Towards Deep Kernel Machines

Julien Mairal

Inria, Grenoble

Ohrid, September, 2016

Part I: Scientific Context

Adaline: a physical neural net for least square regression

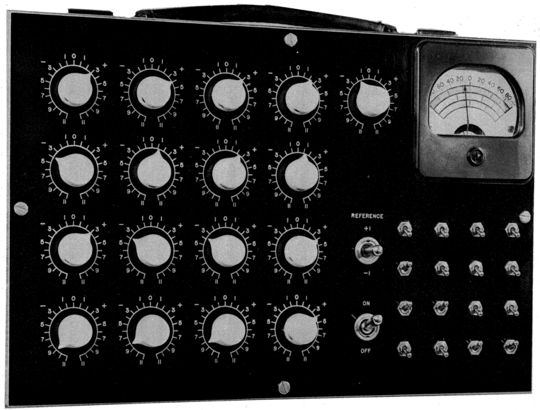
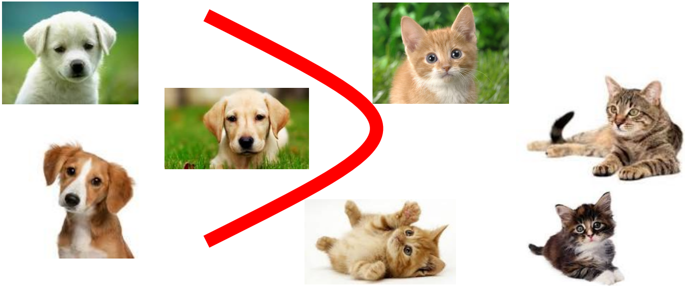


Figure: Adaline, [Widrow and Hoff, 1960]: A physical device that performs **least square regression using stochastic gradient descent**.

A quick zoom on multilayer neural networks

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 $\mathcal{Y}$:

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



A quick zoom on multilayer neural networks

The goal is to learn a **prediction function** $f : \mathcal{X} \rightarrow \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\ldots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

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

What is specific to multilayer neural networks?

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

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

- 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

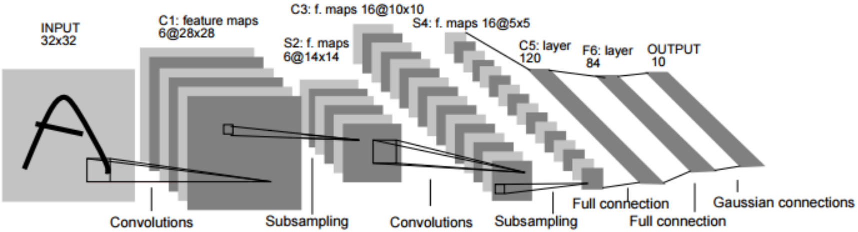


Figure: Picture from LeCun et al. [1998]

- 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

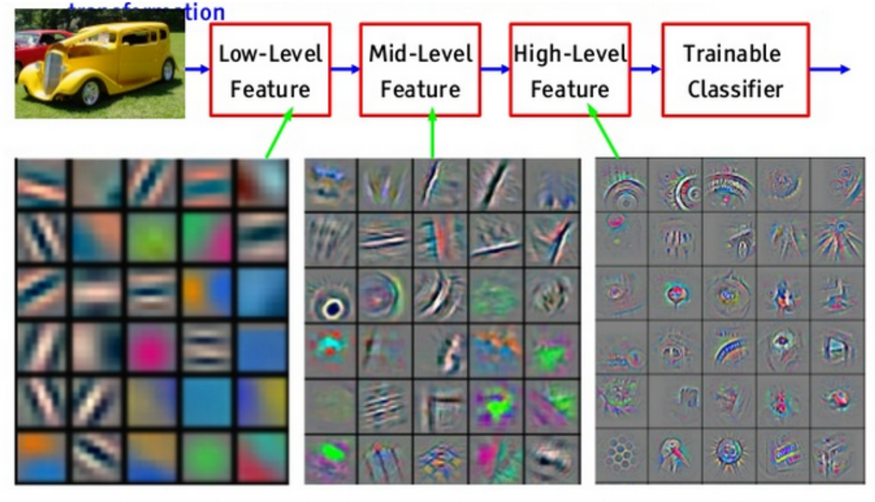


Figure: Picture from Yann LeCun's tutorial, based on Zeiler and Fergus [2014].

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?

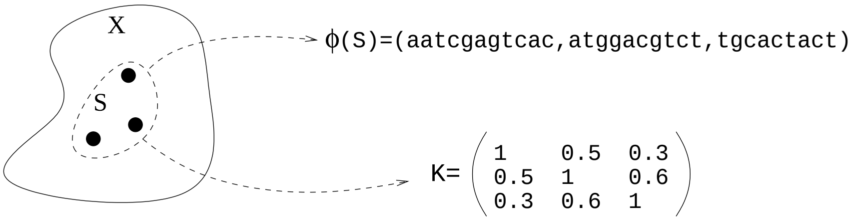
- very little **theoretical understanding**;
- they require **large amounts of labeled data**;
- they require **manual design and parameter tuning**;

Nonetheless...

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

[Choromanska et al., 2015, Livni et al., 2014, Saxena and Verbeek, 2016].

Context of kernel methods



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

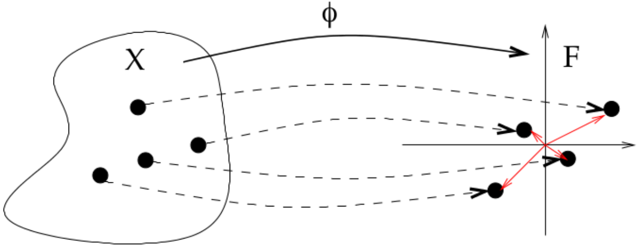
Theorem (Aronszajn, 1950)

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

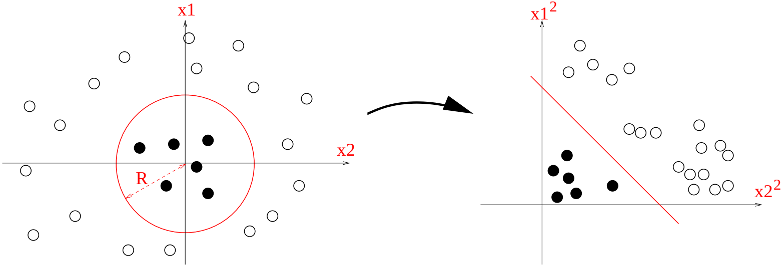
$$\varphi : \mathcal{X} \to \mathcal{H},$$

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

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

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

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

Context of kernel methods

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

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

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

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

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

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

- 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;

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

[e.g., Williams and Seeger, 2001, Rahimi and Recht, 2007, Vedaldi and Zisserman, 2012]

- Design **data-adaptive and task-adaptive** kernels;
- Build **kernel hierarchies** to capture **compositional** structures.

Context of kernel methods

Challenges

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

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

[e.g., Williams and Seeger, 2001, Rahimi and Recht, 2007, Vedaldi and Zisserman, 2012]

- Design **data-adaptive and task-adaptive** kernels;
- Build **kernel hierarchies** to capture **compositional** structures.

We need deep kernel machines!

Context of kernel methods

Challenges

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

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

[e.g., Williams and Seeger, 2001, Rahimi and Recht, 2007, Vedaldi and Zisserman, 2012]

- Design **data-adaptive and task-adaptive** kernels;
- Build **kernel hierarchies** to capture **compositional** structures.

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].

Part II: Convolutional Kernel Networks

Convolutional kernel networks

In a nutshell...

- the (happy?) **marriage** of kernel methods and CNNs;
- a hierarchy of kernels for **local image neighborhoods**;
- kernel approximations with **unsupervised or supervised** training;
- applications to **image retrieval and image super-resolution**.

First proof of concept with unsupervised learning

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

More mature model, compatible with supervised learning

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

This presentation follows the NIPS'16 paper.

Convolutional kernel networks

Idea 1

use the kernel trick to represent image neighborhoods in a RKHS.

Consider an image $I_0 : \Omega_0 \to \mathbb{R}^{p_0}$ with p_0 channels. Given **two image patches** $\mathbf{x}, \mathbf{x}'$ of size $e_0 \times e_0$, represented as vectors in $\mathbb{R}^{p_0 e_0^2}$, define

$$K_1(\mathbf{x}, \mathbf{x}') = \|\mathbf{x}\| \, \|\mathbf{x}'\| \, \kappa_1 \left(\left\langle \frac{\mathbf{x}}{\|\mathbf{x}\|}, \frac{\mathbf{x}'}{\|\mathbf{x}'\|} \right\rangle \right) \quad \text{if} \;\; \mathbf{x}, \mathbf{x}' \neq 0 \quad \text{and 0 otherwise,}$$

To ensure positive-definiteness, κ_1 needs to be smooth and its Taylor expansion have non-negative coefficients (exercise)

Convolutional kernel networks

Idea 1

use the kernel trick to represent image neighborhoods in a RKHS.

Consider an image $I_0 : \Omega_0 \to \mathbb{R}^{p_0}$ with p_0 channels. Given **two image patches** $\mathbf{x}, \mathbf{x}'$ of size $e_0 \times e_0$, represented as vectors in $\mathbb{R}^{p_0 e_0^2}$, define

$$K_1(\mathbf{x}, \mathbf{x}') = \|\mathbf{x}\| \, \|\mathbf{x}'\| \, \kappa_1\left(\left\langle \frac{\mathbf{x}}{\|\mathbf{x}\|}, \frac{\mathbf{x}'}{\|\mathbf{x}'\|} \right\rangle\right) \quad \text{if} \quad \mathbf{x}, \mathbf{x}' \neq 0 \quad \text{and 0 otherwise},$$

To ensure positive-definiteness, κ_1 needs to be smooth and its Taylor expansion have non-negative coefficients (exercise) , e.g.,

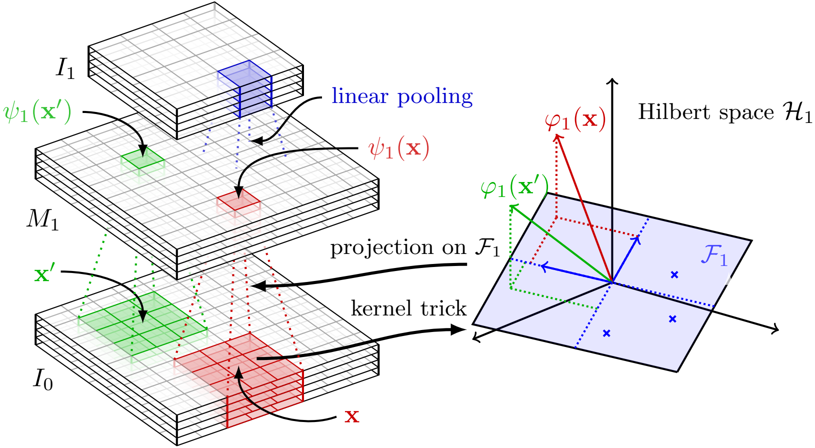
$$\kappa_1(\langle \mathbf{y}, \mathbf{y}' \rangle) = e^{\alpha_1(\langle \mathbf{y}, \mathbf{y}' \rangle - 1)} = e^{-\frac{\alpha_1}{2}\|\mathbf{y} - \mathbf{y}'\|_2^2}.$$

Then, **we have implicitly defined the RKHS $\mathcal{H}_1$ associated to K_1** and a mapping $\varphi_1 : \mathbb{R}^{p_0 e_0^2} \to \mathcal{H}_1$.

Convolutional kernel networks

Idea 1

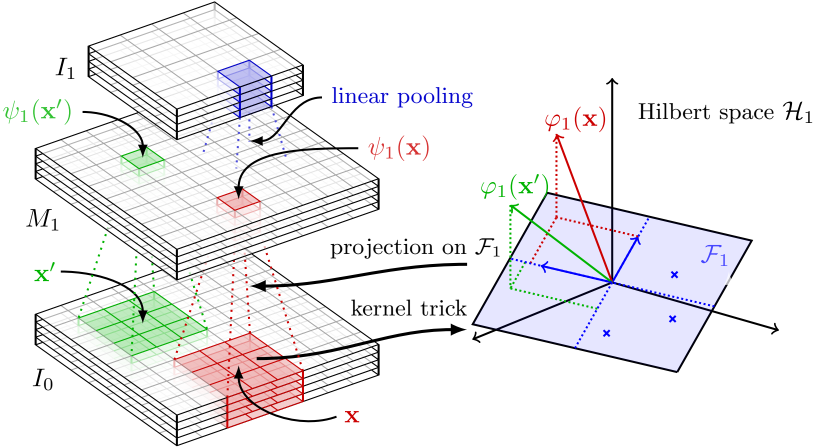
use the kernel trick to represent image neighborhoods in a RKHS.



Convolutional kernel networks

Idea 2

project onto a finite-dimensional subspace $\mathcal{F}_1$ of the RKHS $\mathcal{H}_1$



Convolutional kernel networks

Idea 2

project onto a finite-dimensional subspace $\mathcal{F}_1$ of the RKHS $\mathcal{H}_1$

- $\mathcal{F}_1$ is defined as the span of p_1 anchor points:

$$\mathcal{F}_1 = \text{Span}(\varphi_1(\mathbf{z}_1), \ldots, \varphi_1(\mathbf{z}_{p_1})).$$

  The $\mathbf{z}_j$'s are vectors in $\mathbb{R}^{p_0 e_0^2}$ with unit ℓ_2-norm;
- the orthogonal projection of $\varphi_1(\mathbf{x})$ onto $\mathcal{F}_1$ is defined as

$$f_{\mathbf{x}} := \arg\min_{f \in \mathcal{F}_1} \|\varphi_1(\mathbf{x}) - f\|^2_{\mathcal{H}_1},$$

- which is equivalent to

$$f_{\mathbf{x}} := \sum_{j=1}^{p_1} \alpha_j^\star \varphi_1(\mathbf{z}_j) \quad \text{with} \quad \boldsymbol{\alpha}^\star \in \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^{p_1}} \left\| \varphi_1(\mathbf{x}) - \sum_{j=1}^{p_1} \alpha_j \varphi_1(\mathbf{z}_j) \right\|^2_{\mathcal{H}_1}.$$

Convolutional kernel networks

Idea 2

project onto a finite-dimensional subspace $\mathcal{F}_1$ of the RKHS $\mathcal{H}_1$

- for normalized patches $\mathbf{x}$, we have $\boldsymbol{\alpha}^\star = \kappa_1(\mathbf{Z}^\top \mathbf{Z})^{-1}\kappa_1(\mathbf{Z}^\top \mathbf{x})$
- we can define a mapping $\psi_1 : \mathbb{R}^{p_0 e_0^2} \to \mathbb{R}^{p_1}$ such that

$$\langle f_{\mathbf{x}}, f_{\mathbf{x}'} \rangle_{\mathcal{H}_1} = \left\langle \psi_1(\mathbf{x}), \psi_1(\mathbf{x}') \right\rangle,$$

  with

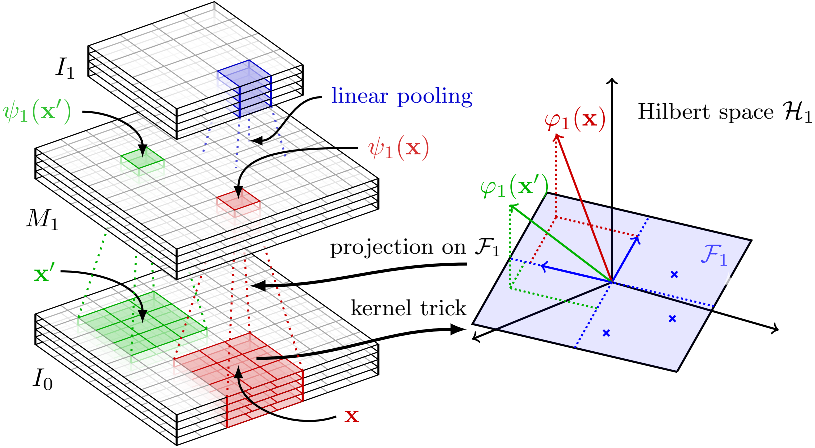
$$\psi_1(\mathbf{x}) := \|\mathbf{x}\| \kappa_1(\mathbf{Z}^\top \mathbf{Z})^{-1/2} \kappa_1\left(\mathbf{Z}^\top \frac{\mathbf{x}}{\|\mathbf{x}\|}\right) \text{ if } \mathbf{x} \neq 0 \text{ and } 0 \text{ otherwise},$$

- and subsequently define the map $M_1 : \Omega_0 \to \mathbb{R}^{p_1}$ that encodes patches from I_0 centered at positions in Ω_0.
- interpretation: **convolution, point-wise non-linearities, 1×1 convolution, contrast normalization**.

Convolutional kernel networks

Idea 2

project onto a finite-dimensional subspace $\mathcal{F}_1$ of the RKHS $\mathcal{H}_1$



Convolutional kernel networks

Idea 2

project onto a finite-dimensional subspace $\mathcal{F}_1$ of the RKHS $\mathcal{H}_1$

- with kernels, we map **patches in infinite dimension**; with the projection, we **manipulate finite-dimensional objects**.
- the projection is classical in kernel approximation techniques (Nyström method [Williams and Seeger, 2001]). The goal is to **align the subspace $\mathcal{F}_1$ with the data**, or minimize residuals. Then,

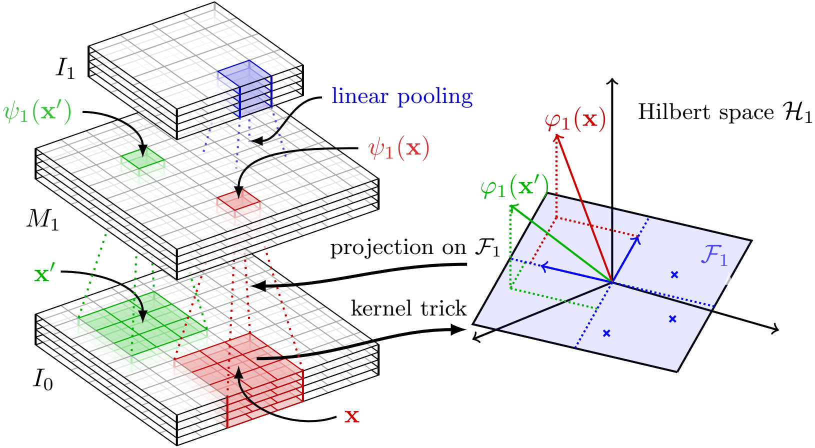
$$K_1(\mathbf{x}, \mathbf{x}') = \langle \varphi_1(\mathbf{x}), \varphi_1(\mathbf{x}') \rangle_{\mathcal{H}_1} \approx \langle f_{\mathbf{x}}, f_{\mathbf{x}'} \rangle_{\mathcal{H}_1} = \left\langle \psi_1(\mathbf{x}), \psi_1(\mathbf{x}') \right\rangle.$$

- this provides us simple techniques for **unsupervised learning** of **Z**, e.g., K-means algorithm [Zhang et al., 2008].
- for **supervised learning**, things are a bit more involved (see later).

Convolutional kernel networks

Idea 3

Linear pooling on M_1 is equivalent to pooling on $\mathcal{F}_1$.



Convolutional kernel networks

Idea 3

Linear pooling on M_1 is equivalent to pooling on $\mathcal{F}_1$.

- like in classical CNNs, we need subsampling to **reduce the dimension of feature maps.**
- we compute $I_1 : \Omega_1 \to \mathbb{R}^{p_1}$ as:

$$I_1(z) = \sum_{z' \in \Omega_0} M_1(z') e^{-\beta_1 \|z'-z\|_2^2}.$$

- linear pooling does not break the interpretation in terms of **subspace learning** in $\mathcal{H}_1$: a linear combinations of points in $\mathcal{F}_1$ is still a point in $\mathcal{F}_1$.

Convolutional kernel networks

Idea 4

Build a multilayer image representation by stacking and composing kernels.

- we obtain a hierarchy of feature maps $I_0, I_1, \ldots, I_k$, similar to CNNs;
- we define a hierarchy of kernels $K_1, \ldots, K_k$ for increasing sizes of image neighborhoods (receptive fields);
- A kernel K_k is defined on $e_k \times e_k$ patches of the map I_{k-1}, equivalently it is defined on the Cartesian product space $\mathcal{H}_{k-1}^{e_k \times e_k}$.

Convolutional kernel networks

Remark on input image pre-processing

CKNs seem to be sensitive to pre-processing; we have experimented with

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

(a) RAW RGB

(b) centering

Convolutional kernel networks

Remark on input image pre-processing

CKNs seem to be sensitive to pre-processing; we have experimented with

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

(c) RAW RGB

(d) whitening

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**.

Idea borrowed from the kernel descriptors of Bo et al. [2010].

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 1 000 000 patches from layers $k-1$ computed on an image database and normalize them;
- perform a **spherical K-means algorithm** to learn the filters $\mathbf{Z}_k$;
- **compute the projection matrix** $\kappa_k(\mathbf{Z}_k^\top \mathbf{Z}_k)^{-1/2}$.

Remember that every patch is encoded with the formula

$$\psi_k(\mathbf{x}) = \|\mathbf{x}\| \kappa_k(\mathbf{Z}_k^\top \mathbf{Z}_k)^{-1/2} \kappa_k\left(\mathbf{Z}_k^\top \frac{\mathbf{x}}{\|\mathbf{x}\|}\right).$$

Convolutional kernel networks

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

$$K_{\mathcal{Z}}(I_0, I_0') = \sum_{z \in \Omega_k} \langle f_k(z), f_k'(z) \rangle_{\mathcal{H}_k} = \sum_{z \in \Omega_k} \langle I_k(z), I_k'(z) \rangle,$$

- and we obtain the simple formulation

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

Convolutional kernel networks

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

- we **alternate** between the optimization of the filters $\mathcal{Z}$ 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 (one pass of SGD+momentum):
- we also use a **pre-conditioning heuristic on the sphere**;
- we can also learn the **kernel hyper-parameters**.

The main originality compared to CNN is the **subspace learning interpretation**, due to the projection matrix.

We also use a heuristic for automatically choosing the learning rate of SGD, which was used in all experiments (was never tuned by hand).

Convolutional kernel networks

Remark on the NIPS'14 paper (older model)

The first paper used a different principle for the kernel approximation:

$$e^{-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{x}'\|_2^2} = \left(\frac{2}{\pi\sigma^2}\right)^{\frac{m}{2}} \int_{\mathbf{w}\in\mathbb{R}^m} e^{-\frac{1}{\sigma^2}\|\mathbf{x}-\mathbf{w}\|_2^2} e^{-\frac{1}{\sigma^2}\|\mathbf{x}'-\mathbf{w}\|_2^2} d\mathbf{w},$$

and a non-convex cost function is formulated to learn the mapping

$$\psi(\mathbf{x}) = [\sqrt{\eta_l} e^{-(1/\sigma^2)\|\mathbf{x}-\mathbf{w}_l\|_2^2}]_{l=1}^p \in \mathbb{R}^p,$$

Convolutional kernel networks

Remark on the NIPS'14 paper (older model)

The first paper used a different principle for the kernel approximation:

$$e^{-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{x}'\|_2^2} = \left(\frac{2}{\pi\sigma^2}\right)^{\frac{m}{2}} \int_{\mathbf{w}\in\mathbb{R}^m} e^{-\frac{1}{\sigma^2}\|\mathbf{x}-\mathbf{w}\|_2^2} e^{-\frac{1}{\sigma^2}\|\mathbf{x}'-\mathbf{w}\|_2^2} d\mathbf{w},$$

and a non-convex cost function is formulated to learn the mapping

$$\psi(\mathbf{x}) = [\sqrt{\eta_l} e^{-(1/\sigma^2)\|\mathbf{x}-\mathbf{w}_l\|_2^2}]_{l=1}^p \in \mathbb{R}^p,$$

This is an **approximation scheme**; the mapping ψ **does not live in the RKHS**. Approximation errors accumulate from one layer to another.

The new scheme (NIPS'16) is **faster to train**, provides **better results** in the unsupervised context, and is **compatible with supervised learning**.

Related work

- first 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]...
- RBF networks [Broomhead and Lowe, 1988].

Convolutional kernel networks

Short summary of features

We obtain a particular type of CNN with

- a novel **unsupervised** learning principle;
- a **regularization function** (the norm $\|.\|_{\mathcal{H}_k}$), effective at least in the unsupervised context;
- also compatible with **supervised learning**;
- learning the filters corresponds to **learning linear subspaces**.

Some perspectives

- use similar principles for **graph-structured data**;
- connect with deep Gaussian processes;
- leverage the literature about **subspace learning**;
- use **union of subspaces**, introduce **sparsity**...

Part III: Applications

Image classification

Experiments were conducted on classical **"deep learning" datasets**, on CPUs only (at the moment).

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

We use the following 9-layer network with 512 filters per layer.

Subsampling	$\sqrt{2}$	1	$\sqrt{2}$	1	$\sqrt{2}$	1	$\sqrt{2}$	1	3
Size patches	3	1	3	1	3	1	3	1	3

- we use the squared hinge loss in a one-vs-all setting;
- we use the **supervised** CKNs;
- The regularization parameter λ and the number of epochs are set by first running the algorithm on a 80/20% validation split.

Image classification

	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

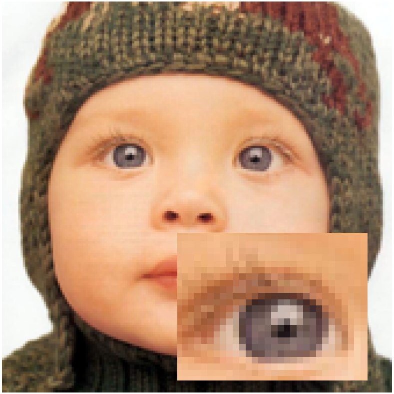
Figure: Figure from the NIPS'16 paper (preprint arXiv). Error rates in percents for single models with no data augmentation.

Remarks on CIFAR-10

- simpler model (5 layers, with integer subsampling factors) also performs well $\approx 12\%$;
- the original model of Krizhevsky et al. [2012] does $\approx 18\%$;
- the best **unsupervised** architecture has two layers, is wide (1024-16384 filters), and achieves 14.2%;
- the **unsupervised** model reported in the NIPS'14 was 21.7% (same model here 19.3%).

Image super-resolution

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

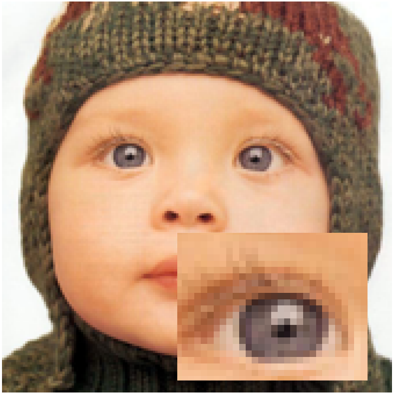


(a) Low-resolution $\mathbf{y}$

(b) High-resolution $\mathbf{x}$

Image super-resolution

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



(c) Low-resolution $\mathbf{y}$

(d) Bicubic interpolation

Image super-resolution

Following classical approaches based on CNNs [Dong et al., 2016], we want to predict high-resolution images from bicubic interpolations.

- we use the **square loss** instead of a classification loss;
- models are trained to **up-scale by a factor** 2, using a database of 200 000 pairs of high/los-res patches of size 32×32 and 16×16;
- we also use a 9-layer network with 3×3 patches, 128 filters at every layer, no pooling, no zero-padding;
- to perform $x3$ upscaling, we simply apply $x2$ twice, and downsample by $3/4$;

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";
- Kim et al. [2016] from CVPR'16 does better by using very deep CNNs and residual learning;
- 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

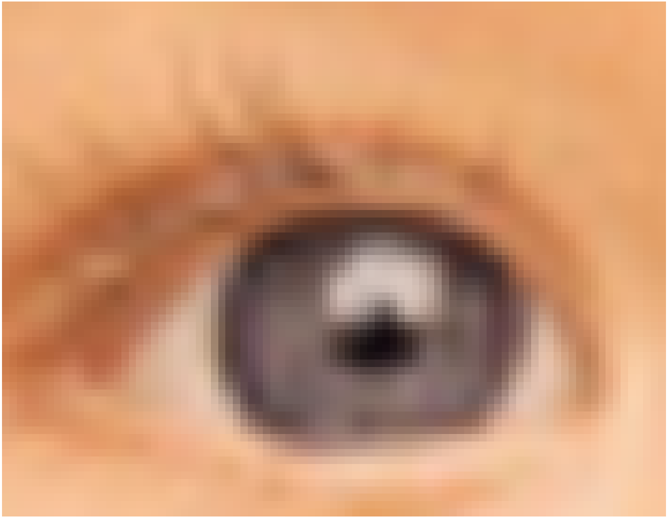


Figure: Bicubic

Image super-resolution

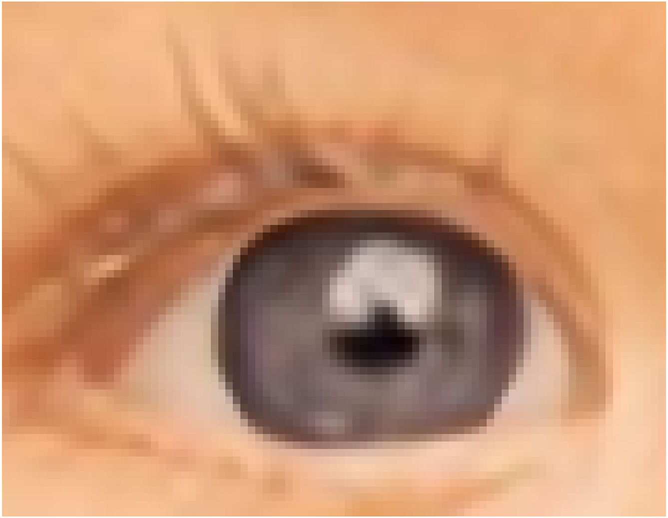
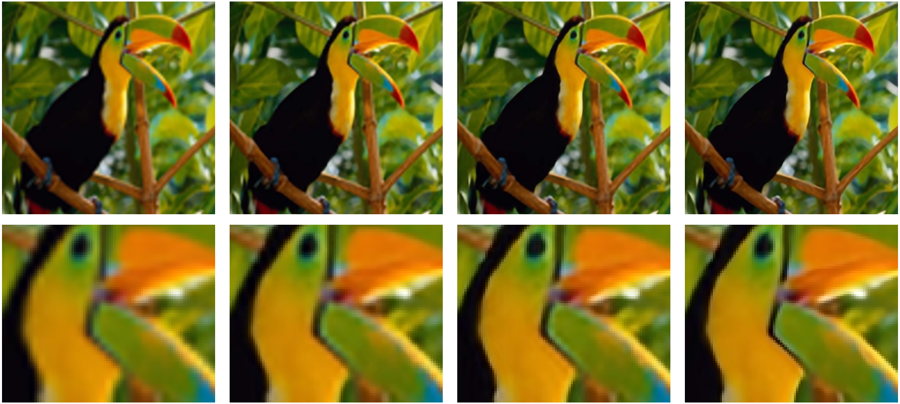


Figure: SCKN

Image super-resolution



Bicubic | Sparse coding | CNN | SCKN (Ours)

Figure: Results for x3 upscaling.

Image super-resolution

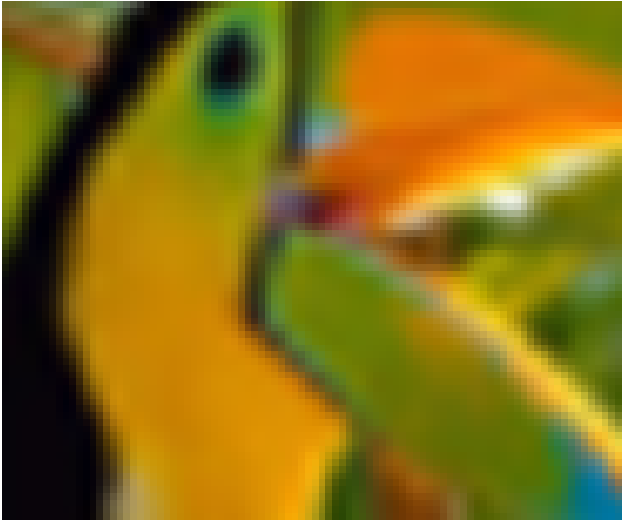
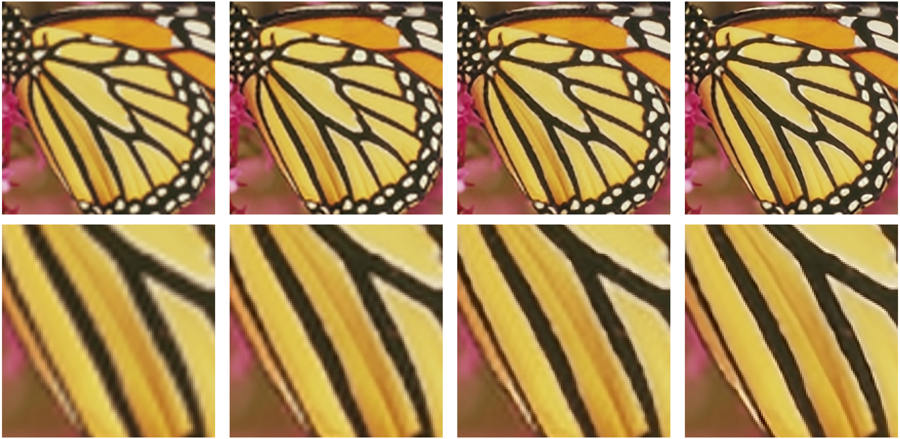


Figure: Bicubic

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

Figure: Results for x3 upscaling.

Image super-resolution

Figure: Bicubic

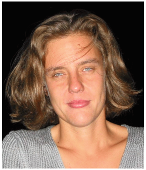
Image super-resolution

Figure: SCKN

Image retrieval

Collaborators

Zaid Harchaoui



Cordelia Schmid

Florent Perronnin

Matthijs Douze

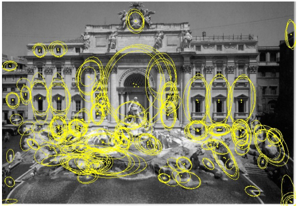
Mattis Paulin

Publications

- M. Paulin, J. Mairal, M. Douze, Z. Harchaoui, F. Perronnin and C. Schmid. Convolutional Patch Representations for Image Retrieval: an Unsupervised Approach. IJCV 2016.
- M. Paulin, M. Douze, Z. Harchaoui, J. Mairal, F. Perronnin and C. Schmid. Local Convolutional Features with Unsupervised Training for Image Retrieval. ICCV 2015.

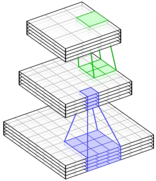
These publications use the older model of the CKN (NIPS'14).

Image retrieval



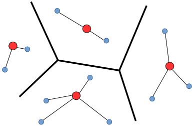
Keypoint detection

Hessian-affine



Patch description

Deep Network



Aggregation

VLAD

Remarks

- possibly followed by PCA to **reduce the dimension**;
- retrieval is performed by **simple inner-product evaluations**;
- here, we evaluate only the **patch representation**.

Image retrieval

From patches...

Image retrieval

To images...

Remarks

- We benchmark both tasks at the same time;
- retrieval **differs significantly from classification**; Training a CNN for retrieval **with supervision** is hard;
- results using supervision have been mitigated until CVPR/ECCV'16.

[Babenko et al., 2014, Babenko and Lempitsky, 2015, Gong et al., 2014, Fischer et al., 2014, Zagoruyko and Komodakis, 2015, Radenović et al., 2016, Gordo et al., 2016].

Image retrieval

- we use a patch retrieval task to optimize model parameters;
- we try different input types: RGB, RGB+whitening, gradients;

Input	Layer 1	Layer 2	dim.
CKN-raw	5x5, 5, 512	—	41,472
CKN-white	3x3, 3, 128	2x2, 2, 512	32,768
CKN-grad	1x1, 3, 16	4x4,2,1024	50,176

- training is fast, 10mn on a GPU (would be about 1mn on a CPU with the NIPS'16 paper);
- dimensionality is then reduced with PCA + whitening.

Image retrieval

Evaluation of different patch representations for patch retrieval

Architecture	coverage	Dim	RomePatches train	RomePatches test	Miko.
SIFT	51x51	128	91.6	87.9	57.8
AlexNet-conv1	11x11	96	66.4	65.0	40.9
AlexNet-conv2	51x51	256	73.8	69.9	46.4
AlexNet-conv3	99x99	384	81.6	79.2	53.7
AlexNet-conv4	131x131	384	78.4	75.7	43.4
AlexNet-conv5	163x163	256	53.9	49.6	24.4
PhilippNet	64x64	512	86.1	81.4	59.7
PhilippNet	91x91	2048	88.0	83.7	61.3
CKN-grad	51x51	1024	**92.5**	**88.1**	59.5
CKN-raw	51x51	1024	79.3	76.3	50.9
CKN-white	51x51	1024	91.9	87.7	**62.5**

[Krizhevsky et al., 2012, Fischer et al., 2014].

Image retrieval

...which become, in the same pipeline, for **image** retrieval

	Holidays	UKB	Oxford	Rome	
				train	test
SIFT	64.0	3.44	43.7	52.9	62.7
AlexNet-conv1	59.0	3.33	18.8	28.9	36.8
AlexNet-conv2	62.7	3.19	12.5	36.1	21.0
AlexNet-conv3	**79.3**	3.74	33.3	47.1	54.7
AlexNet-conv4	77.1	3.73	34.3	47.9	55.4
AlexNet-conv5	75.3	3.69	33.4	45.7	53.1
PhilippNet 64x64	74.1	3.66	38.3	50.2	60.4
PhilippNet 91x91	74.7	3.67	43.6	51.4	61.3
CKN-grad	66.5	3.42	**49.8**	**57.0**	**66.2**
CKN-raw	69.9	3.54	23.0	33.0	43.8
CKN-white	78.7	3.74	41.8	51.9	62.4
CKN-mix	**79.3**	**3.76**	43.4	54.5	65.3

Image retrieval

Comparison with other pipelines

Method \ Dataset	Holidays	UKB	Oxford
VLAD [Jégou et al., 2012]	63.4	3.47	-
VLAD++ [Arandjelovic and Zisserman, 2013]	64.6	-	**55.5**
Global-CNN [Babenko et al., 2014]	79.3	3.56	54.5
MOP-CNN [Gong et al., 2014]	80.2	-	-
Sum-pooling OxfordNet [Babenko and Lempitsky, 2015]	80.2	3.65	53.1
Ours	79.3	3.76	49.8
Ours+PCA 4096	**82.9**	**3.77**	47.2

Remarks

- this is a comparison with relatively high-dimensional descriptors;
- with dense feature extraction, our model does 55.5 for Oxford;
- supervised CNNs for retrieval have been mitigated until CVPR/ECCV'16 (see O. Chum's talk + [Gordo et al., 2016]);
- these results use the older NIPS'14 model and no supervision.

Conclusion

First achievements

- new type of convolutional networks where **learning filters amount to learning subspaces**;
- new principles for **unsupervised learning** of deep network, also compatible with **supervised learning**.
- **competitive results** for image super-resolution, classification, and patch representation in image retrieval;

Future work

- build semi-generic models for **structured data**;
- explore novel **subspace learning models and algorithms**;
- study theoretically invariant properties of the kernels;

Software

- coming soon (with GPU implementation)...

References I

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

Relja Arandjelovic and Andrew Zisserman. All about VLAD. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1269–1277, 2015.

Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *European Conference on Computer Vision*, 2014.

L. Bo, X. Ren, and D. Fox. Kernel descriptors for visual recognition. In *Adv. NIPS*, 2010.

References II

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. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *Proc. AISTATS*, 2015.

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

References III

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.

Philipp Fischer, Alexey Dosovitskiy, and Thomas Brox. Descriptor matching with Convolutional Neural Networks: a comparison to SIFT. arXiv Preprint, 2014.

Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision*, 2014.

Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. *arXiv preprint arXiv:1604.01325*, 2016.

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

References IV

Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

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

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Adv. NIPS*, 2012.

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

Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems*, pages 855–863, 2014.

References V

Grégoire Montavon, Mikio L Braun, and Klaus-Robert Müller. 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.

Filip Radenović, Giorgos Tolias, and Ondřej Chum. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. *arXiv preprint arXiv:1604.02426*, 2016.

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

Shreyas Saxena and Jakob Verbeek. Convolutional neural fabrics. *arXiv preprint arXiv:1606.02492*, 2016.

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

References VI

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

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.

B. Widrow and M. E. Hoff. Adaptive switching circuits. In *IRE WESCON convention record*, volume 4, pages 96–104. New York, 1960.

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

Serguey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

References VII

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

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

K. Zhang, I. W. Tsang, and J. T. Kwok. Improved Nyström low-rank approximation and error analysis. In *Proc. ICML*, 2008.