning*. PhD thesis, Technischen Universität Berlin, 1997.

Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10 (5):1299–1319, 1998.

References VII

Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. *arXiv preprint arXiv:1206.6471*, 2012.

John Shawe-Taylor and Nello Cristianini. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2004.

Laurent Sifre and Stéphane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2013.

Alex J Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2000.

Alex J Smola, Zoltan L Ovari, and Robert C Williamson. Regularization with dot-product kernels. In *Advances in neural information processing systems*, pages 308–314, 2001.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.

References VIII

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1995.

Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

D. Wrinch and H. Jeffreys. XLII. On certain fundamental principles of scientific inquiry. *Philosophical Magazine Series 6*, 42(249):369–390, 1921.

R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, pages 711–730. 2010.

Y. Zhang, P. Liang, and M. J. Wainwright. Convexified convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2017.