Views of Deep Networks from Reproducing Kernel Hilbert Spaces

Lecture 6, STATS 385, Stanford University

Zaid Harchaoui

November 1st, 2017

UNIVERSITY *of* WASHINGTON

Collaborators

PhD students

- Corinne Jones, UW
- Mattis Paulin, Inria

Collaborators

- Matthijs Douze, Facebook AI Research
- Julien Mairal, Inria
- Florent Perronnin, Naver Labs Europe (formerly Xerox Research Centre Europe)
- Cordelia Schmid, Inria

Papers

Unsupervised Convolutional Kernel Networks

- J. Mairal, P. Koniusz, Z. Harchaoui and C. Schmid. *Convolutional Kernel Networks*. Adv. NIPS, 2014.
- M. Paulin, J. Mairal, M. Douze, Z. Harchaoui, F. Perronnin, and C. Schmid. *Local Convolutional Features with Unsupervised Training for Image Retrieval*. IJCV, 2015

Readings

Kernel-based Methods

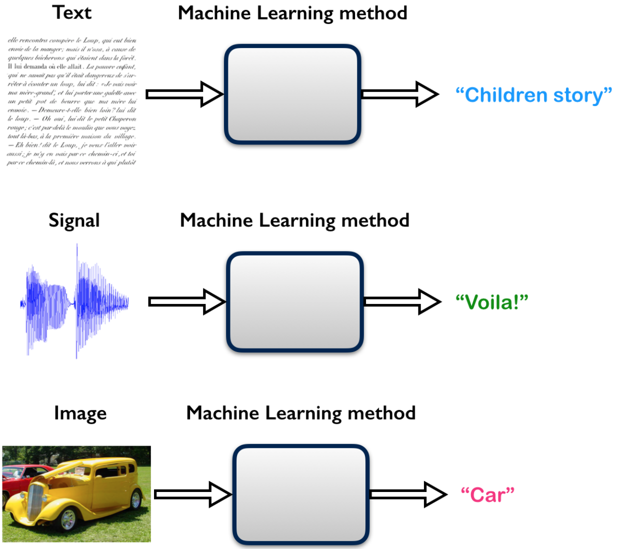
- Z. Harchaoui, F. Bach, O. Cappé, E. Moulines, *Kernel-Based Methods for Hypothesis Testing: A Unified View*, IEEE Signal Proc. Magazine, 2013.
- Z. Harchaoui, F. Bach, *Tree-walk kernels for computer vision*, in "Image Processing and Analysis with Graphs: Theory and Practice", 2012.
- F. Cucker, D.X. Zhou.*Learning Theory: An Approximation Theory*, Cambridge UP, 2007.
- J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge UP, 2004.
- B. Schölkopf, A. Smola, *Learning with Kernels*, MIT Press, 2002.

Software and Datasets

Software and Datasets

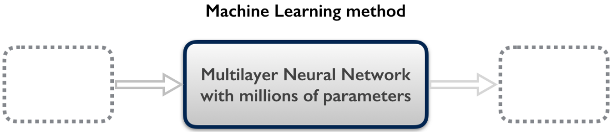
- `ckn.gforge.inria.fr/`
- `lear.inrialpes.fr/people/paulin/projects/RomePatches/`

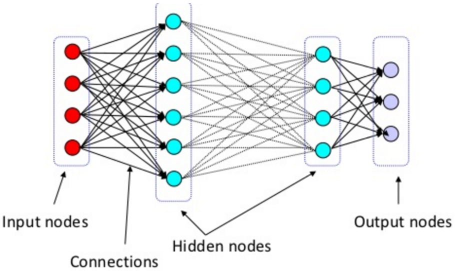
Machine Learning



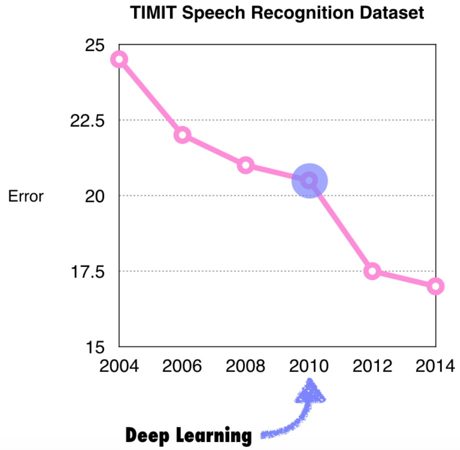
The “Deep Learning Revolution”

Deep Learning Revolution



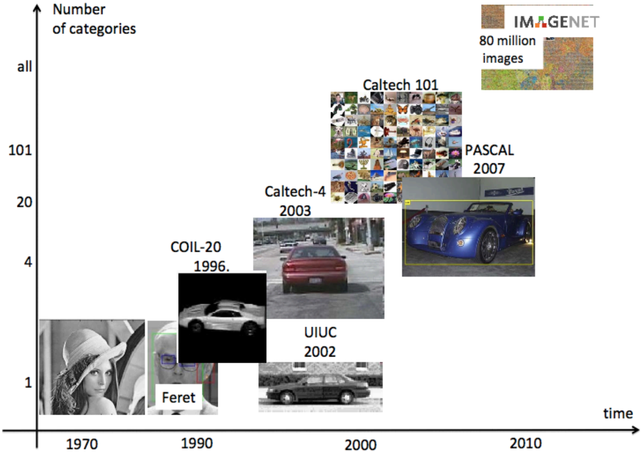


Deep Learning for Speech Recognition



Performance improvements in spoken word error rate over the years on the TIMIT acoustic-phonetic continuous speech corpus dataset.

Deep Learning for Computer Vision



From "The Promise and Perils of Benchmark Datasets and Challenges", D. Forsyth, A. Efros, F.-F. Li, A. Torralba and A. Zisserman, Talk at "Frontiers of Computer Vision"

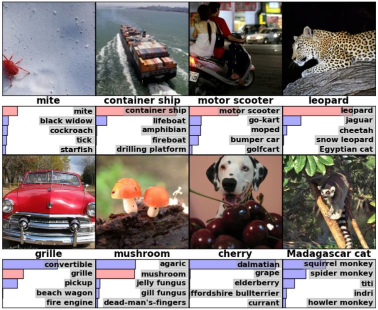
Deep Learning for Computer Vision

Labelling with crowds

Collecting data through social computing and crowdsourcing

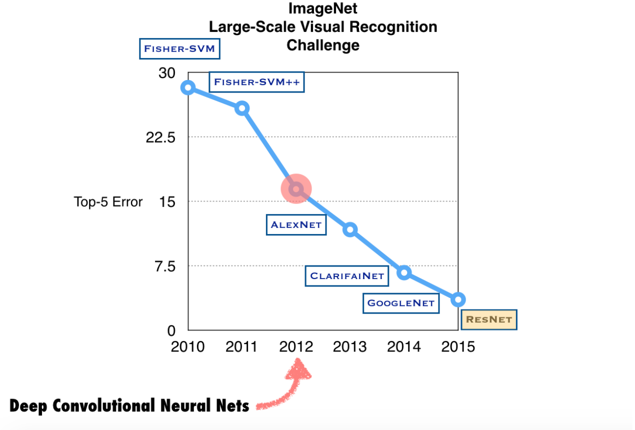
Deep Learning for Computer Vision

Deep Learning for Image Categorization



Results on ImageNet Large-scale Visual Recognition Challenge 2010.

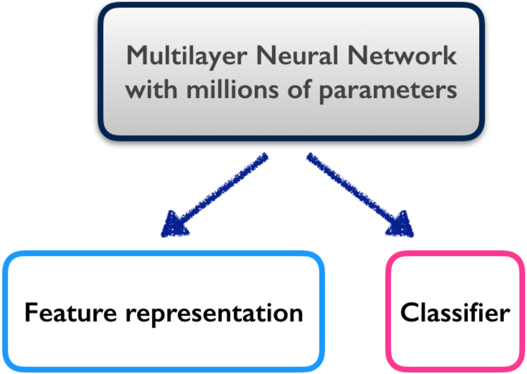
Deep Learning for Computer Vision



Performance improvements in top-5 error over the years on the ImageNet Large-scale Visual Recognition Challenge.

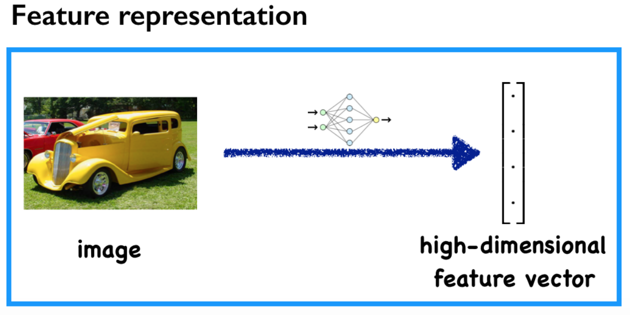
Learning feature representations

Deep Networks learn feature representations



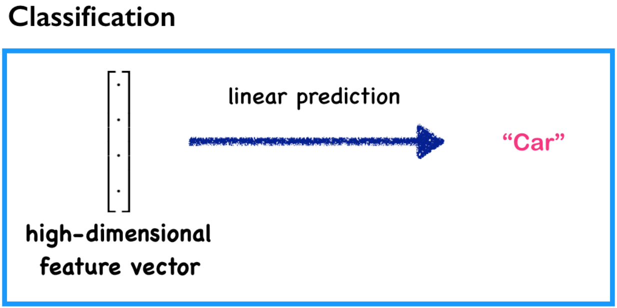
Learning feature representations

Deep Networks learn feature representations



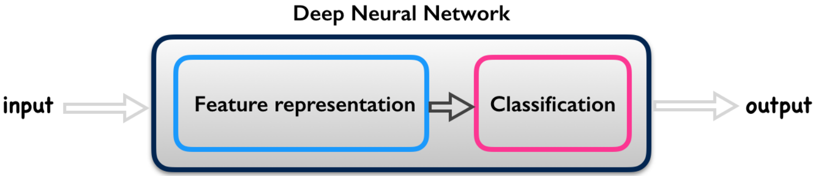
Predicting output label

Deep Networks predict output label



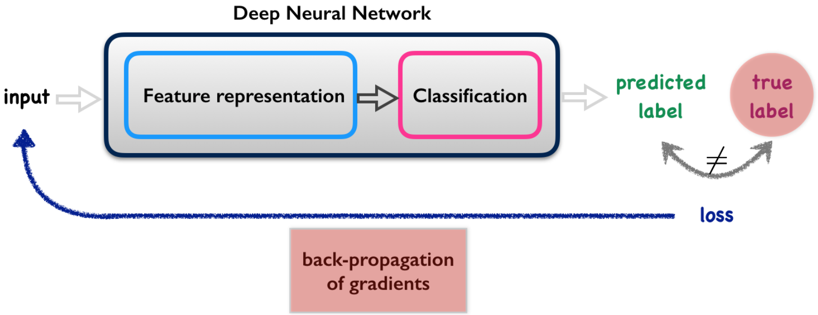
Overview of Deep Networks

Overview of Deep Networks



Training Deep Convolutional Networks

Training Deep Convolutional Networks



Deep Learning approach

Current methodology

1. Frame the task as predicting output **label** from input **example**
2. Collect a **huge training sample**
3. Train using **supervised learning** and **stochastic back-prop**
4. Done

Deep Learning approach

Current methodology

1. Frame the task as predicting output **label** from input **example**
2. Collect a **huge training sample**
3. Train using **supervised learning** and **stochastic back-prop**
4. Done

Challenges

1. Can any task be framed as a prediction task?
2. Where do I get the huge training sample?
3. Training with stochastic back-prop, is it that easy?

Framing the task as prediction

Image retrieval

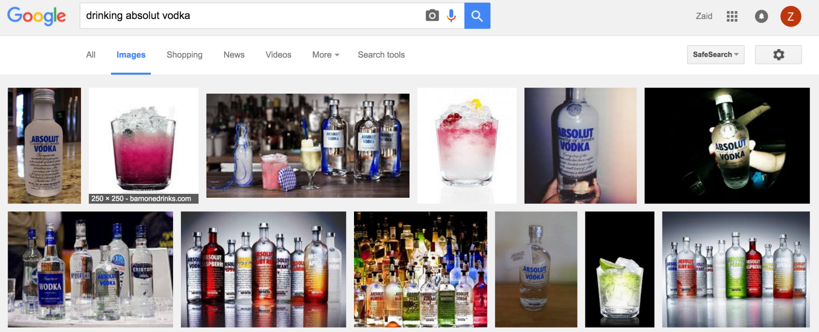


Figure: Google image search results for query “drinking absolut vodka”.

Framing the task as prediction

Limitations of reframing

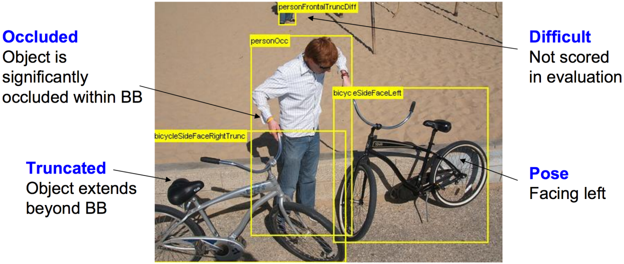
Computer vision is not a statistical problem

Car examples in ImageNet

Is this less of a car
because the context is wrong?

Figure: From Leon Bottou's keynote at ICML 2015.

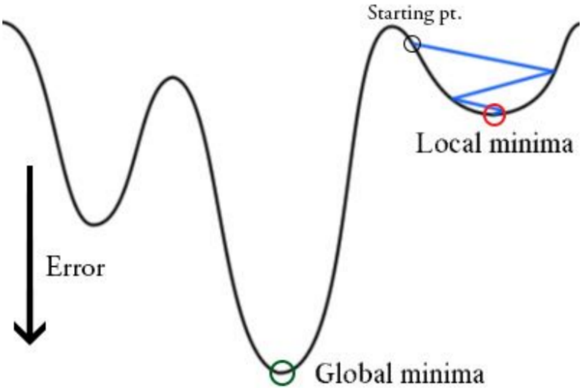
Collecting huge labelled training sample



Getting reliable human annotations is:

- time-consuming
- expensive
- often ambiguous

Training Deep ConvNets with back-prop



The wall of supervision

Current methodology

1. Frame the task as predicting output label from input example
2. Collect a huge training sample
3. Train using supervised learning and stochastic back-prop
4. Done

Current methodology hits a wall.

Yann, the cake, the icing on the cake, and the cherry

Reinforcement Learning (cherry)

- The machine predicts a scalar reward given once in a while.
- **A few bits for some samples**

Supervised Learning (icing)

- The machine predicts a category or a few numbers for each input
- **10→10,000 bits per sample**

Unsupervised Learning (cake)

- The machine predicts any part of its input for any observed part.
- Predicts future frames in videos
- **Millions of bits per sample**

From Yann LeCun's opening lecture at College de France (2016).

New Convolutional Networks

Research program

1. Clear design principle
2. Training with little or no supervision
3. Layer-by-layer training

Real-world applications

- Image denoising
- Image retrieval
- Motion estimation (optical flow)
- Music genre classification

Old and New Deep Convolutional Methods

Current Deep Convolutional Methods

1. Design principles inspired by psychophysics and neuroscience
2. Depth hubris (“the deeper, the better”)
3. Supervised learning using large amounts of labelled data
4. End-to-end training using stochastic back-propagation

Convolutional Kernel-based Methods: program

1. Clear and simple mathematical design principle
2. Concise construction
3. Training with little or no supervision
4. Layer-by-layer training using stochastic gradient optimization

New Convolutional Methods

Research program

1. Clear and simple mathematical design principle
2. Concise construction
3. Training with little or no supervision
4. Layer-by-layer training using stochastic gradient optimization

Real-world applications

- Image retrieval
- Motion estimation (optical flow)
- Music genre classification
- Epileptic seizure detection

Kernel methods

Machine Learning methods taking $\mathbf{K} = [k(X_i, X_j)]_{i,j=1,\ldots,n}$ (Gram matrix as input for processing a sample $\{X_1, \ldots, X_n\}$, where $k(x, y)$ is a similarity measure between x and y defining a positive definite kernel.

Strengths of Kernel Methods

- Minimal assumptions on data types (vectors, strings, trees, graphs, etc.)
- Interpretation of $k(x, y)$ as a dot product $k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ in a reproducing kernel Hilbert space $\mathcal{H}$ where the observations are mapped *via* $[\phi : \mathcal{X} \to \mathcal{H}]$ the feature map $\phi(\bullet) = k(\bullet, \cdot)$

See (Wahba, 1990), (Schölkopf and Smola, 2002), (Shawe-Taylor and Cristianini, 2004), (Steinwart, 2008).

Kernel methods

Positive-definite kernel

- definition: given a set of objects $\mathcal{X}$, a positive definite kernel is a symmetric function $k(x, x')$ such that for all finite sequences of $x_i \in \mathcal{X}$ and $\alpha_i \in \mathbf{R}$,

$$\sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) \geq 0 \ .$$

- Aronszajn theorem: k is a positive-definite kernel iif there exists a Hilbert space $\mathcal{H}$ and a mapping $\Phi(\cdot) : \mathcal{X} \to \mathcal{H}$ such that for any $x, x' \in \mathcal{X}$

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}} \ .$$

Kernel-based methods

Why using kernels?

- kernels may model invariance, *e.g.*, shift-invariant kernels:

$$K(\mathbf{x},\mathbf{y}) = e^{-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{y}\|^2},$$

or kernels for sets that are invariant to permutations

$$K(\mathbf{x},\mathbf{y}) = \sum_{i,j=1}^{m} k(s_i(\mathbf{x}), s_j(\mathbf{y})),$$

or kernels with limited invariance

$$K(\mathbf{x},\mathbf{y}) = \sum_{i,j=1}^{m} e^{-(i-j)^2/2\sigma^2} k(s_i(\mathbf{x}), s_j(\mathbf{y})),$$

Kernel-based methods

Why not using kernels?

- they usually require computing the $n \times n$ Gram matrix;
- kernel evaluation may be computationally expensive.
- existing kernels are rigid and not "compositional".

- a new generation of compositional kernels;
- finite-dimensional linear approximations, ie find a mapping $\psi : \mathcal{X} \to \mathbb{R}^p$ st

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

  where ψ is fast to evaluate.

See (Williams and Seeger, 2001), (Rahimi and Recht, 2007), (Vedaldi and Zisserman, 2012), (Le et al., 2013).

Kernel-based methods and Neural Nets

Infinite Neural Nets

A neural net with one hidden layer and an infinite number of hidden neurons drawn from a probability distribution p is equivalent to a kernel-based method with the equivalent kernel

$$K(\mathbf{x}, \mathbf{y}) = \int_{\omega} g(\omega^T \mathbf{x}) g(\omega^T \mathbf{y}) p(\omega) d\omega \ ,$$

where $g(\cdot)$ is nonlinear function

Example

Gaussian kernel with $p(\cdot) = \mathcal{N}\left(0, \frac{\sigma^2}{4}\mathbf{I}\right)$

$$\exp\left(-\frac{|\mathbf{x}-\mathbf{y}|_2^2}{2\sigma^2}\right) = \int_{\omega} \exp\left(\frac{2\omega^T \mathbf{x}}{\sigma^2}\right) \exp\left(\frac{2\omega^T \mathbf{y}}{\sigma^2}\right) p(\omega) d\omega \ .$$

Kernel-based methods and Neural Nets

Infinite Neural Nets

A neural net with one hidden layer and an infinite number of hidden neurons drawn from a probability distribution p is equivalent to a kernel-based method with the equivalent kernel

$$K(\mathbf{x}, \mathbf{y}) = \int_{\omega} g(\omega^T \mathbf{x}) g(\omega^T \mathbf{y}) p(\omega) d\omega \ ,$$

where $g(\cdot)$ is nonlinear function

Example

Arc-cosine kernels with $p = \mathcal{N}(0, \mathbf{I})$

$$K(\mathbf{x}, \mathbf{y}) = \int_{\omega} \left(\max(\omega^T \mathbf{x}, 0)\right)^p \left(\max(\omega^T \mathbf{y}, 0)\right)^p p(\omega) d\omega \ ,$$

for x, y such that $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$ (Cho and Saul, 2009).

Kernel-based methods and Neural Nets

Infinite Neural Nets

A neural net with one hidden layer and an infinite number of hidden neurons drawn from a probability distribution p is equivalent to a kernel-based method with the equivalent kernel

$$K(\mathbf{x}, \mathbf{y}) = \int_{\omega} g(\omega^T \mathbf{x}) g(\omega^T \mathbf{y}) p(\omega) d\omega \; .$$

- Established by (Neal, 1996) and by (Williams and Rasmussen, 1996).
- Fueled research on kernel-based methods and (Bayesian) Gaussian Process models.
- Probably contributed to the decline of neural networks in machine learning.
- Parallel results in approx. theory (Barron, 1994) and (Candes, 1995); system identification (Delyon et al., 1995), etc.

Kernel-based methods and Neural Nets

Table 3. Summary of results on the MNIST set. At 0.6% (0.56% before rounding), the system described in Section 5.1.1 performs best.

Classifier	Test err. (60k)	Test err. (10k)	Reference
3-Nearest-neighbor	—	2.4%	(LeCun et al., 1998)
2-Layer MLP	—	1.6%	(LeCun et al., 1998)
SVM	1.6%	1.4%	(Schölkopf, 1997)
Tangent distance	—	1.1%	(Simard et al., 1993) (LeCun et al., 1998)
LeNet4	—	1.1%	(LeCun et al., 1998)
LeNet4, local learning	—	1.1%	(LeCun et al., 1998)
Virtual SVM	1.0%	0.8%	(Schölkopf, 1997)
LeNet5	—	0.8%	(LeCun et al., 1998)
Dual-channel vision model	—	0.7%	(Teow and Loe, 2000)
Boosted LeNet4	—	0.7%	(LeCun et al., 1998)
Virtual SVM, 2-pixel translation	—	0.6%	*this paper; see Section 5.1.1*

Shallow and Deep methods

"Scaling Learning Algorithms towards AI", Y. Bengio and Y. LeCun

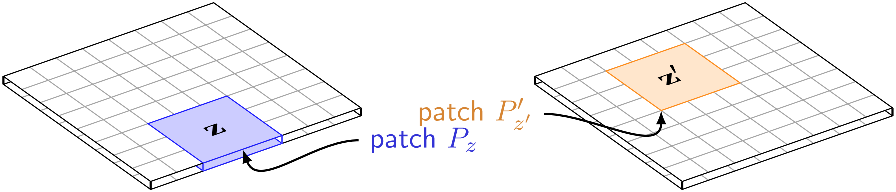
We establish a distinction between shallow architectures, and deep architectures. Shallow architectures are best exemplified by modern kernel machines, such as Support Vector Machines.

One could say that one of the main issues with kernel machine with local kernels is that they are little more than template matchers. It is possible to use kernels that are non-local yet not task-specific, such as the linear kernels and polynomial kernels. However, most practitioners have been preferring linear kernels or local kernels.

Kernel between image patches

Kernel between image patches

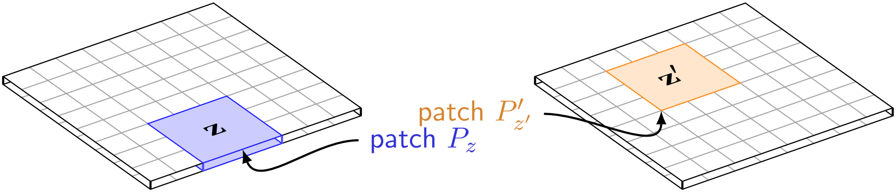
$$\kappa(P, P') = \|P\| \|P'\| e^{-\frac{1}{2\alpha^2}\|\tilde{P}-\tilde{P}'\|^2}$$



Convolutional kernel between images

Single-layer convolutional kernel

$$K(M, M') = \sum_{z,z' \in \Omega} e^{-\frac{1}{2\beta^2}\|z-z'\|^2} \underbrace{\|P\|\|P'\|e^{-\frac{1}{2\alpha^2}\|\tilde{P}-\tilde{P}'\|^2}}_{\kappa\left(P_z, P'_{z'}\right)}$$



Convolutional kernel between images

Single-layer convolutional kernel

$$K(M, M') = \sum_{z,z' \in \Omega} e^{-\frac{1}{2\beta^2}\|z-z'\|^2} \underbrace{\|P\|\|P'\| e^{-\frac{1}{2\alpha^2}\|\tilde{P}-\tilde{P}'\|^2}}_{\kappa\left(P_z, P'_{z'}\right)}$$

Main components

- **Shift-invariance** thanks to kernel $\exp(-\frac{1}{2\beta^2}\|z - z'\|^2)$
- **Matching patches** through kernel κ
- **Permutation-invariance** thanks to sum over all locations $\sum_{z,z' \in \Omega}$

Convolutional kernel between images

Single-layer convolutional kernel

$$K(M, M') = \sum_{z, z' \in \Omega} e^{-\frac{1}{2\beta^2}\|z - z'\|^2} \underbrace{\|P\|\|P'\| e^{-\frac{1}{2\alpha^2}\|\tilde{P} - \tilde{P}'\|^2}}_{\kappa\left(P_z, P'_{z'}\right)}$$

Parameters

- Parameter α controls amount of non-linearity to compare P_z and $P'_{z'}$
- Parameter β controls size of effective neighborhood in which a patch is matched with another one

Convolutional kernel between images

Main components

- **Shift-invariance** thanks to kernel $\exp(-\frac{1}{2\beta^2}\|z - z'\|^2)$
- **Matching patches** through kernel κ
- **Permutation-invariance** thanks to sum over all locations $\sum_{z,z' \in \Omega}$

Parameters

- Parameter α controls amount of non-linearity to compare P_z and $P'_{z'}$
- Parameter β controls size of effective neighborhood in which a patch is matched with another one

Positive semi-definiteness

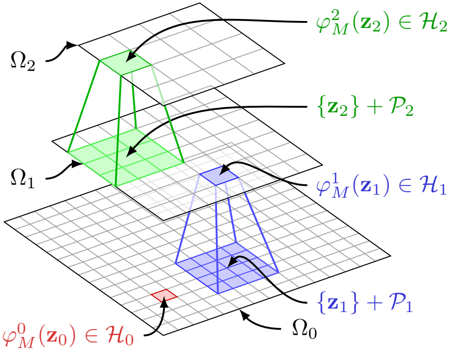
The single-layer convolutional kernel $K(\cdot, \cdot)$ is positive semi-definite.

Multi-layer convolutional kernel

Recursive construction

Assume that we managed to build k layers. We now have at hand

$$\text{a feature map } \varphi_M^k(z) \text{ for any } z \text{ in } \Omega_k$$



Multi-layer convolutional kernel

Recursive construction

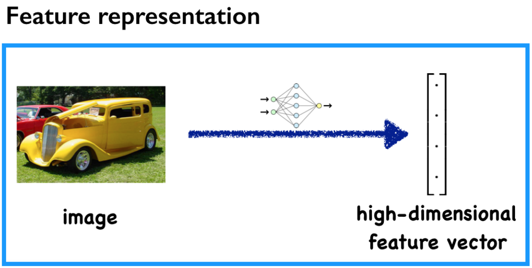
Assume that we managed to build k layers. We now have at hand

$$\text{a feature map } \varphi_M^k(z) \text{ for any } z \text{ in } \Omega_k$$

Properties of feature map

- for any z in Ω_k, the pointwise feature map $\varphi_M^k(z)$ carries information from a local neighborhood from φ_M^{k-1} centered at location z
- the feature map φ_M^k is expected to be "more invariant" than φ_M^{k-1}

Feature representation of Deep ConvNets



Design	Neuroscience-inspired
Property	Local invariance
Aesthetic	Depth

What we want

Design	**Theoretically-grounded**
Property	Local invariance
Aesthetic	**Conciseness**

Convolutional similarity

M

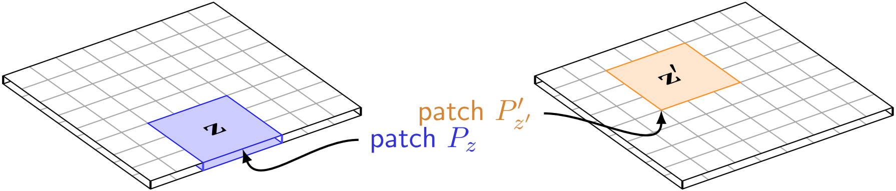
M'

similarity measure
"kernel"

Design	**Theoretically-grounded**
Property	Local invariance
Aesthetic	**Conciseness**

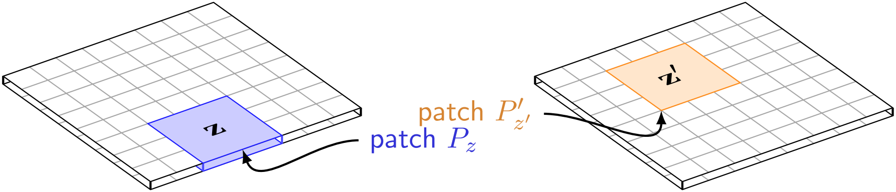
Similarity between sub-patches

$$\kappa(P, P') = \|P\|\|P'\|e^{-\frac{1}{2\alpha^2}\|\tilde{P}-\tilde{P}'\|^2}$$



Similarity between images

$$K(M, M') = \sum_{z,z' \in \Omega} e^{-\frac{1}{2\beta^2}\|z-z'\|^2} \underbrace{\|P\|\|P'\| e^{-\frac{1}{2\alpha^2}\|\tilde{P}-\tilde{P}'\|^2}}_{\kappa\left(P_z, P'_{z'}\right)}$$



Convolutional similarity between images

Single-layer convolutional similarity

$$K(M, M') = \sum_{z, z' \in \Omega} e^{-\frac{1}{2\beta^2}\|z-z'\|^2} \underbrace{\|P\|\|P'\| e^{-\frac{1}{2\alpha^2}\|\tilde{P}-\tilde{P}'\|^2}}_{\kappa\left(P_z, P'_{z'}\right)}$$

Main components

- **Shift-invariance** thanks to kernel $\exp\left(-\frac{1}{2\beta^2}\|z - z'\|^2\right)$
- **Matching patches** through kernel κ
- **Permutation-invariance** thanks to sum over all locations $\sum_{z, z' \in \Omega}$

Convolutional similarity between images

Single-layer convolutional similarity

$$K(M, M') = \sum_{z,z' \in \Omega} \underbrace{e^{-\frac{1}{2\beta^2}\|z-z'\|^2}}_{\text{kernel bw positions}} \underbrace{\|P\|\|P'\|e^{-\frac{1}{2\alpha^2}\|\tilde{P}-\tilde{P}'\|^2}}_{\text{kernel bw sub-patches}}$$

Main components

- **Shift-invariance** thanks to kernel $\exp\left(-\frac{1}{2\beta^2}\|z - z'\|^2\right)$
- **Matching patches** through kernel κ
- **Permutation-invariance** thanks to sum over all locations $\sum_{z,z' \in \Omega}$

Multi-layer convolutional similarity

Multi-layer convolutional similarity

Comparing patches from φ_M^{k-1} and $\varphi_{M'}^{k-1}$

$$\sum_{u,u' \in \Omega_{k-1}} e^{-\frac{1}{2\beta_k^2}\|u-u'\|^2} \kappa_k(\varphi_M^{k-1}(u), \varphi_M^{k-1}(u'))$$

where

$$\kappa_k(\varphi, \varphi') = \|\varphi\|_{\mathcal{H}_{k-1}} \|\varphi'\|_{\mathcal{H}_{k-1}} e^{-\frac{1}{2\alpha_k^2}\|\varphi - \varphi'\|^2_{\mathcal{H}_{k-1}}}$$

What we need

- compact approximation of φ_M to propagate through layers
- efficient scheme to recursively compute similarity measure

Multi-layer convolutional similarity

Multi-layer convolutional similarity

Comparing patches from φ_M^{k-1} and $\varphi_{M'}^{k-1}$

$$\sum_{u,u' \in \Omega_{k-1}} e^{-\frac{1}{2\beta_k^2}\|u-u'\|^2} \kappa_k(\varphi_M^{k-1}(u), \varphi_M^{k-1}(u'))$$

where

$$\kappa_k(\varphi, \varphi') = \|\varphi\|_{\mathcal{H}_{k-1}} \|\varphi'\|_{\mathcal{H}_{k-1}} \underbrace{e^{-\frac{1}{2\alpha_k^2}\|\tilde{\varphi}-\tilde{\varphi}'\|^2_{\mathcal{H}_{k-1}}}}_{\text{expensive to compute!}}$$

What we need

- compact approximation of φ_M to propagate through layers
- efficient scheme to recursively compute similarity measure

Numerical approximation of the similarity *Convolutional Kernel Net*

Step 1: explicit embedding of each layer feature representation

$$\sum_{u,u' \in \mathcal{P}_k} e^{-\frac{1}{2\beta_k^2}\|u-u'\|^2} \kappa_k(\varphi_M^{k-1}(u), \varphi_M^{k-1}(u'))$$
$$\approx \sum_{u,u' \in \mathcal{P}_k} e^{-\frac{1}{2\beta_k^2}\|u-u'\|^2} \tilde{M}_k(u)^T \tilde{M}_k(u)$$

Step 2: uniform sampling approximation of Gaussian kernel

$$\sum_{u,u' \in \mathcal{P}_k} e^{-\frac{1}{2\beta_k^2}\|u-u'\|^2} \tilde{M}_k(u)^T \tilde{M}_k(u')$$
$$\approx \frac{2}{\pi} \sum_{v \in \Omega_k} \left(\sum_{u \in \mathcal{P}_k} e^{-\frac{1}{2\beta_k^2}\|u-v\|^2} \tilde{M}_k(u) \right)^T \left(\sum_{u' \in \mathcal{P}_k} e^{-\frac{1}{2\beta_k^2}\|u'-v\|^2} \tilde{M}_k(u') \right)$$

Key idea

Finite-dimensional approximation (aka explicit embedding)

For all $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^q$,

$$e^{-\frac{1}{2\alpha^2}\|\mathbf{x}-\mathbf{y}\|_2^2} \approx \sum_{j=1}^{p} f_j(\mathbf{x}) f_j(\mathbf{y})$$

Division of Gaussian kernel in the convolution sense

We have the relation

$$e^{-\frac{1}{2\alpha^2}\|\mathbf{x}-\mathbf{y}\|_2^2} = \left(\frac{2}{\pi\alpha^2}\right)^{\frac{q}{2}} \int_{\mathbf{z}\in\mathbb{R}^q} e^{-\frac{1}{\alpha^2}\|\mathbf{x}-\mathbf{z}\|_2^2} e^{-\frac{1}{\alpha^2}\|\mathbf{y}-\mathbf{z}\|_2^2} d\mathbf{z}$$

Key idea

Factorization of Gaussian kernel in the convolution sense

We have the relation

$$e^{-\frac{1}{2\alpha^2}\|\mathbf{x}-\mathbf{y}\|_2^2} = \left(\frac{2}{\pi\alpha^2}\right)^{\frac{q}{2}} \int_{\mathbf{z}\in\mathbb{R}^q} e^{-\frac{1}{\alpha^2}\|\mathbf{x}-\mathbf{z}\|_2^2} e^{-\frac{1}{\alpha^2}\|\mathbf{y}-\mathbf{z}\|_2^2} d\mathbf{z}$$

Possible strategies

- Monte-Carlo approximation $\rightarrow$ random Fourier features
- Integral quadrature $\rightarrow$ Nyström approximation, kernel herding
- k-means, matrix factorization.
- Direct optimization

Key idea

Factorization of Gaussian kernel in the convolution sense

We have the relation

$$e^{-\frac{1}{2\alpha^2}\|\mathbf{x}-\mathbf{y}\|_2^2} = \left(\frac{2}{\pi\alpha^2}\right)^{\frac{q}{2}} \int_{\mathbf{z}\in\mathbb{R}^q} e^{-\frac{1}{\alpha^2}\|\mathbf{x}-\mathbf{z}\|_2^2} e^{-\frac{1}{\alpha^2}\|\mathbf{y}-\mathbf{z}\|_2^2} d\mathbf{z}$$

Direct Optimization

1. Initialize with k-means
2. Minimize using randomized incremental gradient method

$$\min_{W,b} \quad \mathbb{E}_{x,x'\sim\mathbb{P}_X}\left[e^{\frac{\|x-x'\|^2}{2\alpha^2}} - \sum_{j=1}^{p} e^{w_j^\top x+b_j} e^{w_j^\top x'+b_j}\right]^2$$

Convolutional Kernet Nets

Overview

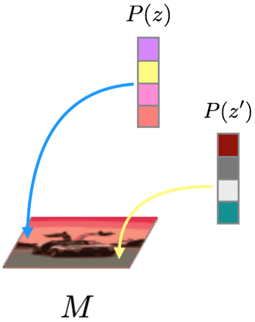
Convolution $W_k^T P_k(z)$

Exponential Nonlinearity $\tilde{M}_k(z) = \|P_k(z)\| \exp\left(W_k^T P_k(z) + b_k\right)$

Gaussian Pooling $M_k(z) = \sum_{u \in \Omega_{k-1}} \exp\left(-\frac{1}{\beta_k^2}\|u - z\|^2\right) \tilde{M}_k(u)$

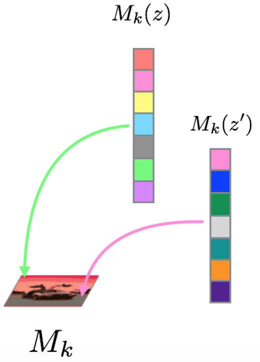
Convolutional Kernet Nets

Zoom on zero-th layer



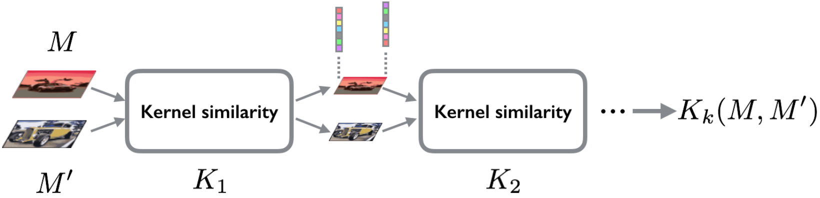
Convolutional Kernel Nets

Zoom on k-th layer



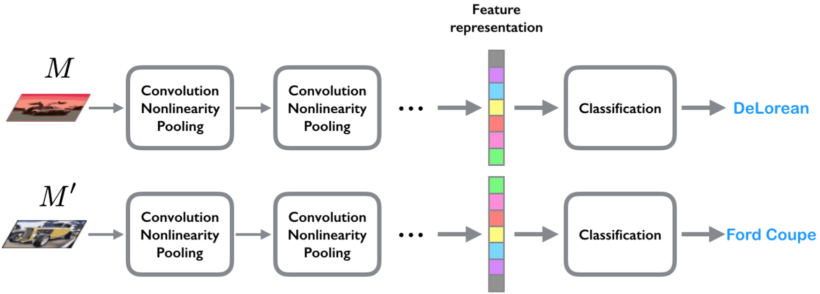
Convolutional Kernel Nets vs Standard ConvNets

Convolutional Kernel Nets



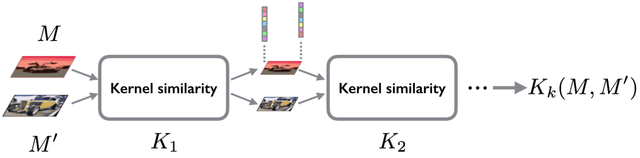
Convolutional Kernel Nets vs Standard ConvNets

Standard ConvNets

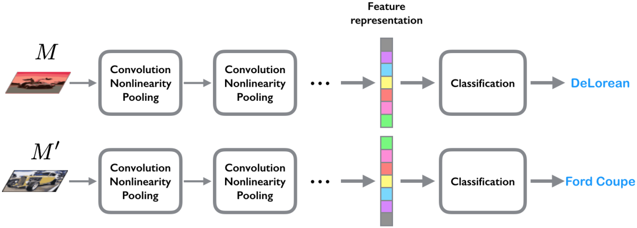


Convolutional Kernel Nets vs Standard ConvNets

Convolutional Kernel Nets



Standard ConvNets

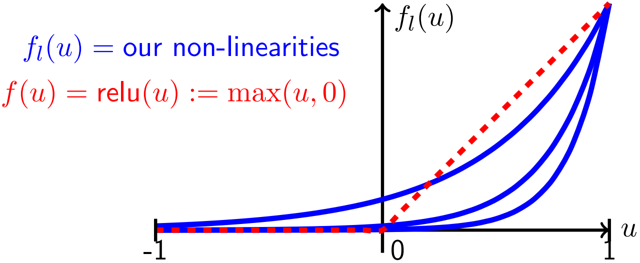


Exponential non-linear units

When applying the mapping to unit-norm vectors $\mathbf{x}$, we may write

$$\mathbf{x} \mapsto [f_l(\mathbf{w}_l^\top \mathbf{x}) := \sqrt{\eta_l} e^{-(1/\sigma^2)\|\mathbf{x}-\mathbf{w}_l\|_2^2}]_{l=1}^p,$$

and when the $\mathbf{x}$'s are patches from an image, the inner-product $\mathbf{w}_l^\top \mathbf{x}$ are simply convolutions, and the functions f_l pointwise non-linearities.



Algorithm 1 Training layer k of a CKN.

HYPER-PARAMETERS: Kernel parameter α_k, patch size $e_k \times e_k$, number of filters p_k.

INPUT MODEL: A CKN trained up to layer k–1.

INPUT DATA: A set of training images.

ALGORITHM:

- Encode the input images using the CKN up to layer k–1;
- Extract randomly n pairs of patches (P_i, P_i') from the maps obtained at layer k–1;
- Normalize the patches to make then unit-norm;
- Learn the model parameters by direct stochastic optimization, with $(x_i, x_i') = (P_i, P_i')$ for all $i = 1, \dots, n$;

OUTPUT: Weight matrix W_k in $\mathbb{R}^{p_{k-1}e_k^2 \times p_k}$ and b_k in $\mathbb{R}^{p_k}$.

Algorithm 2 Encoding layer k of a CKN.

HYPER-PARAMETERS: Kernel parameter β_k;
INPUT MODEL: CKN parameters learned from layer 1 to k;
INPUT DATA: A map $M_{k-1} : \Omega_{k-1} \to \mathbb{R}^{p_{k-1}}$;
ALGORITHM:

- Extract patches $\{P_{k,z}\}_{z \in \Omega_{k-1}}$ of size $e_k \times e_k$ from the input map M_{k-1};
- Compute contrast-normalized patches

$$\tilde{P}_{k,z} = \frac{1}{\|P_{k,z}\|} P_{k,z} \quad \text{if} \quad P_{k,z} \neq 0 \quad \text{and} \quad 0 \quad \text{otherwise.}$$

- Produce an intermediate map $\tilde{M}_k : \Omega_{k-1} \to \mathbb{R}^{p_k}$ with linear operations followed by non-linearity:

$$\tilde{M}_k(z) = \|P_{k,z}\| e^{W_k^\top \tilde{P}_{k,z} + b_k}, \tag{13}$$

where the exponential function is meant "pointwise".
- Produce the output map M_k by linear pooling with Gaussian weights:

$$M_k(z) = \sum_{u \in \Omega_{k-1}} e^{-\frac{1}{\beta_k^2}\|u-z\|^2} \tilde{M}_k(u).$$

Algorithm 1 Training layer k of a CKN.

HYPER-PARAMETERS: Kernel parameter α_k, patch size $e_k \times e_k$, number of filters p_k.
INPUT MODEL: A CKN trained up to layer k–1.
INPUT DATA: A set of training images.
ALGORITHM:

- Encode the input images using the CKN up to layer k–1;
- Extract randomly n pairs of patches (P_i, P_i') from the maps obtained at layer k–1;
- Normalize the patches to make then unit-norm;
- Learn the model parameters by direct stochastic optimization, with $(x_i, x_i') = (P_i, P_i')$ for all $i = 1, \ldots, n$;

OUTPUT: Weight matrix W_k in $\mathbb{R}^{p_{k-1}e_k^2 \times p_k}$ and b_k in $\mathbb{R}^{p_k}$.

Algorithm 2 Encoding layer k of a CKN.

HYPER-PARAMETERS: Kernel parameter β_k;

INPUT MODEL: CKN parameters learned from layer 1 to k;

INPUT DATA: A map $M_{k-1} : \Omega_{k-1} \to \mathbb{R}^{p_{k-1}}$;

ALGORITHM:

- Extract patches $\{P_{k,z}\}_{z \in \Omega_{k-1}}$ of size $e_k \times e_k$ from the input map M_{k-1};
- Compute contrast-normalized patches

$$\tilde{P}_{k,z} = \frac{1}{\|P_{k,z}\|} P_{k,z} \quad \text{if} \quad P_{k,z} \neq 0 \quad \text{and} \quad 0 \quad \text{otherwise.}$$

- Produce an intermediate map $\tilde{M}_k : \Omega_{k-1} \to \mathbb{R}^{p_k}$ with linear operations followed by non-linearity:

$$\tilde{M}_k(z) = \|P_{k,z}\| e^{W_k^\top \tilde{P}_{k,z} + b_k}, \tag{13}$$

where the exponential function is meant "pointwise".

- Produce the output map M_k by linear pooling with Gaussian weights:

$$M_k(z) = \sum_{u \in \Omega_{k-1}} e^{-\frac{1}{\beta_k^2}\|u-z\|^2} \tilde{M}_k(u).$$

Experiments

First experiment on natural image patches.

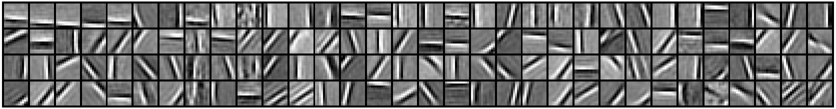


Figure: Filters obtained by the first layer of the convolutional kernel network on natural images. Database of $300,000$ whitened natural image patches of size 12×12 and learn $p = 256$ filters.

Experiments

Simple experiments on MNIST, CIFAR-10, STL-10 conducted **without data augmentation or data pre-processing**;

Tr. size	CNN	Scat-1	Scat-2	CKN-GM1 (12/50)	CKN-GM2 (12/400)	CKN-PM1 (200)	CKN-PM2 (50/200)	[32]	[18]	[19]
300	7.18	4.7	5.6	4.39	4.24	5.98	**4.15**	NA		
$1K$	3.21	2.3	2.6	2.60	**2.05**	3.23	2.76	NA		
$2K$	2.53	**1.3**	1.8	1.85	1.51	1.97	2.28	NA		
$5K$	1.52	**1.03**	1.4	1.41	1.21	1.41	1.56	NA		
$10K$	0.85	**0.88**	1	1.17	**0.88**	1.18	1.10	NA		
$20K$	0.76	0.79	**0.58**	0.89	0.60	0.83	0.77	NA		
$40K$	0.65	0.74	0.53	0.68	**0.51**	0.64	0.58	NA		
$60K$	0.53	0.70	0.4	0.58	**0.39**	0.63	0.53	0.47	0.45	0.53

Table: Test error in % for various approaches on the MNIST dataset.

Method	CoatesNg	SohnLee	MOut	SPN	Zeiler-Fergus	CKN-GM	CKN-PM	CKN-CO
CIFAR-10	82.0	82.2	**88.32**	83.96	84.87	74.84	78.30	82.18
STL-10	60.1	58.7	NA	62.3	NA	60.04	60.25	62.32

Image retrieval

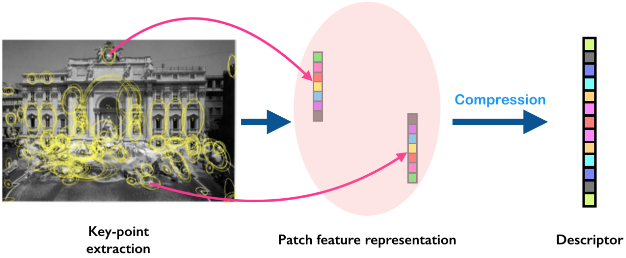


Figure: Image retrieval pipeline

State-of-the-art and evaluation

Evaluation

Benchmark datasets with images of many different scenes, for which lots of image views are available.

For each dataset, a subset of images are defined as **queries**. Performance is measured in mean average precision (mAP).

	Oxford	UKB	Holidays
# scenes	1000	10200	500
# views per scene	5	4	1500

State-of-the-art and evaluation

Supervised training in Rome-Patches

We have **patch-level** annotations on **Rome-Patches**.

- PhilippNet: supervised training with surrogate classes
- AlexNet: supervised training on ImageNet + fine-tuning with surrogate classes

Unsupervised training in Rome-Patches

- Convolutional Kernel Net (CKN): un-supervised training with random pairs of patches

Results on image retrieval

	Oxford	UKB	Holidays
SIFT	43.7	3.44	64.0
AlexNet	34.3	3.74	79.3
PhilippNet	43.6	3.67	74.7
CKN	**49.8**	**3.80**	**79.3**

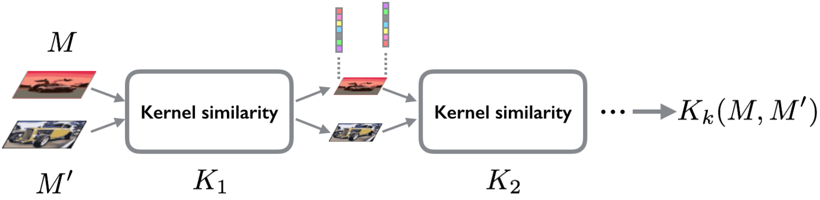
Supervised approaches, trained on Rome:
- **AlexNet**
- **PhilippNet**

Un-supervised approaches:
- **CKN**
- **SIFT**

CKN: Descriptor based on two-layer Convolutional Kernel Nets

Results in Mean Average Precision on the benchmark datasets *Oxford*, *UKB*, and *Holidays*.

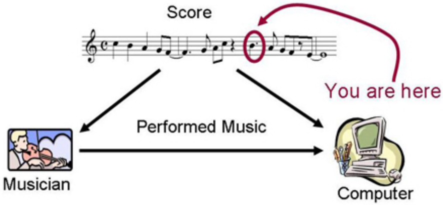
A new hope



- different approach to design **similarity** between signals
- simpler and trainable in an **unsupervised manner**
- competes with **standard ConvNets trained with supervision**
- further improvable using **supervised learning** (Mairal, 2016)

Feature representations of general data

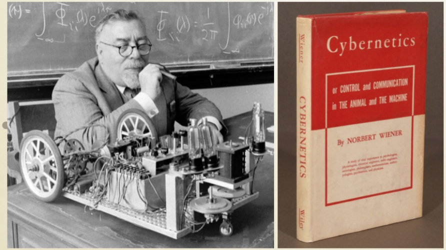
Feature representations of general data



- learning feature representations for general data (videos, music, text)
- statistical analysis of learning local invariances
- never-ending learning of feature representations from streams of data

Safety of machine-learning-based AI systems

Norbert Wiener



Norbert Wiener, 1894-1964

1947

Safety of machine-learning-based AI systems

Norbert Wiener

"Again and I again I have heard the statement that learning machines cannot subject us to any dangers, because we can turn them off when we feel like it. But can we? To turn a machine off effectively, we must be in possession of information as to whether the danger point has come. [...] The very speed of operation of modern digital machines stands in the way of our ability to perceive and think through the indications of the danger."

Safety of machine-learning-based AI systems

Safety of learning-based AI systems

- how can we certify the robustness of a ML-based AI system?
- how can safely unleash ML-based AI systems in the wild?

Final words

We need more theory!

Final words

Thank you for your attention.