Towards Deep Kernel Machines

Julien Mairal

Inria, Grenoble

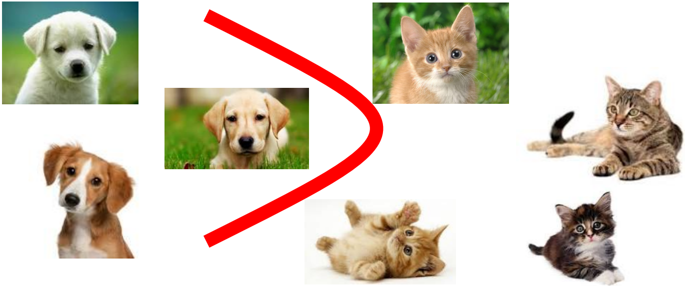
Amazon Berlin, February, 2017

Part I: Scientific Context

A quick zoom on multilayer neural networks

The goal is to learn a **prediction function** $f : \mathbb{R}^p \to \mathbb{R}$ given labeled training data $(\mathbf{x}_i, y_i)_{i=1,\ldots,n}$ with $\mathbf{x}_i$ in $\mathbb{R}^p$, and y_i in $\mathbb{R}$:

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



A quick zoom on multilayer neural networks

The goal is to learn a **prediction function** $f : \mathbb{R}^p \to \mathbb{R}$ given labeled training data $(\mathbf{x}_i, y_i)_{i=1,\ldots,n}$ with $\mathbf{x}_i$ in $\mathbb{R}^p$, and y_i in $\mathbb{R}$:

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

What is specific to multilayer neural networks?

- The "neural network" space $\mathcal{F}$ is explicitly parametrized by:

$$f(\mathbf{x}) = \sigma_k(\mathbf{A}_k\sigma_{k-1}(\mathbf{A}_{k-1}\ldots\sigma_2(\mathbf{A}_2\sigma_1(\mathbf{A}_1\mathbf{x}))\ldots)).$$

- Finding the optimal $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_k$ yields a **non-convex** optimization problem in **huge dimension.**

A quick zoom on convolutional neural networks

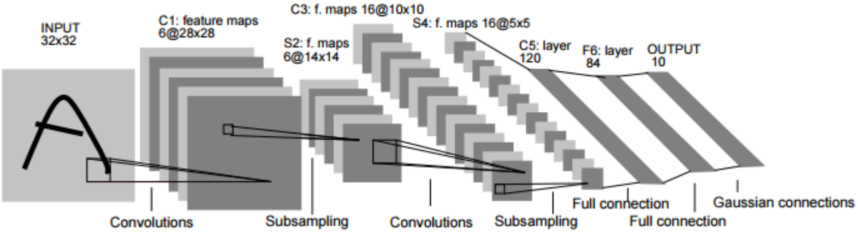


Figure: Picture from LeCun et al. [1998]

- CNNs perform "simple" operations such as convolutions, pointwise non-linearities and subsampling.
- for most successful applications of CNNs, **training is supervised**.

A quick zoom on convolutional neural networks

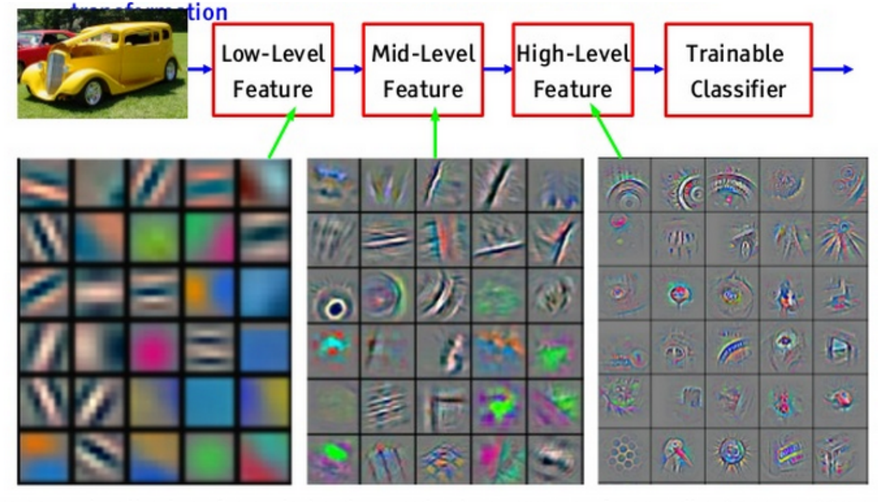


Figure: Picture from Yann LeCun's tutorial, based on Zeiler and Fergus [2014].

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.

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.

What are the main open problems?

- very little **theoretical understanding**;
- they require **large amounts of labeled data**;
- they require **manual design and parameter tuning**;

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.

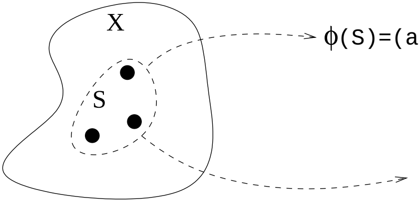
What are the main open problems?

- very little **theoretical understanding**;
- they require **large amounts of labeled data**;
- they require **manual design and parameter tuning**;

Nonetheless...

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

Context of kernel methods



$$K = \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0.5 & 1 & 0.6 \\ 0.3 & 0.6 & 1 \end{pmatrix}$$

Idea: representation by pairwise comparisons

- Define a "comparison function": $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.
- Represent a set of n data points $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ by the $n \times n$ **matrix**:

$$\mathbf{K}_{ij} := K(\mathbf{x}_i, \mathbf{x}_j).$$

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

Context of kernel methods

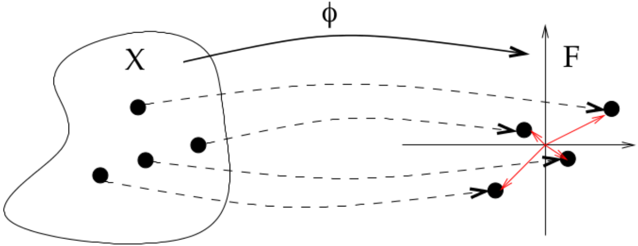
Theorem (Aronszajn, 1950)

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

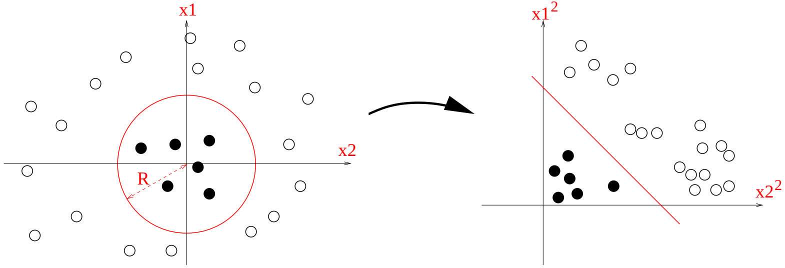
$$\varphi : \mathcal{X} \to \mathcal{H},$$

such that, for any $\mathbf{x}, \mathbf{x}$ in $\mathcal{X}$,

$$K(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}}.$$



Context of kernel methods



The classical challenge of kernel methods

Find a kernel K such that

- the data in the feature space $\mathcal{H}$ has **nice properties**, e.g., linear separability, cluster structure.
- K is **fast to compute**.

Context of kernel methods

Mathematical details

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

$$\forall \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}, \alpha_1, \ldots, \alpha_n \in \mathbb{R}, \quad \sum_{ij} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

- 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

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

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

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

Context of kernel methods

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

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

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

- **non-linear** operations in the original space $\mathcal{X}$ become **linear** in the feature space $\mathcal{H}$ since

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

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

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

Context of kernel methods

What are the main features of kernel methods?

- **decoupling** of data representation and learning algorithm;
- a huge number of **unsupervised and supervised** algorithms;
- typically, **convex optimization problems** in a supervised context;
- **versatility**: applies to vectors, sequences, graphs, sets,. . . ;
- **natural regularization function** to control the learning capacity;
- **well studied theoretical framework**.

Context of kernel methods

What are the main features of kernel methods?

- **decoupling** of data representation and learning algorithm;
- a huge number of **unsupervised and supervised** algorithms;
- typically, **convex optimization problems** in a supervised context;
- **versatility**: applies to vectors, sequences, graphs, sets, . . . ;
- **natural regularization function** to control the learning capacity;
- **well studied theoretical framework**.

But...

- **poor scalability in n**, at least $O(n^2)$;
- **decoupling** of data representation and learning may not be a good thing, according to recent **supervised** deep learning success.

Context of kernel methods

Challenges

- **Scaling-up kernel methods** with approximate feature maps;

$$K(\mathbf{x}, \mathbf{x}') \approx \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle.$$

[Williams and Seeger, 2001, Rahimi and Recht, 2007, Vedaldi and Zisserman, 2012, Le et al., 2013]...

- Design **data-adaptive and task-adaptive** kernels;
- Build **kernel hierarchies** to capture **compositional** structures.
- Introduce **supervision** in the kernel design.

Context of kernel methods

Challenges

- **Scaling-up kernel methods** with approximate feature maps;

$$K(\mathbf{x}, \mathbf{x}') \approx \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle.$$

  [Williams and Seeger, 2001, Rahimi and Recht, 2007, Vedaldi and Zisserman, 2012, Le et al., 2013]...
- Design **data-adaptive and task-adaptive** kernels;
- Build **kernel hierarchies** to capture **compositional** structures.
- Introduce **supervision** in the kernel design.

We need deep kernel machines!

Context of kernel methods

Challenges

- **Scaling-up kernel methods** with approximate feature maps;

$$K(\mathbf{x}, \mathbf{x}') \approx \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle.$$

  [Williams and Seeger, 2001, Rahimi and Recht, 2007, Vedaldi and Zisserman, 2012, Le et al., 2013]...

- Design **data-adaptive and task-adaptive** kernels;
- Build **kernel hierarchies** to capture **compositional** structures.
- Introduce **supervision** in the kernel design.

Remark

- there exists already successful **data-adaptive** kernels that rely on probabilistic models, e.g., Fisher kernel.

[Jaakkola and Haussler, 1999, Perronnin and Dance, 2007].

Some more motivation

Longer term objectives

- build a kernel for images (abstract object), for which we can precisely quantify the **invariance, stability to perturbations, recovery, and complexity** properties.
- build deep networks which can be easily **regularized**.
- build deep networks for **structured objects** (graph, sequences)...
- add more **geometric interpretation** to deep networks.
- . . .

Part II: Basic Principles of Deep Kernel Machines

Basic principles of deep kernel machines: composition

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.

$$K_3(\mathbf{x}, \mathbf{x}') = K_2(\varphi_1(\mathbf{x}), \varphi_1(\mathbf{x}')),$$

and its RKHS mapping is $\varphi_3 = \varphi_2 \circ \varphi_1$.

Basic principles of deep kernel machines: composition

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.

$$K_3(\mathbf{x}, \mathbf{x}') = K_2(\varphi_1(\mathbf{x}), \varphi_1(\mathbf{x}')),$$

and its RKHS mapping is $\varphi_3 = \varphi_2 \circ \varphi_1$.

Examples

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

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

Basic principles of deep kernel machines: composition

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?

Basic principles of deep kernel machines: composition

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).
- **it does not address any of previous challenges**.
- 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.

Basic principles of deep kernel machines: composition

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).
- **it does not address any of previous challenges**.
- 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.

Nonetheless, we will see later that this idea can be used to build a hierarchies of kernels that operate on more and more complex objects.

Basic principles of deep kernel machines: infinite NN

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

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

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

Basic principles of deep kernel machines: infinite NN

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

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

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

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

- Gaussian kernel

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

Basic principles of deep kernel machines: infinite NN

Example, arc-cosine kernels

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

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

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...

Basic principles of DKM: dot-product kernels

Another basic link between kernels and neural networks can be obtained by considering dot-product kernels.

A classical old result

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

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

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

Remark

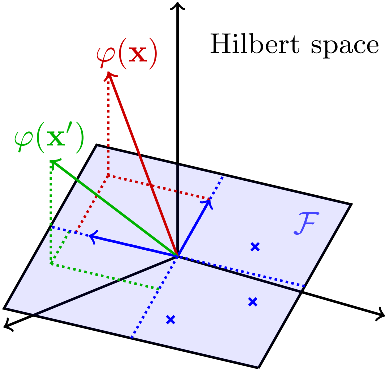
- the proposition holds if $\mathcal{X}$ is the unit sphere of some Hilbert space and $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{R}^d}$ is replaced by the corresponding inner-product.

Basic principles of DKM: dot-product kernels

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

$$\mathcal{F} = \text{span}(\varphi(\mathbf{z}_1), \ldots, \varphi(\mathbf{z}_p)),$$

for some anchor points $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_p]$ in $\mathbb{R}^{d \times p}$



Basic principles of DKM: dot-product kernels

The projection is equivalent to

$$\Pi_{\mathcal{F}}[\mathbf{x}] := \sum_{j=1}^{p} \beta_j^\star \varphi(\mathbf{z}_j) \quad \text{with} \quad \boldsymbol{\beta}^\star \in \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \left\| \varphi(\mathbf{x}) - \sum_{j=1}^{p} \beta_j \varphi(\mathbf{z}_j) \right\|_{\mathcal{H}}^2,$$

Then, it is possible to show that with $K(\mathbf{x}, \mathbf{y}) = \kappa(\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{R}^d})$,

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

with

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

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.

Part III: Convolutional Kernel Networks

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.

The model of this presentation

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

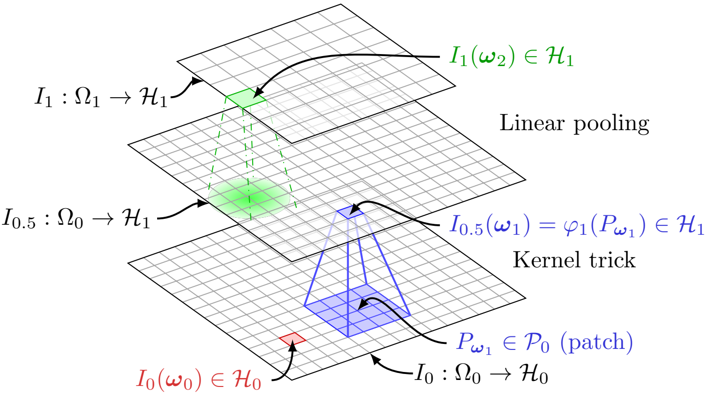
Related work

- 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].

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 $I : \Omega \rightarrow \mathcal{H}$, 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 $I_0 : \Omega_0 \rightarrow \mathcal{H}_0$ to $I_1 : \Omega_1 \rightarrow \mathcal{H}_1$?

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** $\mathcal{P}_0 := \mathcal{H}_0^{e_0 \times e_0}$ endowed with its natural inner-product.
- **Define a p.d. kernel on such patches**: For all $\mathbf{x}, \mathbf{x}'$ in $\mathcal{P}_0$,

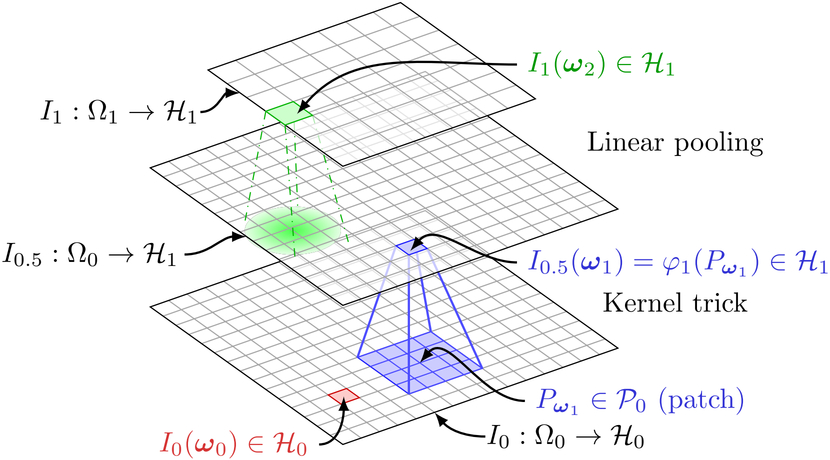
$$K_1(\mathbf{x}, \mathbf{x}') = \|\mathbf{x}\|_{\mathcal{P}_0} \|\mathbf{x}'\|_{\mathcal{P}_0} \kappa_1 \left(\frac{\langle \mathbf{x}, \mathbf{x}' \rangle_{\mathcal{P}_0}}{\|\mathbf{x}\|_{\mathcal{P}_0} \|\mathbf{x}'\|_{\mathcal{P}_0}} \right) \text{ if } \mathbf{x}, \mathbf{x}' \neq 0 \text{ and 0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

  Note that for $\mathbf{y}, \mathbf{y}'$ normalized, we may choose

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

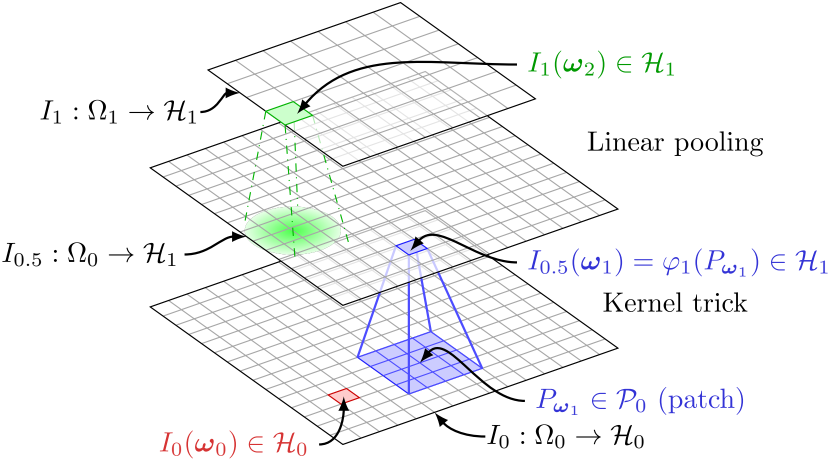
- **We call** $\mathcal{H}_1$ the RKHS and define a **mapping** $\varphi_1 : \mathcal{P}_0 \to \mathcal{H}_1$.
- Then, we may define the map $I_{0.5} : \Omega_0 \to \mathcal{H}_1$ that carries the representations in $\mathcal{H}_1$ of the patches from I_0 at all locations in Ω_0.

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

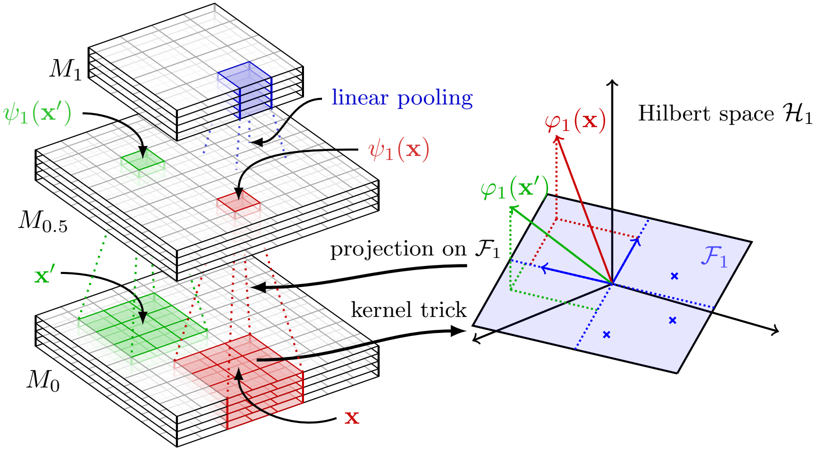


Figure: The convolutional kernel network model between layers 0 and 1.

Unsupervised learning for convolutional kernel networks

Formally, this means using the Nyström approximation

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

$$\mathcal{F}_j := \text{Span}\left(\varphi(\mathbf{z}_{j,1}), \ldots, \varphi(\mathbf{z}_{j,p_j})\right),$$

  where the $\mathbf{z}_{1,j}$'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

$$\psi_j(\mathbf{x}) := \|\mathbf{x}\|\kappa_j(\mathbf{Z}_j^\top \mathbf{Z}_j)^{-1/2}\kappa_1\left(\mathbf{Z}_j^\top \frac{\mathbf{x}}{\|\mathbf{x}\|}\right) \text{ if } \mathbf{x} \neq 0 \text{ and } 0 \text{ otherwise,}$$

  where $\mathbf{Z}_j = [\mathbf{z}_{j,1}, \ldots, \mathbf{z}_{j,p_j}]$ and $\|.\|$ is the Euclidean norm.
- The interpretation is **convolution** with filters $\mathbf{Z}_j$, **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 $I_{0.5} : \Omega_0 \to \mathcal{H}_1$.

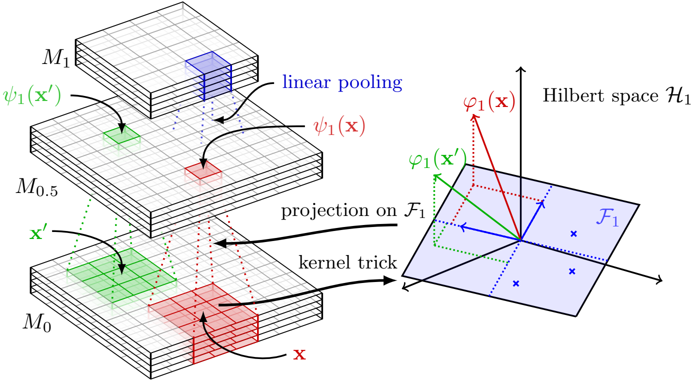


Figure: The convolutional kernel network model between layers 0 and 1.

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 $\mathbf{Z}_j$;
- **compute the projection matrix** $\kappa_j(\mathbf{Z}_j^\top \mathbf{Z}_j)^{-1/2}$.

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

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

where $\rho = \|\mathbf{x}\|$ is the gradient intensity and θ is the orientation.

- A natural choice of filters $\mathbf{Z}$ would be

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

- Then, the vector $\psi(\mathbf{x}) = \|\mathbf{x}\|\kappa_1(\mathbf{Z}^\top\mathbf{Z})^{-1/2}\kappa_1\left(\mathbf{Z}^\top\frac{\mathbf{x}}{\|\mathbf{x}\|}\right)$, 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

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

- 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

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

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 $\mathbf{W}$.
- we **alternate** between the optimization of $\mathcal{Z}$ and of $\mathbf{W}$;
- for $\mathbf{W}$, 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 $\kappa_j(\mathbf{Z}_j^\top \mathbf{Z}_j)^{-1/2}$ w.r.t $\mathbf{Z}_j$, 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**.

Part IV: Applications

Image classification

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

Dataset	$\sharp$ 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

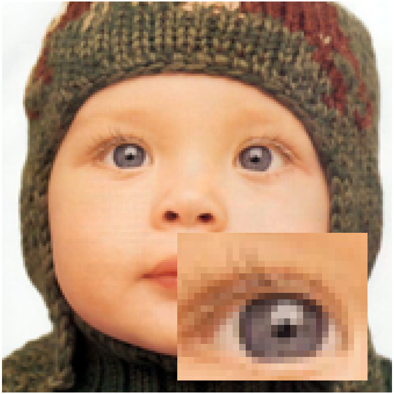
Figure: Figure from the NIPS'16 paper. Error rates in percents.

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**.

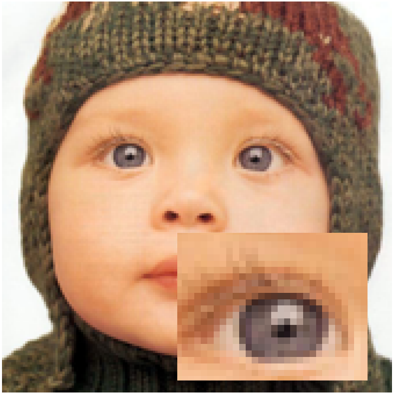


(a) Low-resolution **y**

(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

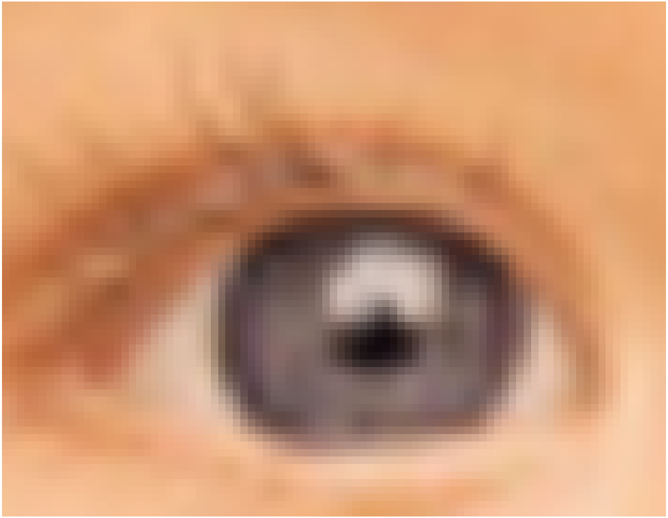


Figure: Bicubic

Image super-resolution

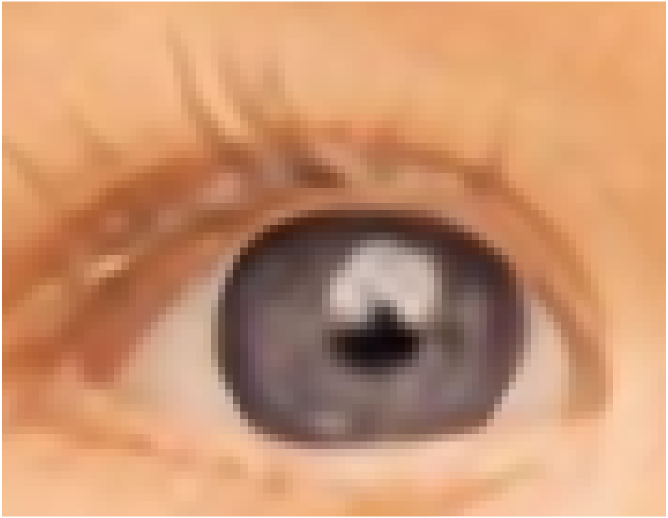
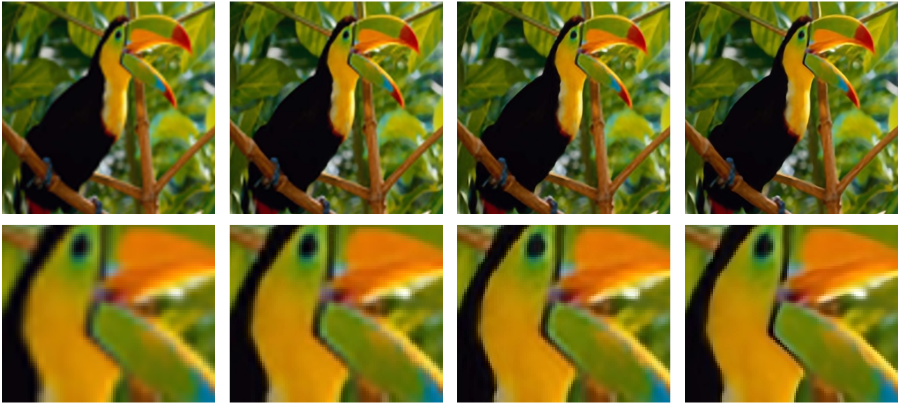


Figure: SCKN

Image super-resolution



Bicubic | Sparse coding | CNN | SCKN (Ours)

Figure: Results for x3 upscaling.

Image super-resolution

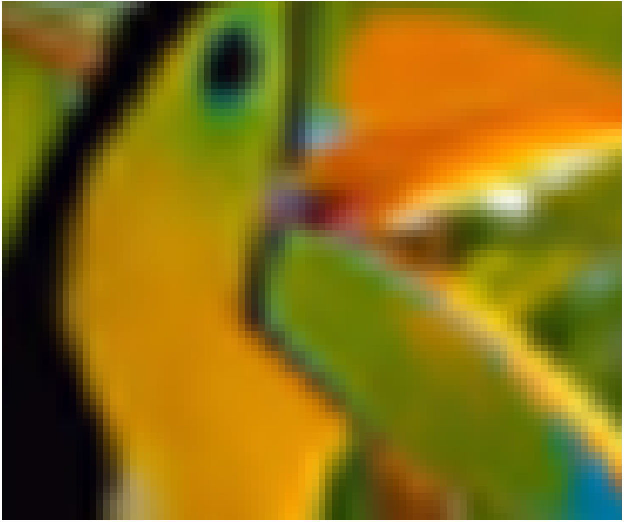
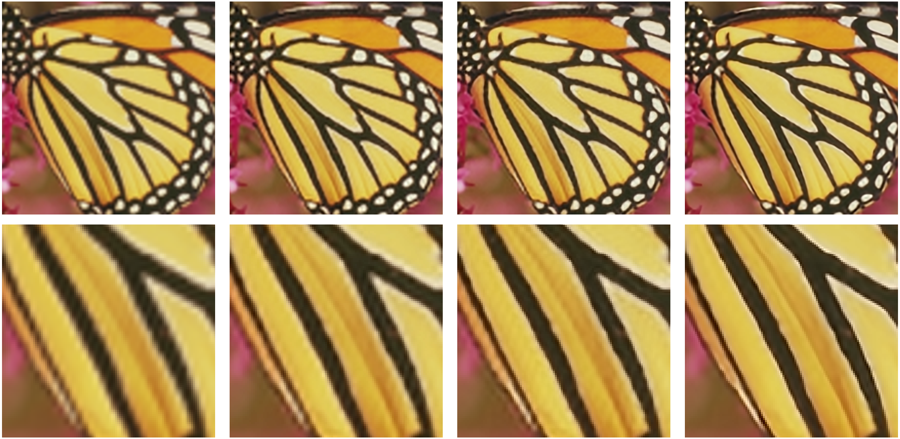


Figure: Bicubic

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

References I

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

L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *Proc. CVPR*, 2011.

J. V. Bouvrie, L. Rosasco, and T. Poggio. On invariance in hierarchical models. In *Adv. NIPS*, 2009.

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

Y. Cho and L. K. Saul. Kernel methods for deep learning. In *Adv. NIPS*, 2009.

A. Damianou and N. Lawrence. Deep Gaussian processes. In *Proc. AISTATS*, 2013.

References II

C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE T. Pattern Anal.*, 38(2):295–307, 2016.

Shai Fine and Katya Scheinberg. Efficient svm training using low-rank kernel representations. *J. Mach. Learn. Res.*, 2(Dec):243–264, 2001.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, 1999.

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proc. CVPR*, 2016.

Quoc Le, Tamás Sarlós, and Alexander Smola. Fastfood-computing hilbert space expansions in loglinear time. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 244–252, 2013.

References III

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

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.

F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Proc. CVPR*, 2007.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Adv. NIPS*, 2007.

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.

References IV

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

J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. 2004.

Alex J Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. 2000.

A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *IEEE T. Pattern Anal.*, 34(3):480–492, 2012.

Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *Proc. ICCV*, 2015.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Adv. NIPS*, 2001.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proc. ECCV*, 2014.

References V

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