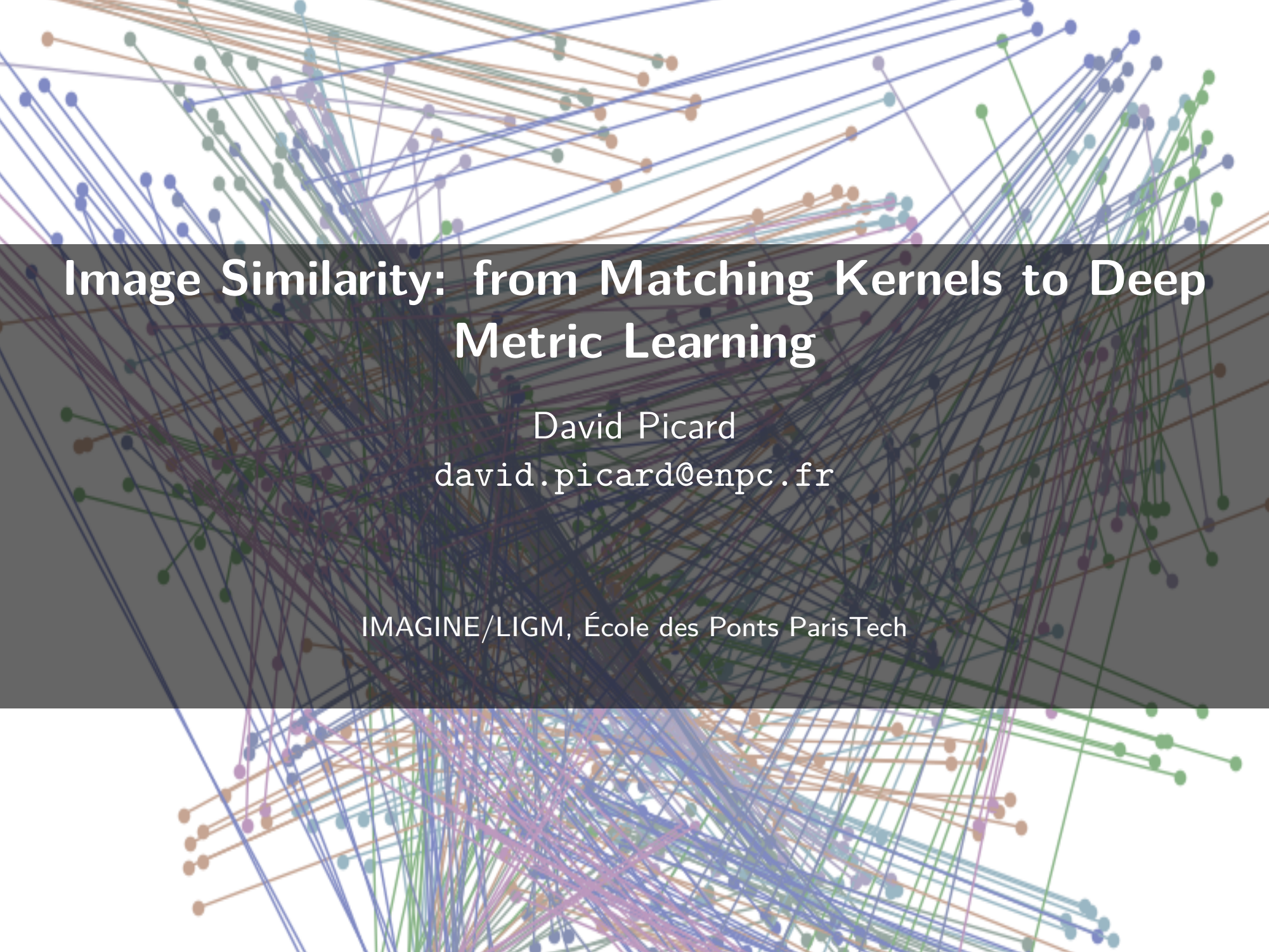


Image Similarity: from Matching Kernels to Deep Metric Learning

David Picard
david.picard@enpc.fr

IMAGINE/LIGM, École des Ponts ParisTech

Outline

EMD

$$\min \sum w_{ij} d(x_i, x_j)$$

$$w_{ij} \geq 0, \sum_i w_{ij} = 1, \sum_j w_{ij} = 1$$

Wasserstein

MMD

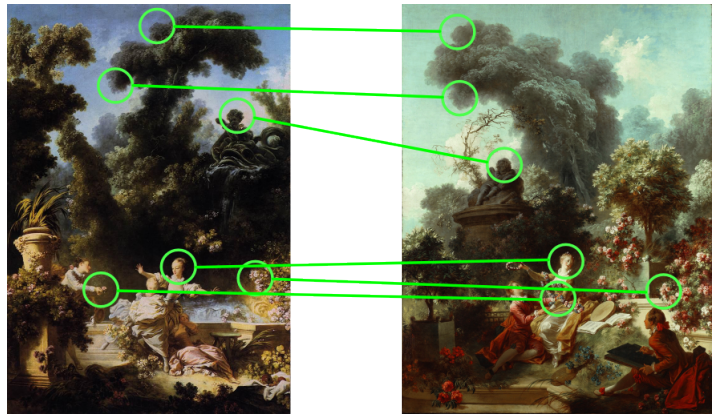
$$\sup_T \mathbb{E}_p[T(x)] - \mathbb{E}_q[T(x)]$$

HORDE $\mathbb{E}_{x \sim \mathcal{J}}[x^{\otimes k}]$

Matching Kernels

$(\approx \langle \phi(B_i), \phi(B_j) \rangle)$

Keypoints matching



Outline

EMD

$$\min \sum w_{ij} d(x_i, x_j)$$

$$w_{ij} \geq 0, \sum_i w_{ij} = 1, \sum_j w_{ij} = 1$$

Wasserstein

MMD

$$\sup_T \mathbb{E}_p[T(x)] - \mathbb{E}_q[T(x)]$$

HORDE $\mathbb{E}_{x \sim \mathscr{S}}[x^{\otimes k}]$

Matching Kernels

$(\approx \langle \phi(B_i), \phi(B_j) \rangle)$

Keypoints matching

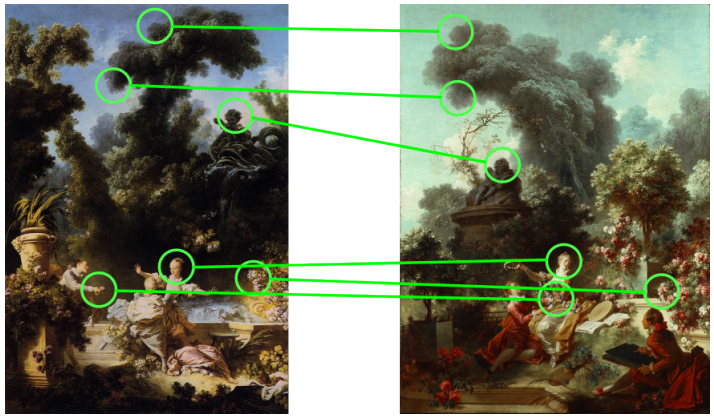
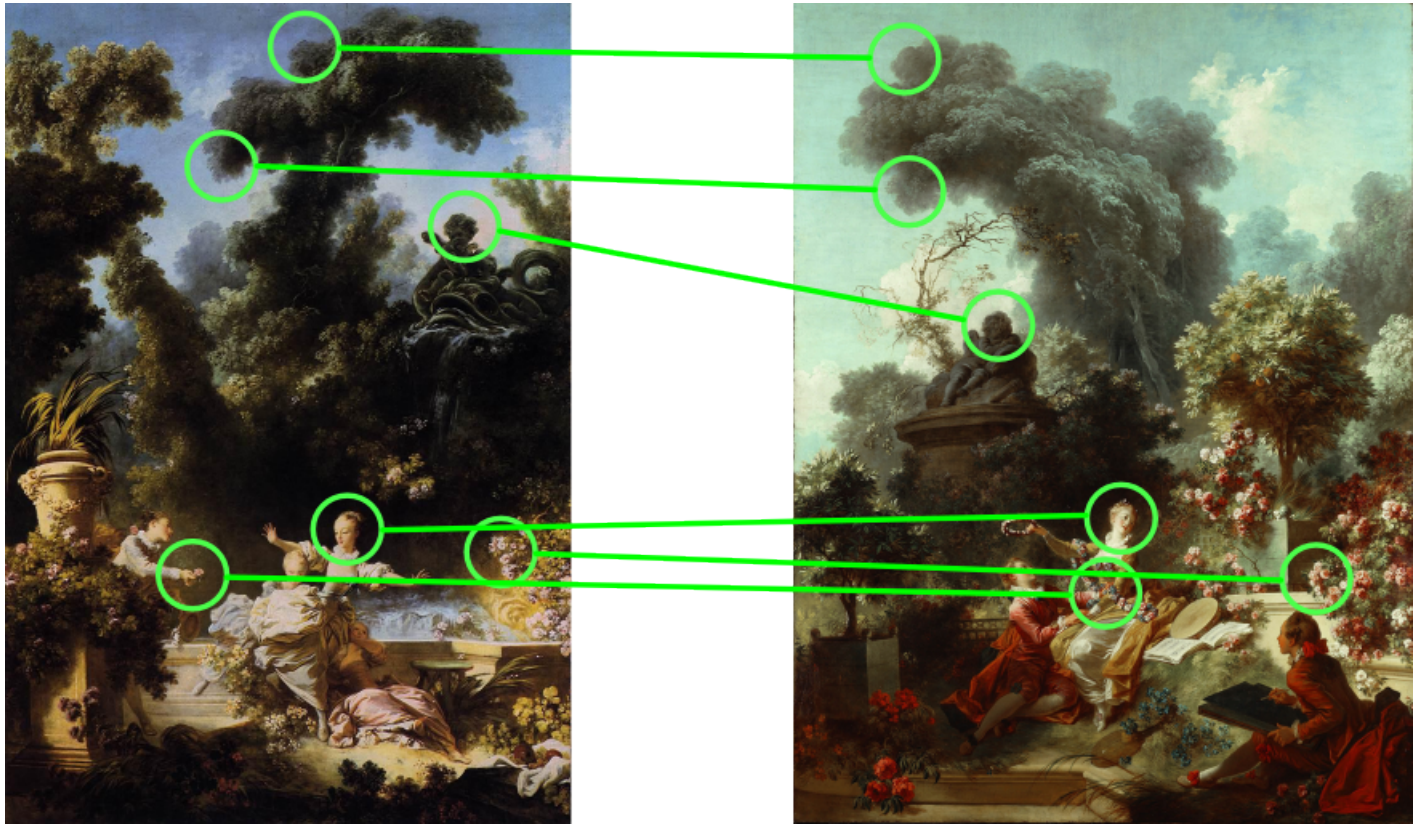


Image similarity using keypoints

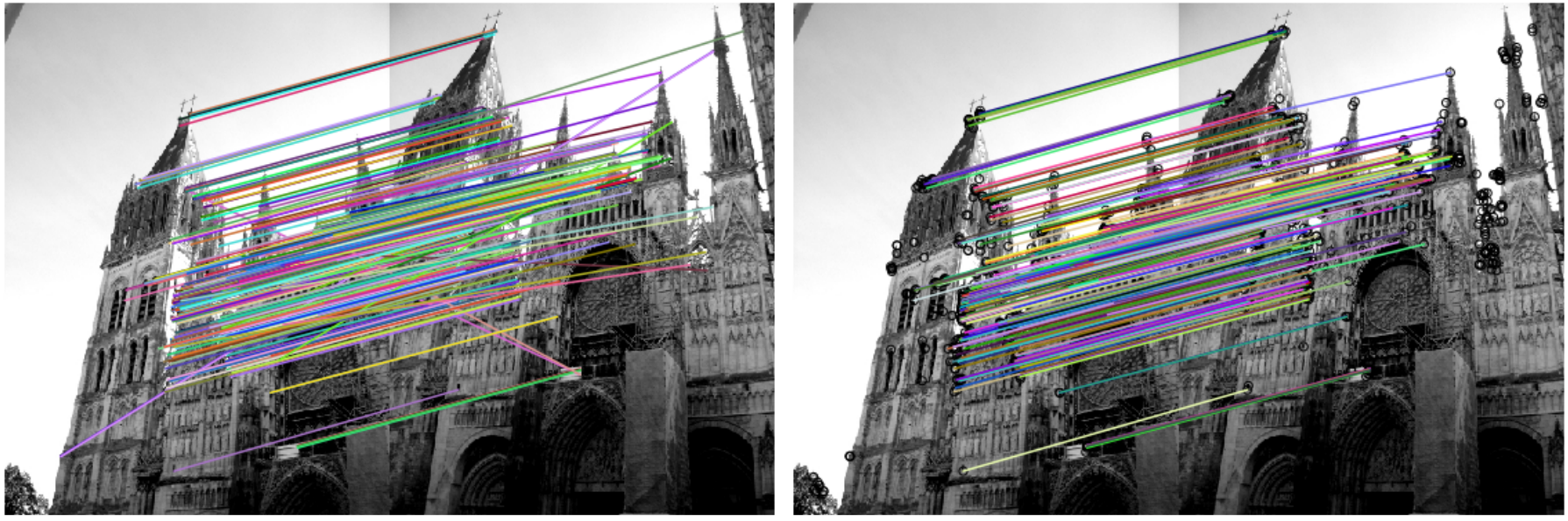
Principle: Visualy similar image ⇒ Many similar regions



- Find informative regions
- Describe informative regions (map to a vector space)
- Count number of matching regions (*e.g.*, 1-NN + distance threshold)

Geometric consistency

Keypoint matching: reconstruction methods ⇒ Preserving geometry



Use consensus algorithm to remove outliers (*e.g.*, ransac)

Generic image similarity

- Different views → different scenes
- Local geometric consistency (*e.g.*, inside an object)

Keypoint matching limitations

Keypoints matching does not correspond to an inner product

- Not symmetric

Limited use for machine learning

- No intrinsic representation
- No classifier, regressor, etc

Limited use for downstream tasks (detection, recognition)

Caveat: Recent deep neural networks approaches to learn correspondances. *E.g.*, S2DNet: Learning Accurate Correspondences for Sparse-to-Dense Feature Matching. Hugo Germain, Guillaume Bourmaud, and Vincent Lepetit. In Proceedings of the European Conference on Computer Vision (ECCV), 2020.

Outline

EMD

$\min \sum w_{ij} d(x_i, x_j)$

$w_{ij} \geq 0, \sum_i w_{ij} = 1, \sum_j w_{ij} = 1$

Wasserstein

MMD

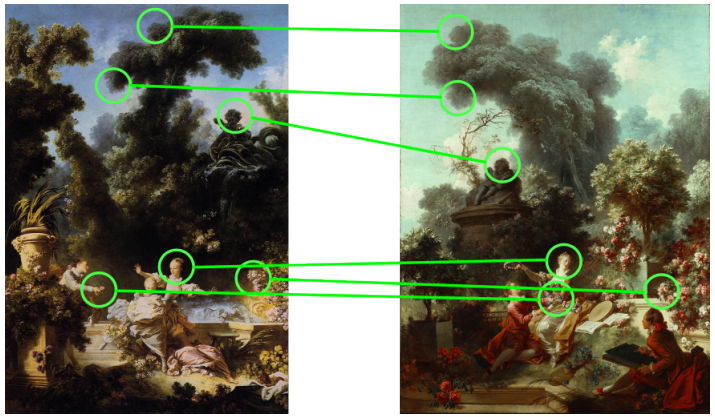
$\sup_T \mathbb{E}_p[T(x)] - \mathbb{E}_q[T(x)]$

HORDE $\mathbb{E}_{x \sim \mathcal{S}}[x^{\otimes k}]$

Matching Kernels

$(\approx \langle \phi(B_i), \phi(B_j) \rangle)$

Keypoints matching



Kernel on bags

Image := unordered set of descriptors $B_i = \{x_{ri}\}$

$\Rightarrow$ Similarity on unordered sets (**bags**)

Kernel on bags

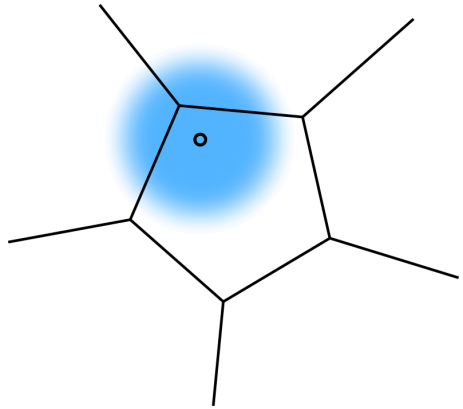


Image := unordered set of descriptors $B_i = \{x_{ri}\}$

$\Rightarrow$ Similarity on unordered sets (**bags**)

Define similarity between descriptors (**minor kernel**):

- ex: dot product $k(x_{ri}, x_{sj}) = \langle x_{ri}, x_{sj} \rangle$
- ex: Gaussian kernel : $k(x_{ri}, x_{sj}) = \exp(-\gamma \| x_{ri} - x_{sj} \|^2)$
- Implicit mapping: $k(x_{ri}, x_{sj}) = \langle \phi(x_{ri}), \phi(x_{sj}) \rangle$

Kernel on bags

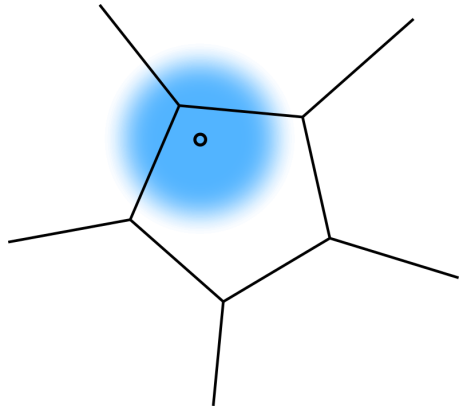


Image := unordered set of descriptors $B_i = \{x_{ri}\}$

$\Rightarrow$ Similarity on unordered sets (**bags**)

Define similarity between descriptors (**minor kernel**):

- ex: dot product $k(x_{ri}, x_{sj}) = \langle x_{ri}, x_{sj} \rangle$
- ex: Gaussian kernel : $k(x_{ri}, x_{sj}) = \exp(-\gamma \| x_{ri} - x_{sj} \|^2)$
- Implicit mapping: $k(x_{ri}, x_{sj}) = \langle \phi(x_{ri}), \phi(x_{sj}) \rangle$

Kernel on Bags: **kernel combination**

- Sum Kernel (Shawe-Taylor 2004)

$$K(B_i, B_j) = \sum_{x_{ri} \in B_i} \sum_{x_{sj} \in B_j} k(x_{ri}, x_{sj})$$

$$K(B_i, B_j) = \left\langle \sum_{x_{ri} \in B_i} \phi(x_{ri}), \sum_{x_{sj} \in B_j} \phi(x_{sj}) \right\rangle$$

Quantization kernel

Dictionary (codebook) of prototypical descriptors $\mathscr{D} = \{\mu_c\}_{1 \leq c \leq M}$

- k-means clustering over many descriptors
- Centroïds $\mu_c := \mathbf{codewords}$

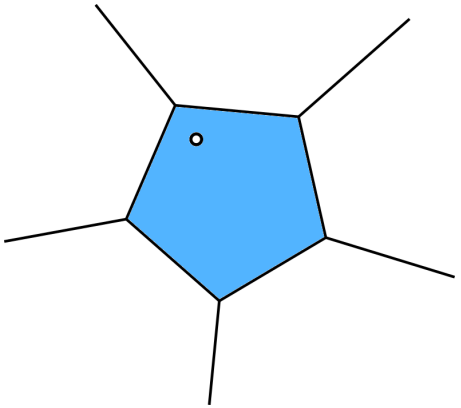
Quantization kernel

Dictionary (codebook) of prototypical descriptors $\mathscr{D} = \{\mu_c\}_{1 \leq c \leq M}$

- k-means clustering over many descriptors
- Centroïds $\mu_c := \textbf{codewords}$

Quantization function h:

- $h_m(x_{ri}) = \begin{cases} 1 \text{ if } m = \arg\min_c \|x_{ri} - \mu_c\|^2, \\ 0 \text{ else} \end{cases}$
- **1-hot assignment** to clusters ($\in \mathbb{R}^M$)



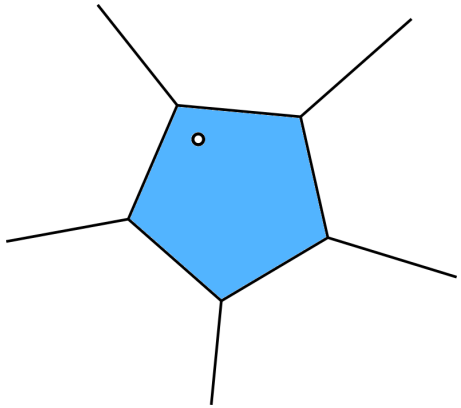
Quantization kernel

Dictionary (codebook) of prototypical descriptors $\mathscr{D} = \{\mu_c\}_{1 \leq c \leq M}$

- k-means clustering over many descriptors
- Centroïds $\mu_c := \textbf{codewords}$

Quantization function h:

- $h_m(x_{ri}) = \begin{cases} 1 \text{ if } m = \arg\min_c \|x_{ri} - \mu_c\|^2, \\ 0 \text{ else} \end{cases}$
- **1-hot assignment** to clusters $(\in \mathbb{R}^M)$



Bag of Visual Words:

$$K(B_i, B_j) = \sum_{x_{ri} \in B_i} \sum_{x_{sj} \in B_j} \sum_m h_m(x_{ri}) h_m(x_{sj})$$

$$= \langle \phi_q(B_i), \phi_q(B_j) \rangle, \quad \phi_q(B) = \sum_{x_r \in B} [h_m(x_r)]_m$$

- Histogram of occurrences of codewords (Sivic 03, Csurka 04)

Need for kernel linearization

Kernel on bags: ML compatible alternative to descriptor matching

- Implicit definition of an **image representation space**
- Choice of minor kernel ⇒ combination (sum, product)

Need for kernel linearization

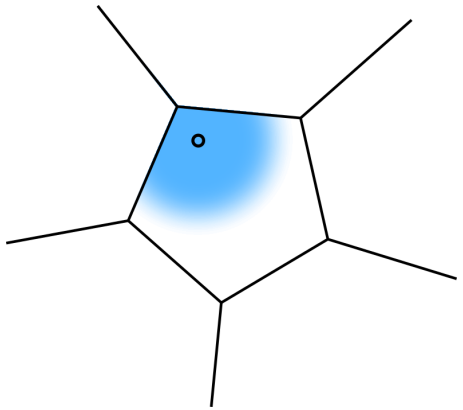
Kernel on bags: ML compatible alternative to descriptor matching

- Implicit definition of an **image representation space**
- Choice of minor kernel ⇒ combination (sum, product)

Gaussian-Quantization product kernel:

$$k(x_{ri}, x_{sj}) = k_q(x_{ri}, x_{sj}) k_g(x_{ri}, x_{sj})$$

$$K(B_i, B_j) = \sum_{x_{ri} \in B_i} \sum_{x_{sj} \in B_j} k_q(x_{ri}, x_{sj}) k_g(x_{ri}, x_{sj})$$



Need for kernel linearization

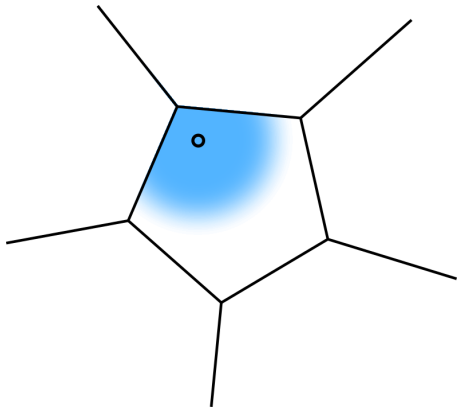
Kernel on bags: ML compatible alternative to descriptor matching

- Implicit definition of an **image representation space**
- Choice of minor kernel ⇒ combination (sum, product)

Gaussian-Quantization product kernel:

$$k(x_{ri}, x_{sj}) = k_q(x_{ri}, x_{sj}) k_g(x_{ri}, x_{sj})$$

$$K(B_i, B_j) = \sum_{x_{ri} \in B_i} \sum_{x_{sj} \in B_j} k_q(x_{ri}, x_{sj}) k_g(x_{ri}, x_{sj})$$



Computational cost:

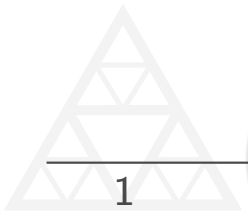
- Quadratic in number of descriptors
- Non-linear (costly) inner loop

We have to linearize → Explicit image representation space

Vector of Locally Aggregated Tensors

Gaussian-quantization kernel linearization:

- Assume $\|x_{ri}\| = 1$, then $k_g(x_{ri}, x_{sj}) \propto \exp(\gamma \langle x_{ri}, x_{sj} \rangle)$



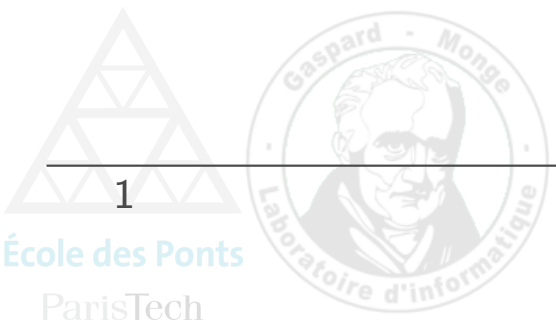
Vector of Locally Aggregated Tensors

Gaussian-quantization kernel linearization:

- Assume $\|x_{ri}\| = 1$, then $k_g(x_{ri}, x_{sj}) \propto \exp(\gamma \langle x_{ri}, x_{sj} \rangle)$

Taylor expansion (using Kronecker product $\otimes$):

- $\exp(\gamma \langle x_{ri}, x_{sj} \rangle) = \sum_p \frac{\gamma^p \langle x_{ri}, x_{sj} \rangle^p}{p!} = \sum_p \frac{\gamma^p}{p!} \langle x_{ri}^{\otimes p}, x_{sj}^{\otimes p} \rangle$
- Approximate explicit mapping: $\phi_g(x_{ri}) = [x_{ri}, x_{ri} \otimes x_{ri}, \ldots, x_{ri}^{\otimes p}]$



Vector of Locally Aggregated Tensors

Gaussian-quantization kernel linearization:

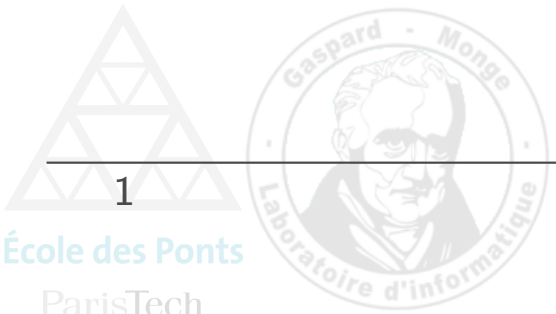
- Assume $\|x_{ri}\| = 1$, then $k_g(x_{ri}, x_{sj}) \propto \exp(\gamma \langle x_{ri}, x_{sj} \rangle)$

Taylor expansion (using Kronecker product $\otimes$):

- $\exp(\gamma \langle x_{ri}, x_{sj} \rangle) = \sum_p \frac{\gamma^p \langle x_{ri}, x_{sj} \rangle^p}{p!} = \sum_p \frac{\gamma^p}{p!} \langle x_{ri}^{\otimes p}, x_{sj}^{\otimes p} \rangle$
- Approximate explicit mapping: $\phi_g(x_{ri}) = [x_{ri}, x_{ri} \otimes x_{ri}, \ldots, x_{ri}^{\otimes p}]$

Minor kernel:

- $\phi(x_{ri}) = h(x_{ri}) \otimes \phi_g(x_{ri})$
- $k(x_{ri}, x_{sj}) = \langle h(x_{ri}) \otimes \phi_g(x_{ri}), h(x_{sj}) \otimes \phi_g(x_{sj}) \rangle$



Vector of Locally Aggregated Tensors

Gaussian-quantization kernel linearization:

- Assume $\|x_{ri}\| = 1$, then $k_g(x_{ri}, x_{sj}) \propto \exp(\gamma \langle x_{ri}, x_{sj} \rangle)$

Taylor expansion (using Kronecker product $\otimes$):

- $\exp(\gamma \langle x_{ri}, x_{sj} \rangle) = \sum_p \frac{\gamma^p \langle x_{ri}, x_{sj} \rangle^p}{p!} = \sum_p \frac{\gamma^p}{p!} \langle x_{ri}^{\otimes p}, x_{sj}^{\otimes p} \rangle$
- Approximate explicit mapping: $\phi_g(x_{ri}) = [x_{ri}, x_{ri} \otimes x_{ri}, \ldots, x_{ri}^{\otimes p}]$

Minor kernel:

- $\phi(x_{ri}) = h(x_{ri}) \otimes \phi_g(x_{ri})$
- $k(x_{ri}, x_{sj}) = \langle h(x_{ri}) \otimes \phi_g(x_{ri}), h(x_{sj}) \otimes \phi_g(x_{sj}) \rangle$

Matching kernel:

- $\Phi(B_i) = \sum_{x_r i \in B_i} h(x_{ri}) \otimes \phi_g(x_{ri})$
- $K(B_i, B_j) = \langle \Phi(B_i), \Phi(B_j) \rangle$

1

Vector of Locally Aggregated Tensors[1]

Gaussian-quantization kernel linearization:

- Assume $\|x_{ri}\| = 1$, then $k_g(x_{ri}, x_{sj}) \propto \exp(\gamma \langle x_{ri}, x_{sj} \rangle)$

Taylor expansion (using Kronecker product $\otimes$):

- $\exp(\gamma \langle x_{ri}, x_{sj} \rangle) = \sum_p \frac{\gamma^p \langle x_{ri}, x_{sj} \rangle^p}{p!} = \sum_p \frac{\gamma^p}{p!} \langle x_{ri}^{\otimes p}, x_{sj}^{\otimes p} \rangle$
- Approximate explicit mapping: $\phi_g(x_{ri}) = [x_{ri}, x_{ri} \otimes x_{ri}, \ldots, x_{ri}^{\otimes p}]$

Minor kernel:

- $\phi(x_{ri}) = h(x_{ri}) \otimes \phi_g(x_{ri})$
- $k(x_{ri}, x_{sj}) = \langle h(x_{ri}) \otimes \phi_g(x_{ri}), h(x_{sj}) \otimes \phi_g(x_{sj}) \rangle$

Matching kernel:

- $\Phi(B_i) = \sum_{x_r i \in B_i} h(x_{ri}) \otimes \phi_g(x_{ri})$
- $K(B_i, B_j) = \langle \Phi(B_i), \Phi(B_j) \rangle$

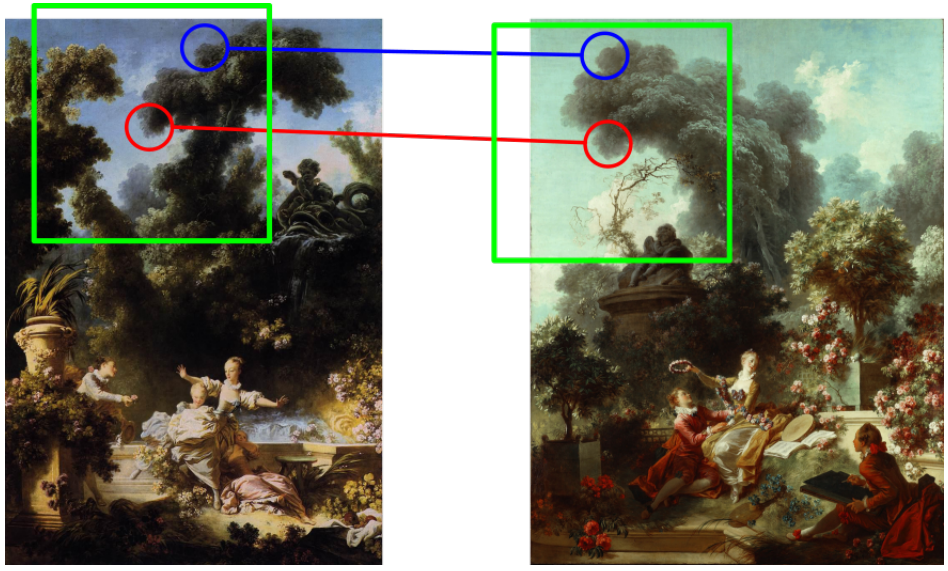
Explicit image representation → Keypoint matching similarity

[1] Picard, David, and Philippe-Henri Gosselin. "Improving image similarity with vectors of locally aggregated tensors." 2011 18th IEEE International Conference on Image Processing. IEEE, 2011.

Geometric consistency kernel

Embedding local geometric consistency in a kernel

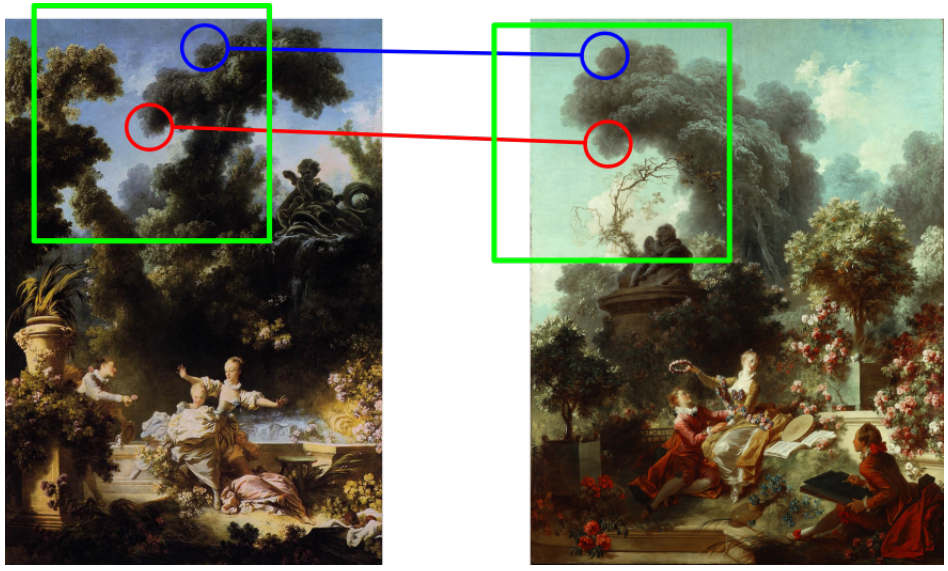
⇒**Similar descriptors** should have **similar neighborhood**



Geometric consistency kernel

Embedding local geometric consistency in a kernel

⇒**Similar descriptors** should have **similar neighborhood**



Nested matching kernels:

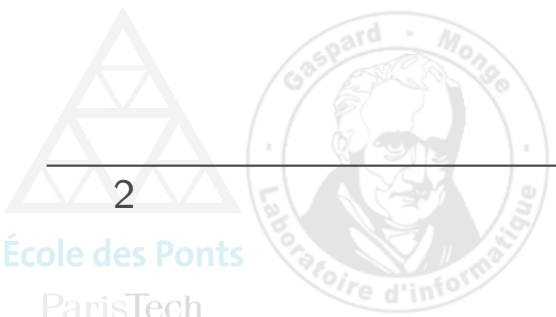
- $\Omega(x_{ri})$: Neighborhood of x_{ri}
- $k(x_{ri}, x_{sj}) = k_s(x_{ri}, x_{sj}) \sum_{u \in \Omega(x_{ri})} \sum_{v \in \Omega(x_{sj})} k_s(x_u, x_v)$

Ω defines the geometric properties (invariances)

Spatial tensor aggregation

Geometric consistency kernel linearization

- $k_s(x_{ri}, x_{sj}) = \langle \phi_s(x_{ri}), \phi_s(x_{sj}) \rangle$
- $k_s(x_{ri}, x_{sj}) k_s(x_u, x_v) = \langle \phi_s(x_{ri}) \otimes \phi_s(x_u), \phi_s(x_{sj}) \otimes \phi(x_v) \rangle$



Spatial tensor aggregation

Geometric consistency kernel linearization

- $k_s(x_{ri}, x_{sj}) = \langle \phi_s(x_{ri}), \phi_s(x_{sj}) \rangle$
- $k_s(x_{ri}, x_{sj}) k_s(x_u, x_v) = \langle \phi_s(x_{ri}) \otimes \phi_s(x_u), \phi_s(x_{sj}) \otimes \phi(x_v) \rangle$

Sum kernel:

- Linearization $\rightarrow$ Global mapping
- $\Phi(B_i) = \sum_{x_{ri} \in B_i} \sum_{x_u \in \Omega(x_{ri})} \phi_s(x_{ri}) \otimes \phi_s(x_u)$

Spatial tensor aggregation

Geometric consistency kernel linearization

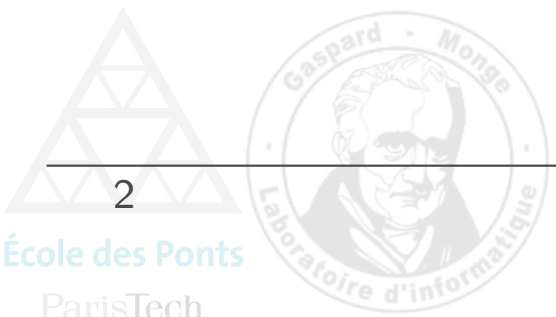
- $k_s(x_{ri}, x_{sj}) = \langle \phi_s(x_{ri}), \phi_s(x_{sj}) \rangle$
- $k_s(x_{ri}, x_{sj}) k_s(x_u, x_v) = \langle \phi_s(x_{ri}) \otimes \phi_s(x_u), \phi_s(x_{sj}) \otimes \phi(x_v) \rangle$

Sum kernel:

- Linearization $\rightarrow$ Global mapping
- $\Phi(B_i) = \sum_{x_{ri} \in B_i} \sum_{x_u \in \Omega(x_{ri})} \phi_s(x_{ri}) \otimes \phi_s(x_u)$

With Quantized-Gaussian kernel approx:

- $k_s(x_{ri}, x_{sj}) = \langle h(x_{ri}) \otimes x_{ri}, h(x_{sj} \otimes x_{sj} \rangle$
- $\Phi(B_i) = \sum_{x_{ri} \in B_i} \sum_{x_u \in \Omega(x_{ri})} h(x_{ri}) \otimes x_{ri} \otimes h(x_u) \otimes x_u$



Spatial tensor aggregation [2]

Geometric consistency kernel linearization

- $k_s(x_{ri}, x_{sj}) = \langle \phi_s(x_{ri}), \phi_s(x_{sj}) \rangle$
- $k_s(x_{ri}, x_{sj}) k_s(x_u, x_v) = \langle \phi_s(x_{ri}) \otimes \phi_s(x_u), \phi_s(x_{sj}) \otimes \phi(x_v) \rangle$

Sum kernel:

- Linearization $\rightarrow$ Global mapping
- $\Phi(B_i) = \sum_{x_{ri} \in B_i} \sum_{x_u \in \Omega(x_{ri})} \phi_s(x_{ri}) \otimes \phi_s(x_u)$

With Quantized-Gaussian kernel approx:

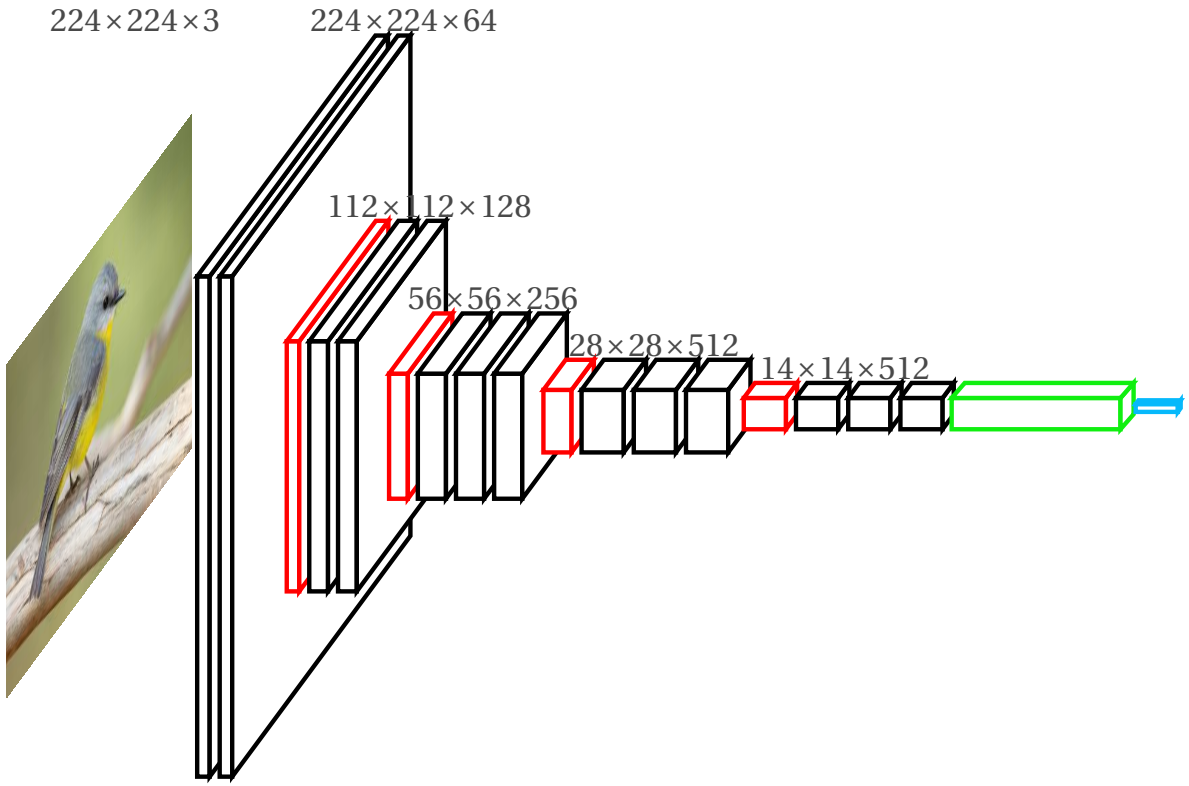
- $k_s(x_{ri}, x_{sj}) = \langle h(x_{ri}) \otimes x_{ri}, h(x_{sj} \otimes x_{sj} \rangle$
- $\Phi(B_i) = \sum_{x_{ri} \in B_i} \sum_{x_u \in \Omega(x_{ri})} h(x_{ri}) \otimes x_{ri} \otimes h(x_u) \otimes x_u$

4th order tensor $\rightarrow$ Locally geometric consistent matching kernel

[2] Picard, David. "Preserving local spatial information in image similarity using tensor aggregation of local features." 2016 IEEE International Conference on Image Processing (ICIP). IEEE, 2016.

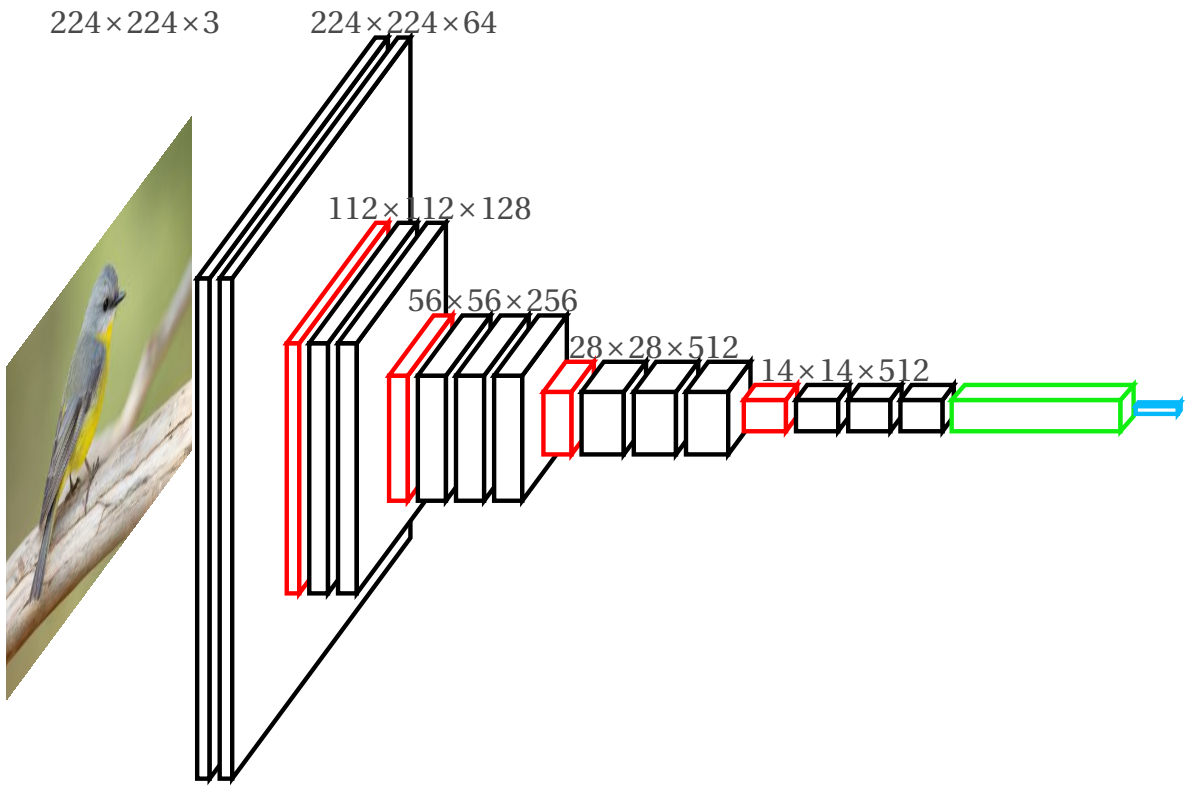
Deep Learning

Features from deep neural network



Deep Learning

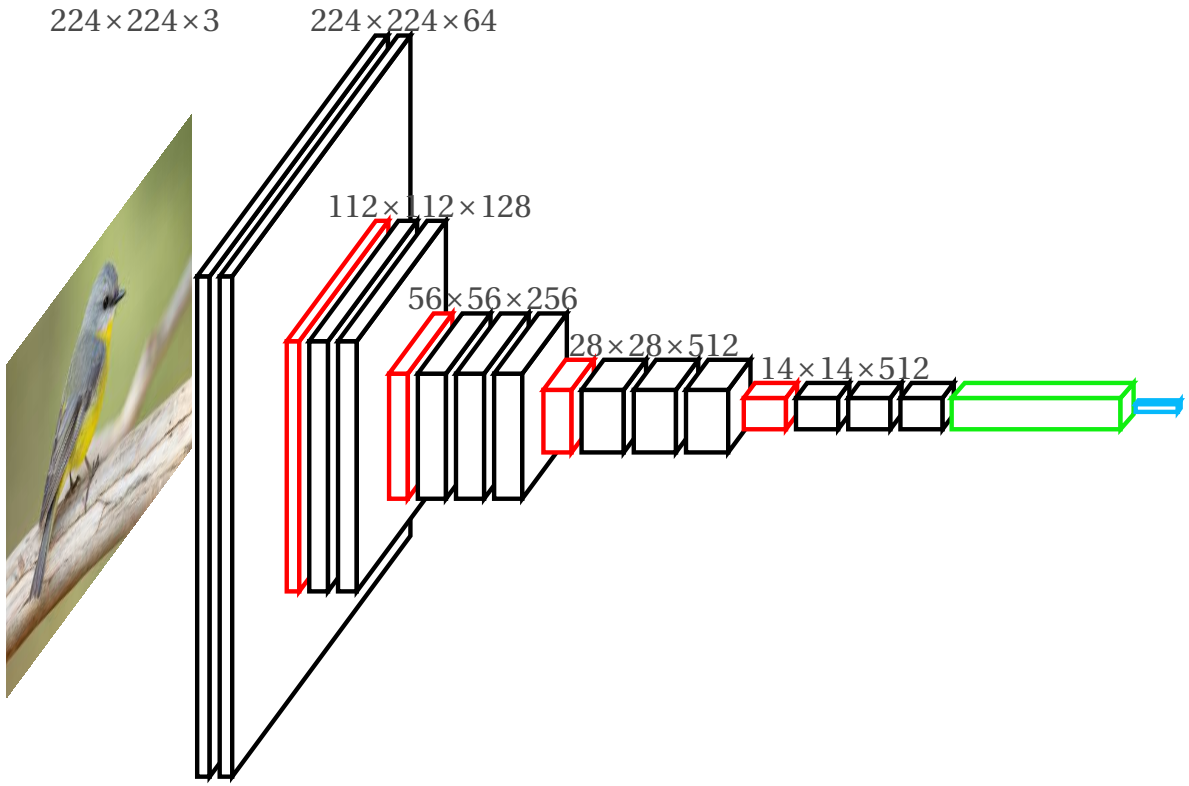
Features from deep neural network



- **Local feature** extractor (CNN)

Deep Learning

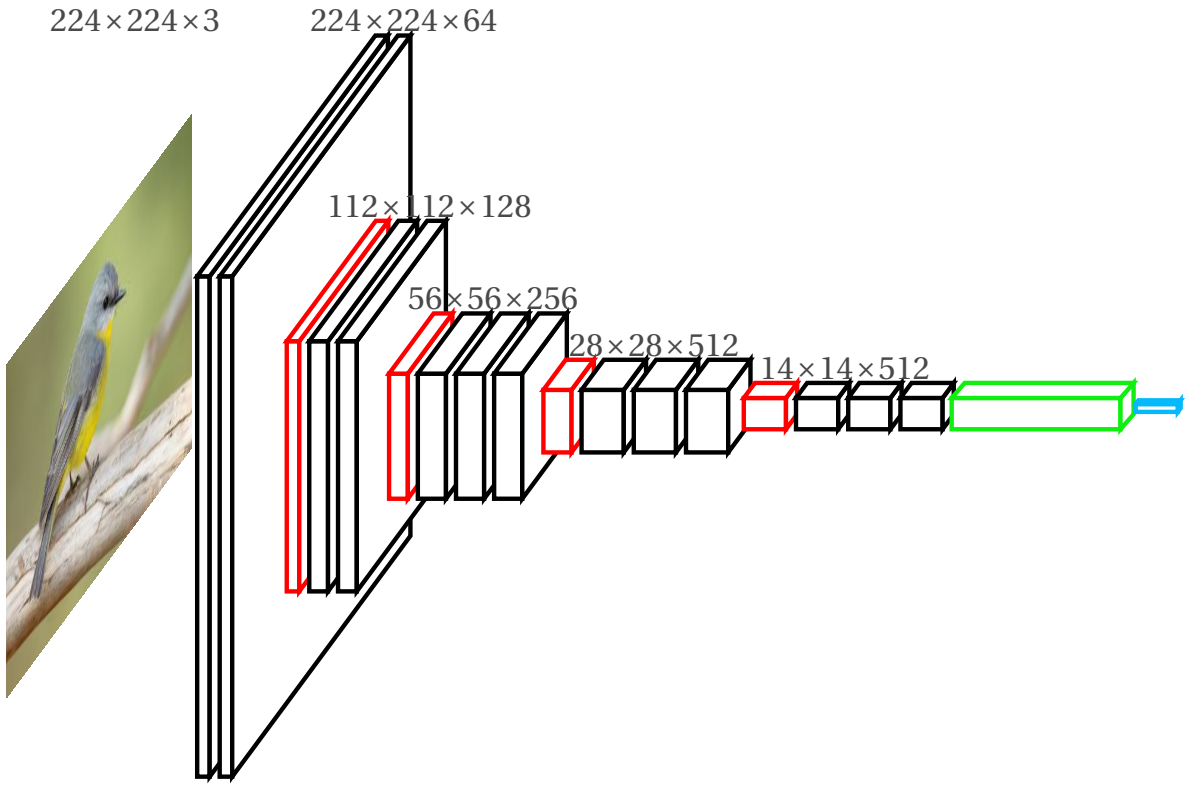
Features from deep neural network



- **Local feature** extractor (CNN)
- Mapping to a vector space (*e.g.*, linearized matching kernel)

Deep Learning

Features from deep neural network



- **Local feature** extractor (CNN)
- Mapping to a vector space (*e.g.*, linearized matching kernel)
- End-to-end **metric learning** $\left(\min \mathbb{E}[\max(0, \langle x_q, x_n \rangle + \alpha - \langle x_q, x_p \rangle)]\right)$

End-to-end training?

Adapting **dictionary based matching kernels** to **deep learning**

End-to-end training?

Adapting **dictionary based matching kernels** to **deep learning**

- Learning discriminative dictionary (**backprop on** $\mathscr{D}$)

End-to-end training?

Adapting **dictionary based matching kernels** to **deep learning**

- Learning discriminative dictionary (**backprop on** $\mathscr{D}$)
- One hot encoding $\rightarrow$ differentiable assignement (**softmax**)

$$h_m(x) = \frac{\exp(\gamma \langle x, \mu_m \rangle)}{\sum_k \exp(\gamma \langle x, \mu_k \rangle)}$$

École des Ponts
ParisTech

End-to-end training?

Adapting **dictionary based matching kernels** to **deep learning**

- Learning discriminative dictionary (**backprop on** $\mathscr{D}$)
- One hot encoding $\rightarrow$ differentiable assignement (**softmax**)

$$h_m(x) = \frac{\exp(\gamma\langle x, \mu_m\rangle)}{\sum_k \exp(\gamma\langle x, \mu_k\rangle)}$$

- Dictionary second order layer

$$y = h(x) \otimes x \otimes h(x) \otimes x$$

École des Ponts
ParisTech

End-to-end training?

Adapting **dictionary based matching kernels** to **deep learning**

- Learning discriminative dictionary (**backprop on** $\mathscr{D}$)
- One hot encoding $\rightarrow$ differentiable assignement (**softmax**)

$$h_m(x) = \frac{\exp(\gamma \langle x, \mu_m \rangle)}{\sum_k \exp(\gamma \langle x, \mu_k \rangle)}$$

- Dictionary second order layer

$$y = h(x) \otimes x \otimes h(x) \otimes x$$

- Equivalent composite layer

École des Ponts ParisTech

End-to-end training?

Adapting **dictionary based matching kernels** to **deep learning**

- Learning discriminative dictionary (**backprop on** $\mathscr{D}$)
- One hot encoding $\rightarrow$ differentiable assignement (**softmax**)

$$h_m(x) = \frac{\exp(\gamma\langle x, \mu_m\rangle)}{\sum_k \exp(\gamma\langle x, \mu_k\rangle)}$$

- Dictionary second order layer

$$y = h(x) \otimes x \otimes h(x) \otimes x$$

- Equivalent composite layer

High dimensional output $\rightarrow$ factorization

École des Ponts ParisTech

Efficient Codebook and Factorization

Compression by projection

- $z_i(x) = \langle p_i, h(x) \otimes x \otimes h(x) \otimes x \rangle,\ p_i \in \mathbb{R}^{M \times d \times M \times d}$
- $z(x) = [z_i(x)]_{1 \leq i \leq D}$

Efficient Codebook and Factorization

Compression by projection

- $z_i(x) = \langle p_i, h(x) \otimes x \otimes h(x) \otimes x \rangle, \ p_i \in \mathbb{R}^{M \times d \times M \times d}$
- $z(x) = [z_i(x)]_{1 \leq i \leq D}$

Compression by rank-1 factorization

- $\langle p_i, h(x) \otimes x \otimes h(x) \otimes x \rangle = \langle U_i, h(x) \otimes x \rangle \langle V_i, h(x) \otimes x \rangle$
- $z_i(x) = (h(x)^\top U_i x) \circ (h(x)^\top V_i x)$
- $U_i, V_i :=$ Mapping of x to the assignment space

Efficient Codebook and Factorization

Compression by projection

- $z_i(x) = \langle p_i, h(x) \otimes x \otimes h(x) \otimes x \rangle,\ p_i \in \mathbb{R}^{M \times d \times M \times d}$
- $z(x) = [z_i(x)]_{1 \leq i \leq D}$

Compression by rank-1 factorization

- $\langle p_i, h(x) \otimes x \otimes h(x) \otimes x \rangle = \langle U_i, h(x) \otimes x \rangle \langle V_i, h(x) \otimes x \rangle$
- $z_i(x) = (h(x)^\top U_i x) \circ (h(x)^\top V_i x)$
- $U_i, V_i :=$ Mapping of x to the assignment space

Compression by sharing

- Common pool of projectors $\tilde{U}, \tilde{V}$
- Combination of projectors $U_i = A_i \tilde{U}, V_i = B_i \tilde{V}$
- $z_i(x) = (h(x)^\top A_i \tilde{U} x) \circ (h(x)^\top B_i \tilde{V} x)$

Efficient Codebook and Factorization [3]

Compression by projection

- $z_i(x) = \langle p_i, h(x) \otimes x \otimes h(x) \otimes x \rangle,\ p_i \in \mathbb{R}^{M \times d \times M \times d}$
- $z(x) = [z_i(x)]_{1 \leq i \leq D}$

Compression by rank-1 factorization

- $\langle p_i, h(x) \otimes x \otimes h(x) \otimes x \rangle = \langle U_i, h(x) \otimes x \rangle \langle V_i, h(x) \otimes x \rangle$
- $z_i(x) = (h(x)^\top U_i x) \circ (h(x)^\top V_i x)$
- $U_i, V_i :=$ Mapping of x to the assignment space

Compression by sharing

- Common pool of projectors $\tilde{U}, \tilde{V}$
- Combination of projectors $U_i = A_i \tilde{U}, V_i = B_i \tilde{V}$
- $z_i(x) = (h(x)^\top A_i \tilde{U} x) \circ (h(x)^\top B_i \tilde{V} x)$

Efficient codebook/2nd order repr. $\rightarrow$ **standard deep learning operators**

[3] Jacob, Pierre, et al. "Efficient codebook and factorization for second order representation learning." 2019 IEEE International Conference on Image Processing (ICIP). IEEE, 2019.

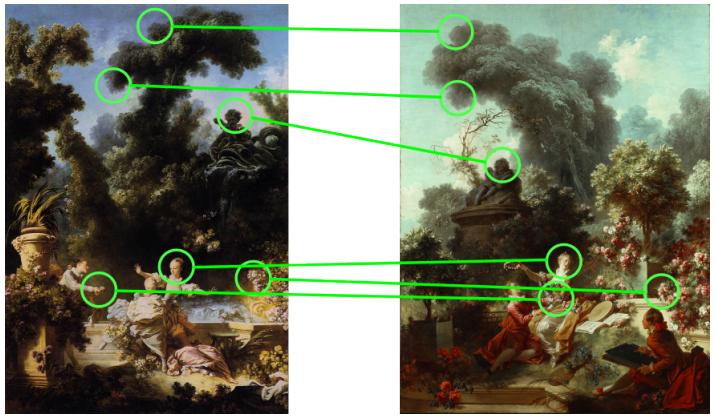
Outline

EMD

$$\min \sum w_{ij} d(x_i, x_j)$$

$$w_{ij} \geq 0, \sum_i w_{ij} = 1, \sum_j w_{ij} = 1$$

Wasserstein



MMD

$$\sup_T \mathbb{E}_p[T(x)] - \mathbb{E}_q[T(x)]$$

HORDE $\mathbb{E}_{x \sim \mathscr{I}}[x^{\otimes k}]$

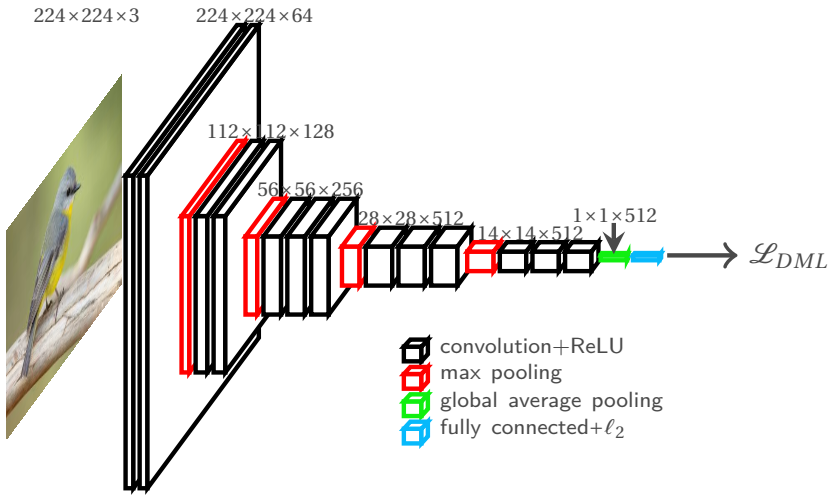
Matching Kernels

$$(\approx \langle \phi(B_i), \phi(B_j) \rangle)$$

Keypoints matching

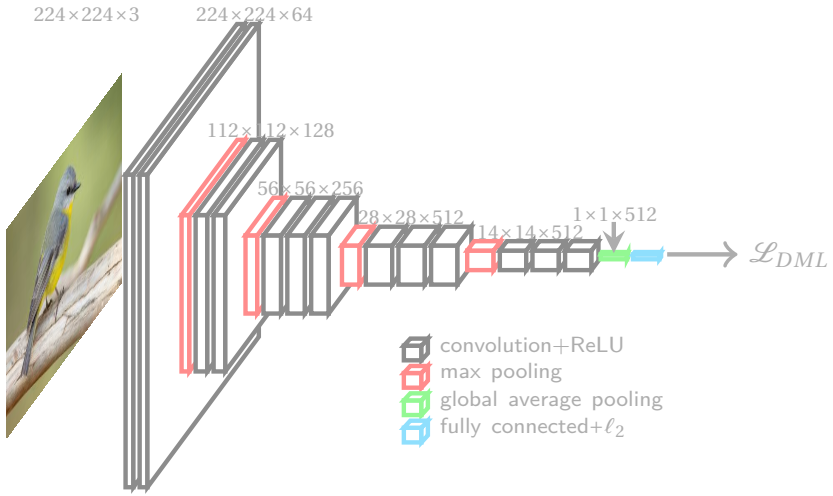
Optimizing the mean

Sum Kernel → Spatial averaging of all features



Optimizing the mean

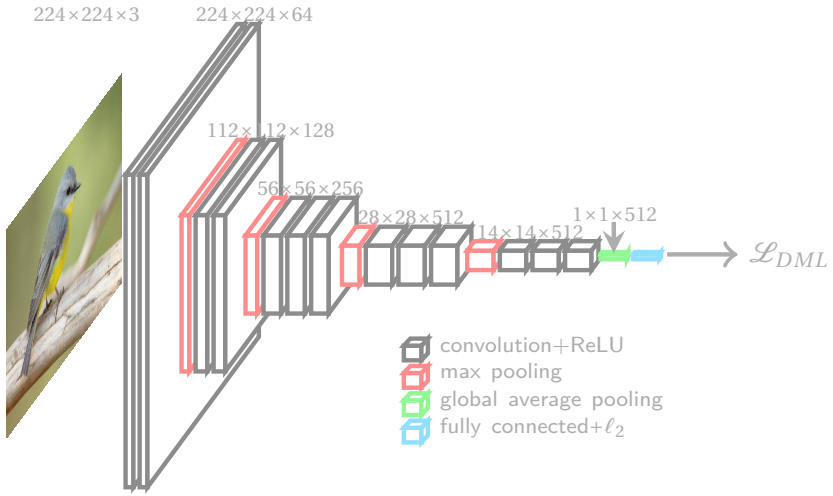
Sum Kernel → Spatial averaging of all features



- Learning similar/dissimilar averages

Optimizing the mean

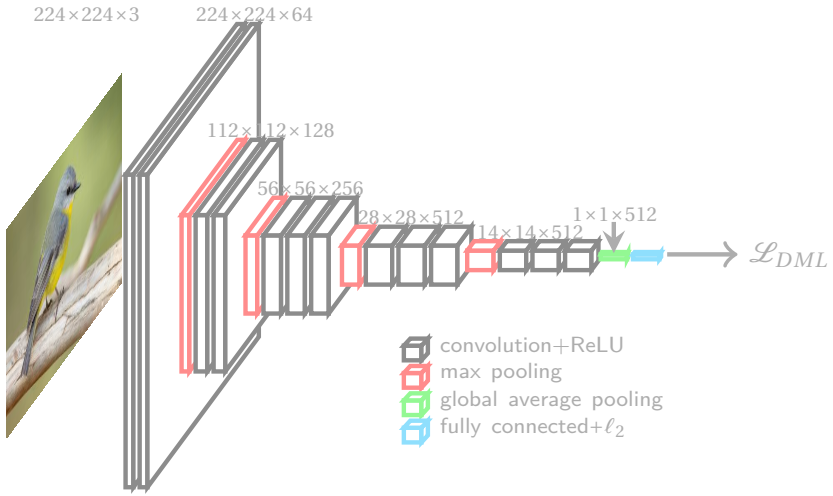
Sum Kernel → Spatial averaging of all features



- Learning similar/dissimilar averages
- Ignoring the distribution of local features

Optimizing the mean

Sum Kernel → Spatial averaging of all features

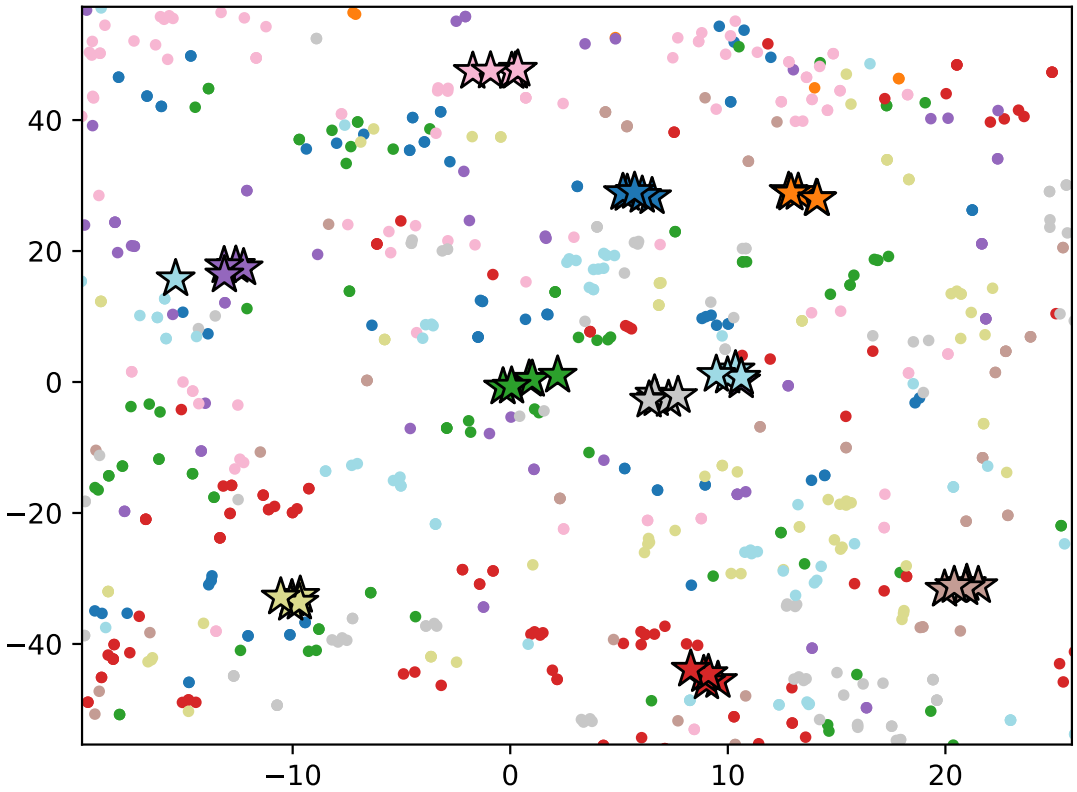


- Learning similar/dissimilar averages
- Ignoring the distribution of local features

Similar image → Similar local features???

Scattering problem

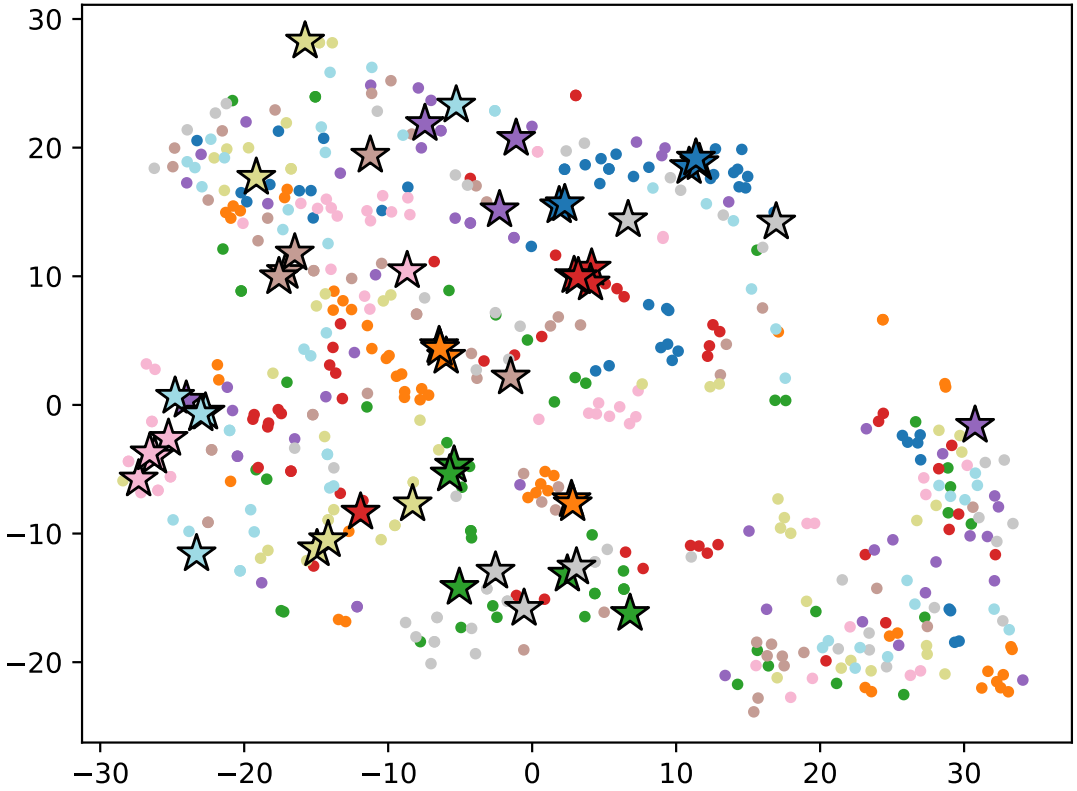
- local features, ⋆ image representations



- Good representations, Non-discriminative local features

Scattering problem

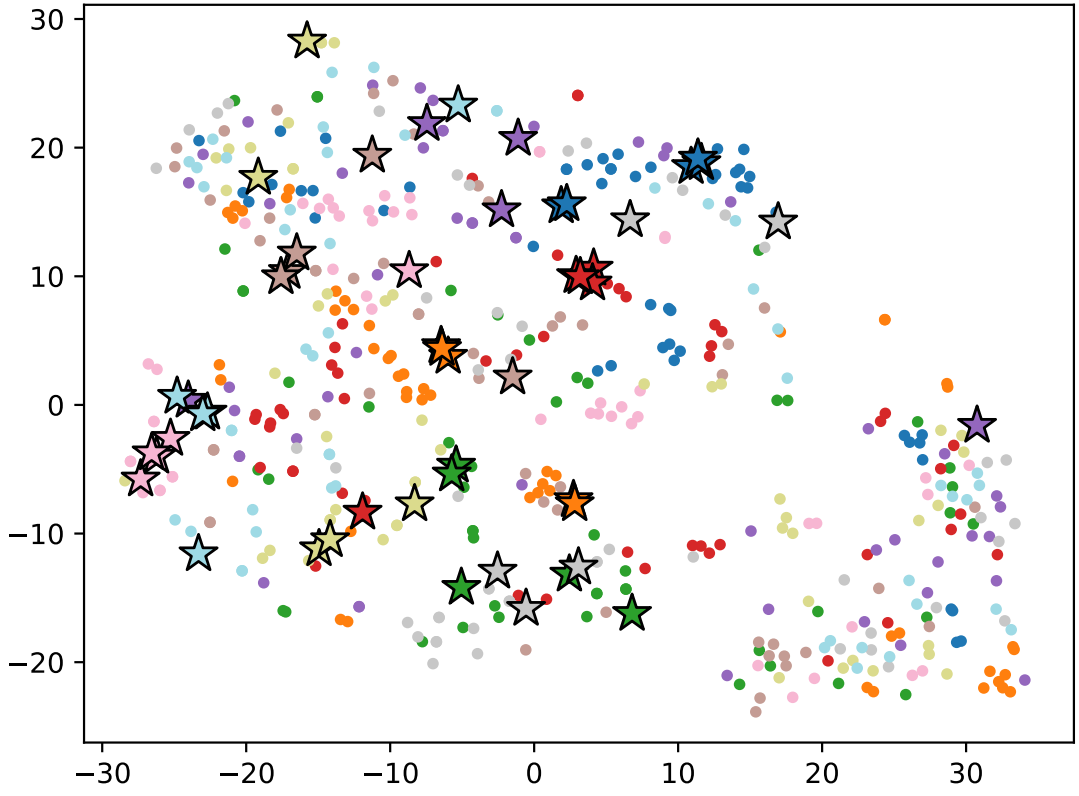
- local features, ⋆ image representations



- Good representations, Non-discriminative local features
- Representations **not robust to sampling issues**

Scattering problem

- local features, ⋆ image representations



- Good representations, Non-discriminative local features
- Representations **not robust to sampling issues**

Distribution metric learning

Similar/dissimilar images have similar/dissimilar deep feature distributions

Tackling deep feature distributions

DML optimizes the mean (GAP):

$$\min_{\phi} \sum_{I,J \in \mathscr{P}} \|\mathbb{E}_{i\sim I}[\phi(x_i)] - \mathbb{E}_{j\sim J}[\phi(x_j)]\|^2 + \sum_{I,J \in \mathscr{N}} \max(0, \alpha - \|\mathbb{E}_{i\sim I}[\phi(x_i)] - \mathbb{E}_{j\sim J}[\phi(x_j)]\|^2)$$

⇒ **Highly different deep feature distributions with the same mean!**

École des Ponts
ParisTech

Tackling deep feature distributions

DML optimizes the mean (GAP):

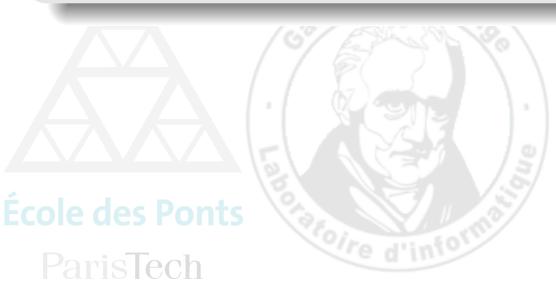
$$\min_{\phi} \sum_{I,J \in \mathscr{P}} \|\mathbb{E}_{i\sim I}[\phi(x_i)] - \mathbb{E}_{j\sim J}[\phi(x_j)]\|^2 + \sum_{I,J \in \mathscr{N}} \max(0, \alpha - \|\mathbb{E}_{i\sim I}[\phi(x_i)] - \mathbb{E}_{j\sim J}[\phi(x_j)]\|^2)$$

⇒ **Highly different deep feature distributions with the same mean!**

Distribution metric learning

We should perform metric learning such that:

- Similar images have similar deep feature distributions
- Dissimilar images have dissimilar deep feature distributions



Distributions of local features

Point set optimal assignment: EMD

$$\min_{w} \sum_{rs} w_{rs} d(\mathbf{x}_{ri}, \mathbf{x}_{sj})$$

$$\text{s.t.} \forall r, \forall s, w_{rs} \geq 0$$

$$\forall r, \sum_{s} w_{rs} = 1; \forall s, \sum_{r} w_{rs} = 1$$

Distributions of local features

Point set optimal assignment: EMD

$$\min_{w} \sum_{rs} w_{rs} d(\mathbf{x}_{ri}, \mathbf{x}_{sj})$$

$$\text{s.t.} \forall r, \forall s, w_{rs} \geq 0$$

$$\forall r, \sum_{s} w_{rs} = 1; \forall s, \sum_{r} w_{rs} = 1$$

Random local features sampled from image distribution $\mathbf{x}_{ri} \sim \mathscr{D}_i$

- Distance between $\mathscr{D}_i$ and $\mathscr{D}_j$
- Wasserstein distance: minimum change to $\mathscr{D}_i$ to obtain local features that look as if sampled from $\mathscr{D}_j$

$$\inf \mathbb{E}[d(\mathbf{x}_{ri}, \mathbf{x}_{sj})]$$

Unknown moving deep feature distribution

Deep features distribution:

- **Unknown** (no model, few assumptions)
- **Changing** (opimized through gradient descent)

Unknown moving deep feature distribution

Deep features distribution:

- **Unknown** (no model, few assumptions)
- **Changing** (opimized through gradient descent)

Metric learning over distributions:

- Distance between distributions (Wasserstein, MMD)
- Kernel mean embedding $\leftrightarrow$ Maximum mean Discrepancy

$$\underbrace{\sup_{\|f\| \leq 1} \int f(x) d\mathbb{P}(x) - \int f(y) d\mathbb{Q}(y)}_{MMD} = \underbrace{\|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|_{\mathcal{H}}}_{\text{Kernel mean embedding}}$$

École des Ponts
ParisTech

Unknown moving deep feature distribution

Deep features distribution:

- **Unknown** (no model, few assumptions)
- **Changing** (opimized through gradient descent)

Metric learning over distributions:

- Distance between distributions (Wasserstein, MMD)
- Kernel mean embedding $\leftrightarrow$ Maximum mean Discrepancy

$$\underbrace{\sup_{\|f\|\leq 1} \int f(x)d\mathbb{P}(x) - \int f(y)d\mathbb{Q}(y)}_{MMD} = \underbrace{\|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|_{\mathcal{H}}}_{\text{Kernel mean embedding}}$$

Kernel Mean Embedding:

- $\mu_{\mathbb{P}} = \mathbb{E}_{\mathbb{P}}[\phi(x)]$, Gaussian kernel $\rightarrow \phi(x) \approx [x, x \otimes x, \ldots, x^{\otimes p}]$
- Approximate $\mathbb{P}$ by its **high order moments**

High Order Moments estimation

Recall:

$$\langle \underbrace{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}_{\text{K times}}, \mathbf{y} \otimes \cdots \otimes \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle^K$$

High Order Moments estimation

Recall:

$$\langle \underbrace{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}_{\text{K times}}, \mathbf{y} \otimes \cdots \otimes \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle^K$$

iid random MacLaurin projections $\mathbf{w}_k$ such that $\mathbb{E}[\mathbf{w}_k \mathbf{w}_k^\top] = I$:

$$\begin{aligned}
\mathbb{E}[\langle \mathbf{x}, \mathbf{w}_1 \rangle \dots \langle \mathbf{x}, \mathbf{w}_K \rangle \langle \mathbf{y}, \mathbf{w}_1 \rangle \dots \langle \mathbf{y}, \mathbf{w}_K \rangle] &= \mathbb{E}[\langle \mathbf{x}, \mathbf{w}_1 \rangle \langle \mathbf{y}, \mathbf{w}_1 \rangle] \dots \mathbb{E}[\langle \mathbf{x}, \mathbf{w}_K \rangle \langle \mathbf{y}, \mathbf{w}_K \rangle] \\
&= \mathbf{x}^\top \mathbb{E}[\mathbf{w}_1 \mathbf{w}_1^\top] \mathbf{x} \dots \mathbf{y}^\top \mathbb{E}[\mathbf{w}_K \mathbf{w}_K^\top] \mathbf{y} \\
&= \langle \mathbf{x}, \mathbf{y} \rangle \dots \langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle^K
\end{aligned}$$

High Order Moments estimation

Recall:

$$\langle \underbrace{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}_{\text{K times}}, \mathbf{y} \otimes \cdots \otimes \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle^K$$

iid random MacLaurin projections $\mathbf{w}_k$ such that $\mathbb{E}[\mathbf{w}_k \mathbf{w}_k^\top] = I$:

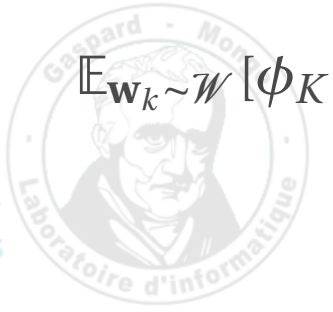
$$\begin{aligned}\mathbb{E}[\langle \mathbf{x}, \mathbf{w}_1 \rangle \dots \langle \mathbf{x}, \mathbf{w}_K \rangle \langle \mathbf{y}, \mathbf{w}_1 \rangle \dots \langle \mathbf{y}, \mathbf{w}_K \rangle] &= \mathbb{E}[\langle \mathbf{x}, \mathbf{w}_1 \rangle \langle \mathbf{y}, \mathbf{w}_1 \rangle] \dots \mathbb{E}[\langle \mathbf{x}, \mathbf{w}_K \rangle \langle \mathbf{y}, \mathbf{w}_K \rangle] \\ &= \mathbf{x}^\top \mathbb{E}[\mathbf{w}_1 \mathbf{w}_1^\top] \mathbf{x} \dots \mathbf{y}^\top \mathbb{E}[\mathbf{w}_K \mathbf{w}_K^\top] \mathbf{y} \\ &= \langle \mathbf{x}, \mathbf{y} \rangle \dots \langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle^K\end{aligned}$$

Define explicit mapping ϕ:

$$\phi_K(\mathbf{x}) = \prod_{k=1}^{K} \langle \mathbf{w}_k \ ; \ \mathbf{x} \rangle$$

then:

$$\mathbb{E}_{\mathbf{w}_k \sim \mathcal{W}}[\phi_K(\mathbf{x}) \phi_K(\mathbf{y})] = \langle \mathbf{x} \ ; \ \mathbf{y} \rangle^K = \langle \underbrace{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}_{K \text{ times}} ; \mathbf{y} \otimes \cdots \otimes \mathbf{y} \rangle$$



High Order Moments estimation

Recall:

$$\langle \underbrace{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}_{\text{K times}}, \mathbf{y} \otimes \cdots \otimes \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle^K$$

iid random MacLaurin projections $\mathbf{w}_k$ such that $\mathbb{E}[\mathbf{w}_k \mathbf{w}_k^\top] = I$:

$$\begin{aligned}\mathbb{E}[\langle \mathbf{x}, \mathbf{w}_1 \rangle \dots \langle \mathbf{x}, \mathbf{w}_K \rangle \langle \mathbf{y}, \mathbf{w}_1 \rangle \dots \langle \mathbf{y}, \mathbf{w}_K \rangle] &= \mathbb{E}[\langle \mathbf{x}, \mathbf{w}_1 \rangle \langle \mathbf{y}, \mathbf{w}_1 \rangle] \dots \mathbb{E}[\langle \mathbf{x}, \mathbf{w}_K \rangle \langle \mathbf{y}, \mathbf{w}_K \rangle] \\ &= \mathbf{x}^\top \mathbb{E}[\mathbf{w}_1 \mathbf{w}_1^\top] \mathbf{x} \dots \mathbf{y}^\top \mathbb{E}[\mathbf{w}_K \mathbf{w}_K^\top] \mathbf{y} \\ &= \langle \mathbf{x}, \mathbf{y} \rangle \dots \langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle^K\end{aligned}$$

Define explicit mapping ϕ:

$$\phi_K(\mathbf{x}) = \prod_{k=1}^{K} \langle \mathbf{w}_k \ ; \ \mathbf{x} \rangle$$

then:

$$\mathbb{E}_{\mathbf{w}_k \sim \mathcal{W}}[\phi_K(\mathbf{x}) \phi_K(\mathbf{y})] = \langle \mathbf{x} \ ; \ \mathbf{y} \rangle^K = \langle \underbrace{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}_{K \text{ times}} ; \mathbf{y} \otimes \cdots \otimes \mathbf{y} \rangle$$

$\Rightarrow$ **Untrainable projections $\mathbf{w}_k$**

Recursive HOM estimation[4]

Using trainable projections $\mathbf{W}_k$:

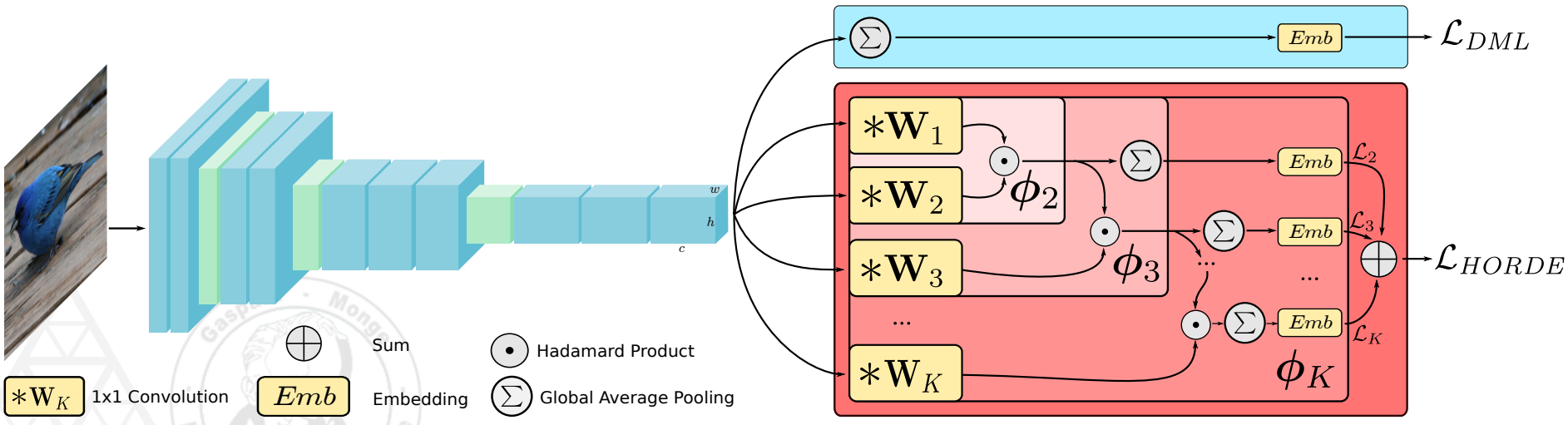
$$\boldsymbol{\phi}_k(\mathbf{x}) = \boldsymbol{\phi}_{k-1}(\mathbf{x}) \odot \left(\mathbf{W}_k^\top \mathbf{x}\right)$$

Average several projections:

$$\langle \mathbf{x} \ ; \ \mathbf{y} \rangle^K = \mathbb{E}[\prod \langle x, \mathbf{w}_k \rangle \langle y, \mathbf{w}_k \rangle] \approx \frac{1}{d} \langle \boldsymbol{\phi}_K(\mathbf{x}) \ ; \ \boldsymbol{\phi}_K(\mathbf{y}) \rangle$$

Cascade architecture

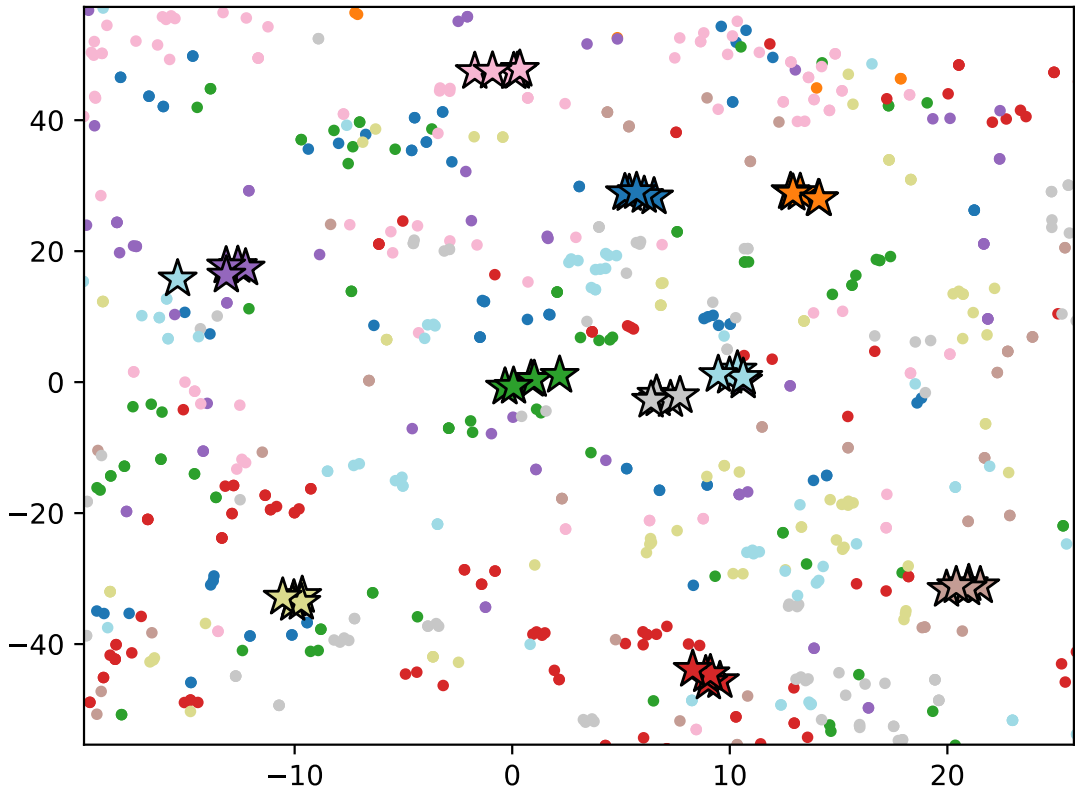
- Fewer weights prevents overfitting and computationally efficient



4 Jacob, Pierre, et al. "Metric learning with horde: High-order regularizer for deep embeddings." Proceedings of the IEEE International Conference on Computer Vision. 2019.

Qualitative results

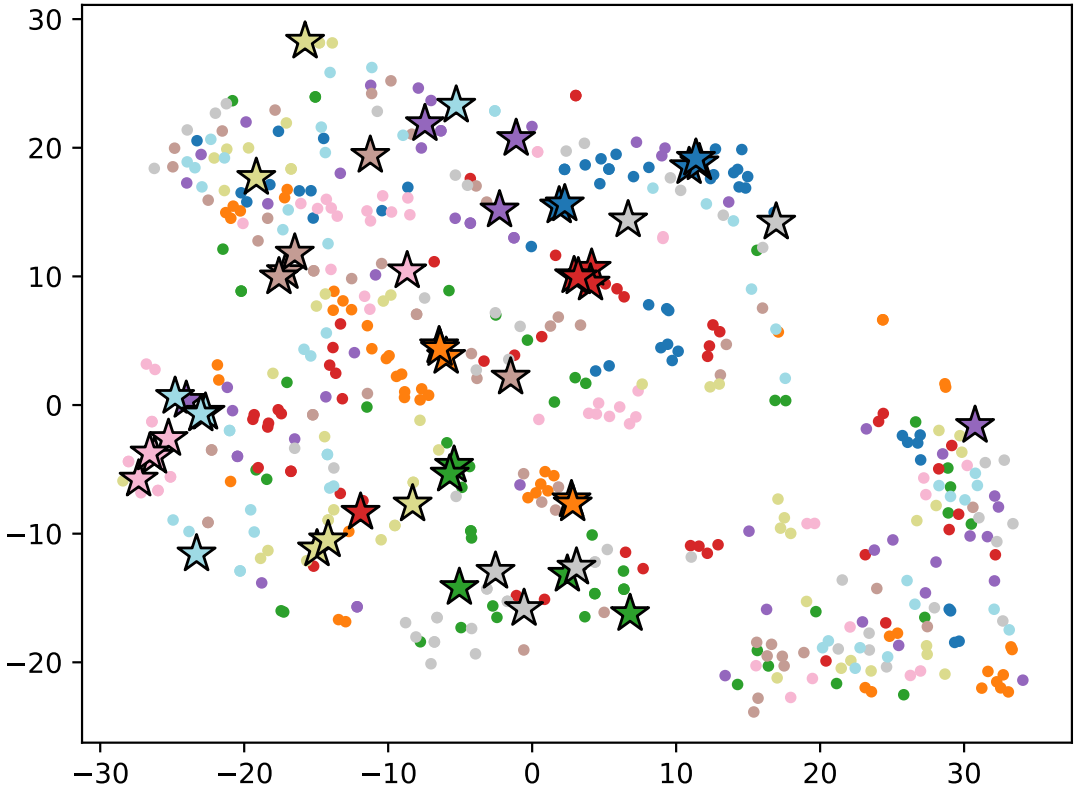
Full sampling **without** HORDE regularization



MNIST, test data, ⋆ image representations, • deep features

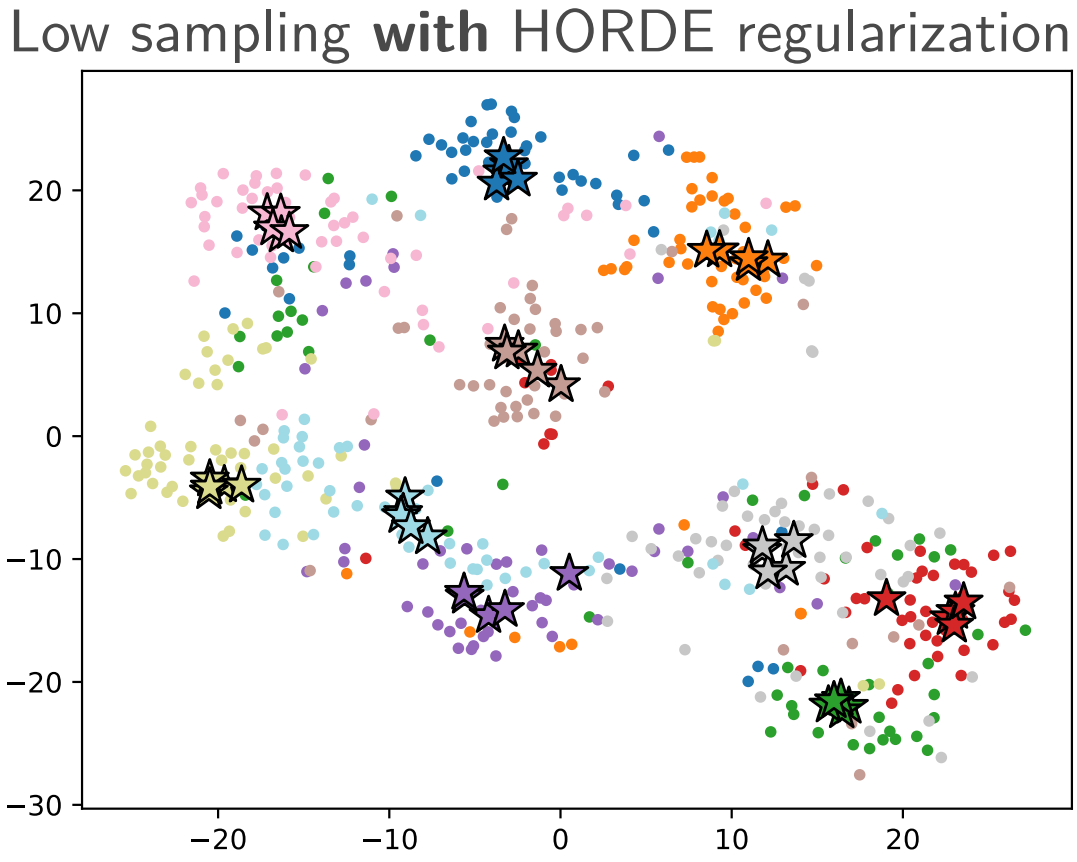
Qualitative results

Low sampling **without** HORDE regularization



MNIST, test data, ⋆ image representations, • deep features

Qualitative results



MNIST, test data, ⋆ image representations, • deep features

HO impact

R@1: Percentage nearest neighbor is correct

k	1	2		3			4				5					6					
n	1	1	2	1	2	3	1	2	3	4	1	2	3	4	5	1	2	3	4	5	6
R@1	55.9	**57.8**	58.6	56.8	58.0	56.9	**57.8**	58.8	57.6	56.1	57.4	57.7	56.8	56.3	53.3	57.4	57.9	57.1	55.6	54.4	50.7

MacLaurin

k	1	2		3			4				5					6					
n	1	1	2	1	2	3	1	2	3	4	1	2	3	4	5	1	2	3	4	5	6
R@1	55.9	57.0	53.4	57.6	54.7	50.6	57.9	55.4	52.3	47.6	58.1	55.9	53.1	48.4	43.7	**58.4**	55.7	52.9	47.8	43.9	40.5

Full train

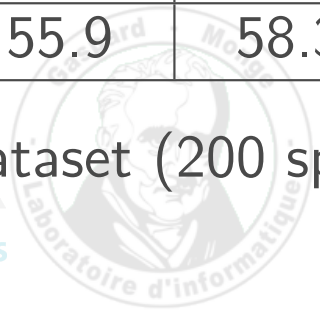
k	1	2		3			4				5					6					
n	1	1	2	1	2	3	1	2	3	4	1	2	3	4	5	1	2	3	4	5	6
R@1	55.9	57.0	53.4	57.9	56.1	54.2	57.6	55.4	54.3	53.0	**58.3**	56.3	56.0	54.7	52.4	57.9	56.6	55.8	55.0	53.9	51.6

HORDE

Concatenation of all HO:

	Baseline	HORDE	HORDE concat
R@1	55.9	58.3	**59.4**

CUB dataset (200 species of birds)



Conclusion

Image similarity based on local patterns:

- Matching keypoints is ++ but non ML/DL compatible
- Matching kernels are ML compatible drop in replacement
- Kernel linearization makes matching kernels DL compatible
- Local features matching rel. to matching distributions
- Matching distribution with MMD eq. to Kernel Mean Embedding
- High order moments rel. to Matching Kernel approx and KME

High order moments: embedding local information

Papers + Code

Contact: jacobpie@fel.cvut.cz (or david.picard@enpc.fr)

HORDE (2019):
Github: https://github.com/pierre-jacob/ICCV2019-Horde
Paper: https://openaccess.thecvf.com/content_ICCV_2019/html/Jacob_Metric_Learning_With_HORDE_High-Order_Regularizer_for_Deep_Embeddings_ICCV_2019_paper.html

JCF (2018):
Paper: https://arxiv.org/abs/1906.01972

STA (2016):
Paper: https://hal.archives-ouvertes.fr/hal-01359109

VLAT (2011):
Paper: https://hal.archives-ouvertes.fr/hal-00591993/