Foundations of Second- and Higher-order Representations

Piotr Koniusz

Data61/CSIRO
Australian National University

piotr.koniusz@data61.csiro.au

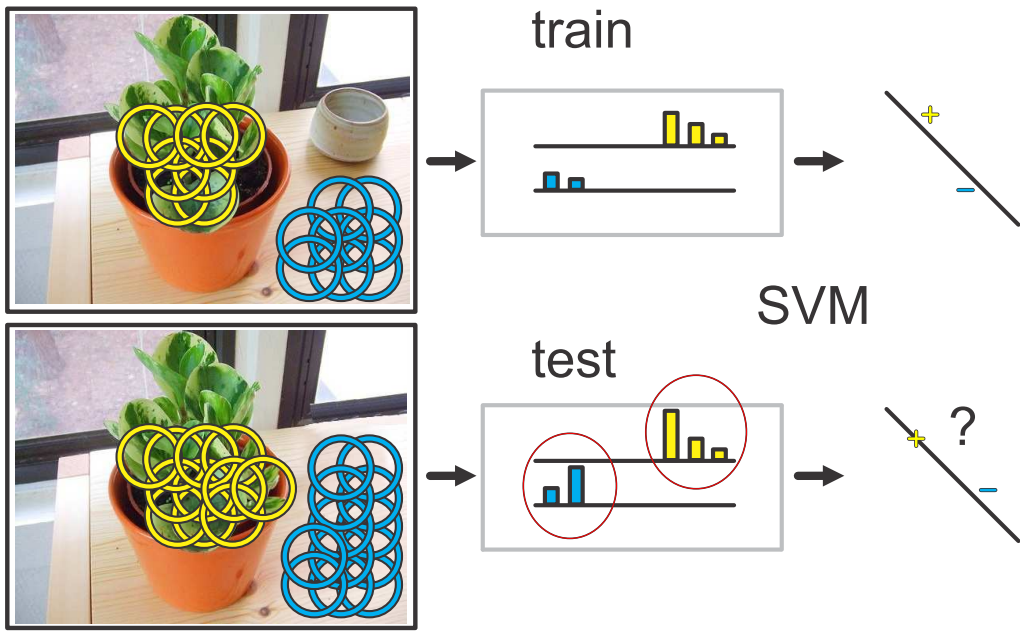
December 14, 2019

The Contents

- Burstiness
- First-order Pooling and Pooling Operators
- Higher-order Occurrence Pooling
- Non-Euclidean Distances
- Tensors: Properties, Factorisation via CANDECOMP and HOSVD
- Tensor Descriptors: TOSST, HOP, Push-forward Distributions, Derivations, Third-order Sparse Coding
- Action Recognition: 3D skeleton sequences, SCK, DCK, HOP kernels, EPN
- Domain Adaptation: So-HoT, Riemannian Alignment, Open MIC dataset, Action Recognition
- Faces and Style Transfer
- Deeper Look at Power Normalisations
- Few-shot Learning

Burstiness

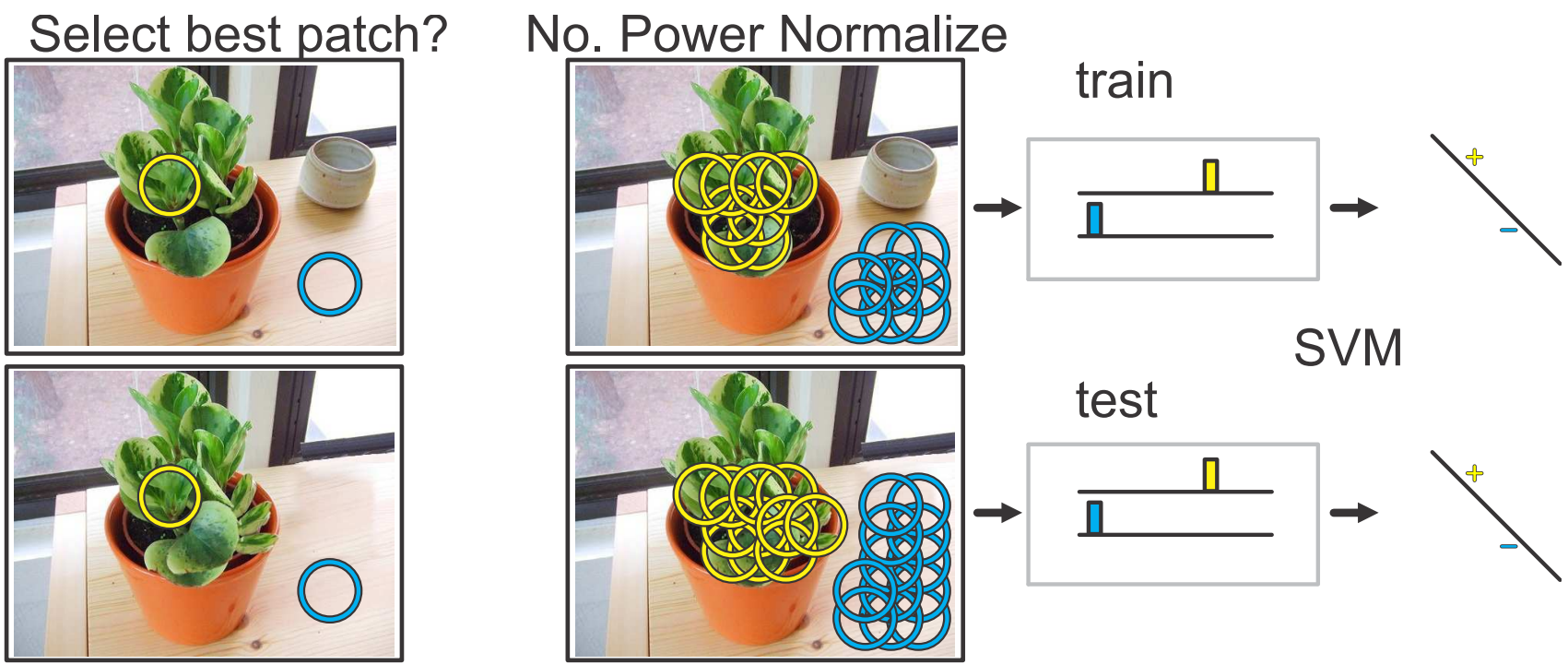
- A property that a visual element appears more frequently (or less frequently) than statistically expected.
- Source of nuissance variability (a signal but statistically speaking it is a noise to us)



- Term coined in [Jegou *et al.*, On the Burstiness of Visual Elements (CVPR'09)]
- But the idea explored earlier [Boughorbel *et al.*, Generalized Histogram Intersection Kernel for Image Recognition (ICIP'05)] via the square root+intersection kernel: $k(\phi, \phi') = \sum_i \min(\phi_i^{0.5}, \phi_i'^{0.5})$.

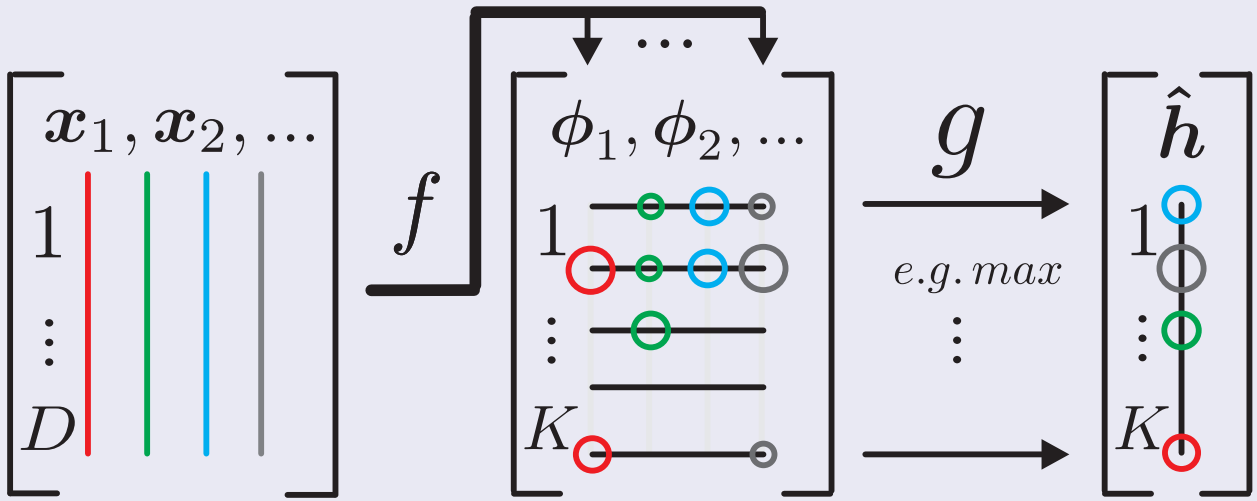
Burstiness

- Detecting consistent across images most representative patches may be hard.
- Instead, equalize statistics via Power Normalization: $\psi(\phi) = \phi^{\gamma},\ 0 < \gamma \leq 1.$



First-order Occurrence Pooling

- The local descriptors $\boldsymbol{x}$ are extracted from an image and coded by f that operates on columns.
- Pooling g aggregates visual words from the mid-level features ϕ along rows:



- Three simple steps:

$$\phi_n = f(\boldsymbol{x}_n, \mathbf{D}), \quad \forall n \in \mathcal{N} \quad \text{(encode, } e.g. \text{ SC, LLC, SA)} \tag{1}$$

$$\hat{h}_k = g\left(\{\phi_{kn}\}_{n \in \mathcal{N}}\right) \quad \text{(pool, Max-pooling, Average, MaxExp)} \tag{2}$$

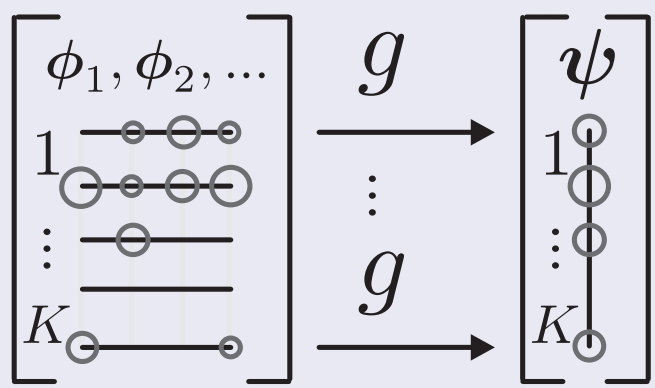
$$\boldsymbol{h} = \hat{\boldsymbol{h}} / \|\hat{\boldsymbol{h}}\|_2 \quad \text{(normalise)} \tag{3}$$

Baseline pooling operators

- Operator g aggregates mid-level features ϕ into image signature ψ.

Pooling g:

e.g.:
- avg
- max



- Baseline operators:
  - **Average pooling**: performs count of a given visual word in image.
  - **Max-pooling**: detects presence of a visual word in image.

Observation: Max-pooling is invariant to the repeatability of the feature: a proxy for invariance to repeatable visual stimuli.

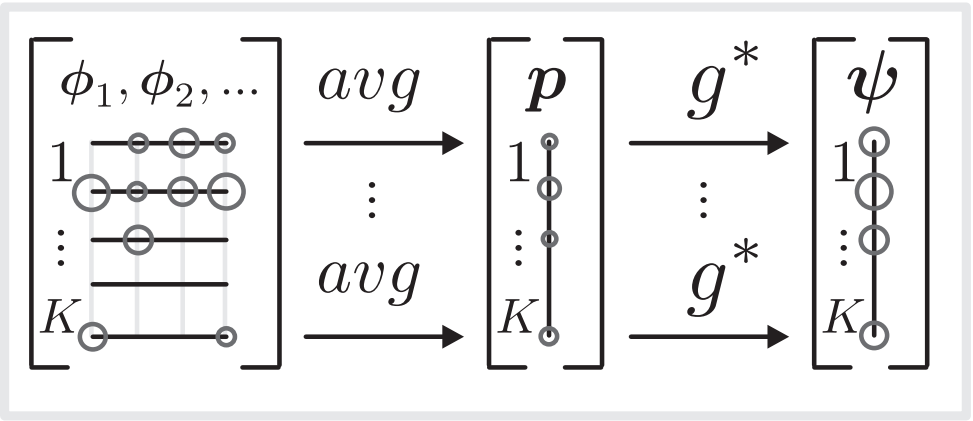
Analytical pooling operators

g^* can be:

$1-(1-p)^N$

or

$p^{0.5}$



- **MaxExp** - *probability of at least one mid-level feature to be 1 given* $\boldsymbol{d}_k$: $1-(1-p)^N$ (Bernoulli i.i.d. features) [Boureau *et al.*, A Theoretical Analysis of Feature Pooling in Vision Alg. (ICML'10)]
- **ExaPro** - *the probability of at least one visual word* $\boldsymbol{d}_k$ *present in image* i: $1-\prod_{n\in\mathcal{N}}(1-\phi_{kn})$ (i.i.d. features)
  [Lingqiao *et al.*, In Defence of Soft-assignment Coding (ICCV'11)]
- **Gamma** - Power Normalisation p^γ
- **AxMin** - we proposed a universal approximator: $\min(\beta p, 1)$
- We showed that all these operators are similar.
  [Koniusz *et al.*, Comparison of Mid-Level Feature Coding Approaches And Pooling Strategies in Visual Concept Detection (CVIU'12)]

Analytical pooling operators

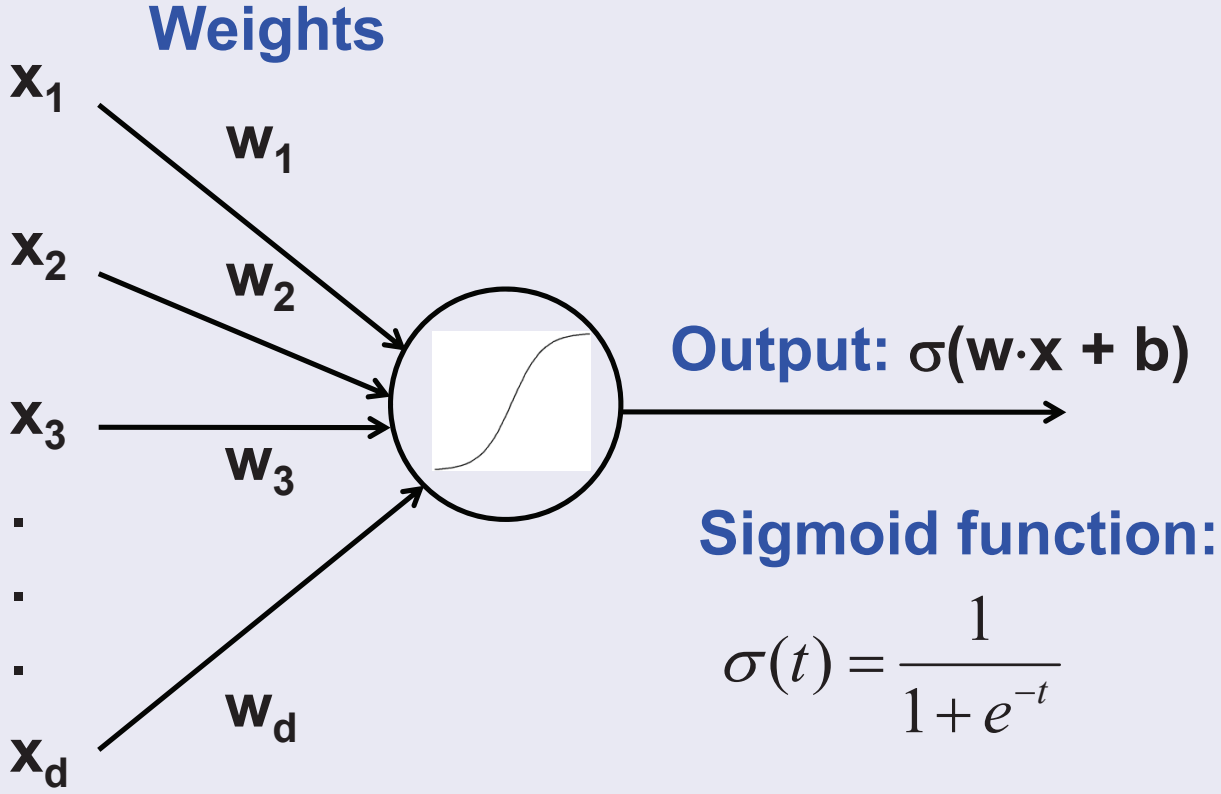
- MaxExp is derived from the Binomial distribution model with two events for (ϕ_n), that is events $(\phi_n = 1)$ and $(\phi_n = 0)$ which model the presence and absence of a feature. The probability of at least one feature occurrence $(\phi_n = 1)$ in N trials becomes:

$$\sum_{n=1}^{N} \binom{N}{n} p^n (1-p)^{N-n} = 1-(1-p)^N$$

- But, MaxExp given by $1-(1-p)^N$ assumes Bernoulli i.i.d. feature draws. In the reality (as I know it), the features are never independent.
- We showed that a simple relaxation $1-(1-p)^\eta$ where $\eta \approx N$ can 'correct' overestimated cummulant p (thus removing the i.i.d. assumption).
- But what η in $1-(1-p)^\eta$ and what γ in p^γ is best? Do you know to what degree your features are dependent? Do you know to what degree 'count' vs. 'detection' correlates with your class label? No.

Analytical pooling operators

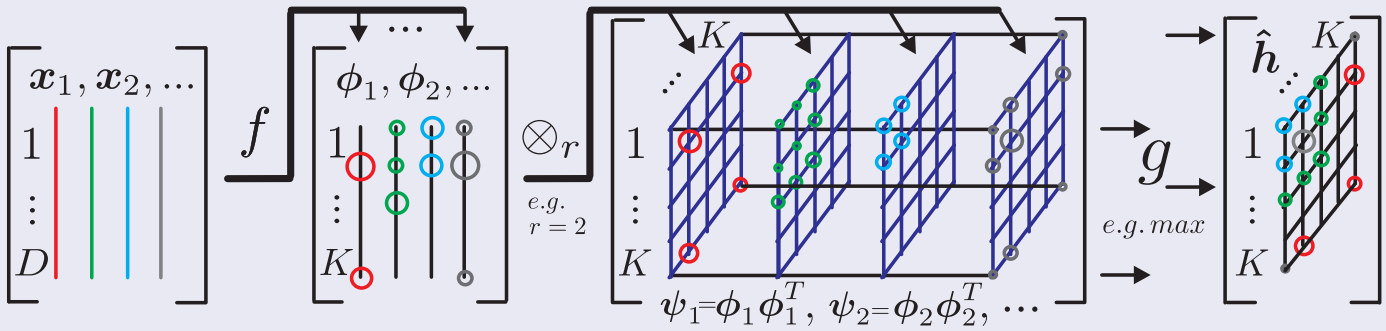
- CNNs are not that different, *e.g.* Perceptrons use point-wise convolution (linear operation) followed by a non-linearity.



Turns out pooling operators and non-linearity, *e.g.* Sigmoid, are the same (or a very similar) thing. But more about that later on!
[Koniusz *et al.*, A Deeper Look at Power Normalizations (CVPR'18)]

Higher-order Occurrence Pooling

- Use second-order statistics with $\uparrow\otimes_r$, e.g. $\uparrow\otimes_2\phi = \phi\phi^T$:



- Formally, this can be expressed in four steps:

$$\phi_n = f(\boldsymbol{x}_n, \mathbf{D}), \quad \forall n \in \mathcal{N} \qquad \text{(encode/zero-center or not)} \tag{4}$$

$$\psi_n = \otimes_r \phi_n \qquad \text{(co-occurrences)} \tag{5}$$

$$\psi_n := u_{\text{:}}(\psi_n) \qquad \text{(vectorise or not)} \tag{6}$$

$$\hat{h}_k = g\left(\{\psi_{kn}\}_{n\in\mathcal{N}}\right) \qquad \text{(pool with some Power Norm or not)} \tag{7}$$

$$\boldsymbol{h} = \hat{\boldsymbol{h}}/\|\hat{\boldsymbol{h}}\|_2 \qquad \text{(normalise or not)} \tag{8}$$

- HO reduces uncert. of Max-pooling: VOC07 $\sim$ 70% vs. 65% FV.
- TO better than SO? Depends on the complexity of the pipeline.

Pretty much all modern bilinear pooling pipelines with CNNs are variations of the above pipeline. Alas, many recent papers conveniently avoid citing the Eigenvalue (Spectral) Power Normalization which we proposed first in 2013, or they even rebrand it...
[Koniusz *et al.*, Higher-order occurrence pooling on mid-and low-level features: Visual concept detection (HAL archives 2013)]
[Koniusz *et al.*, Higher-order Occurrence Pooling for Bags-of-Words: Visual Concept Detection (TPAMI, 2016)]

Connection to non-Euclidean distances

- But, before that we had Covariance Region Descriptors (CRD).
  [O. Tuzel *et al.*, Region covariance: A fast descriptor for detection and classification (ECCV, 2006)].
- They were often used with Riemannian-inspired distances (not Power Norm $\mathbf{\Sigma}^{\gamma}$ but *e.g.* , the Log-Euclidean $\log(\mathbf{\Sigma})$ map.)
  [Carreira *et al.*, Semantic Segmentation with Second-Order Pooling (ECCV, 2012)]
- Also, a bunch of non-Euclidean distances which 'dislike' semi-definite matrices but 'love' the geodesic path on the cone:

$$||\operatorname{Log}(\mathbf{\Sigma}) - \operatorname{Log}(\mathbf{\Sigma}^*)||_F \qquad \text{(Log-Euclidean)} \tag{9}$$

[Arsigny *et al.*, Log-euclidean metrics for fast and a simple calculus on diffusion tensors (Magnetic Resonance in Medicine, 2006)]

$$||\operatorname{Log}(\mathbf{\Sigma}^{-\frac{1}{2}}\mathbf{\Sigma}^*\mathbf{\Sigma}^{-\frac{1}{2}})||_F \qquad \text{(Affine Invariant Reimmannian Metric)} \tag{10}$$

[Pennec *et al.*, A Riemannian Framework for Tensor Computing (IJCV, 2006)]

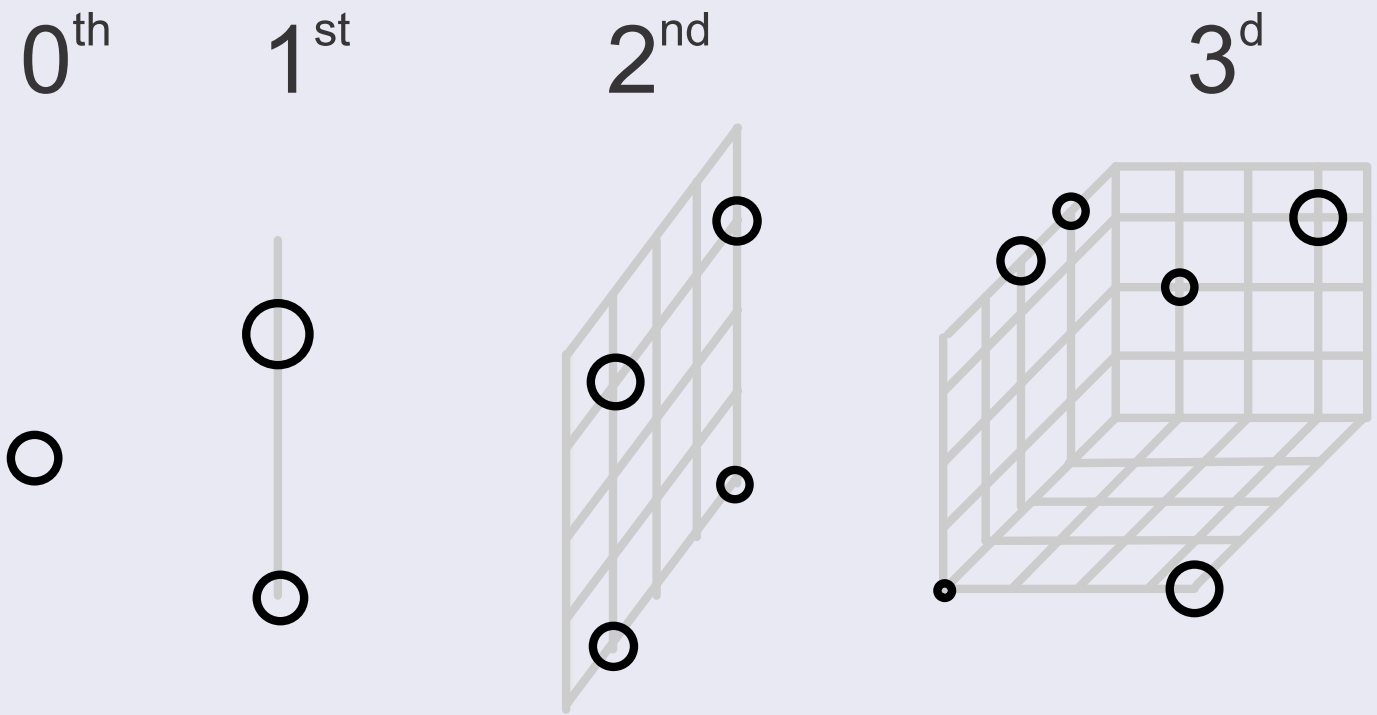
$$\frac{1}{\gamma}||\mathbf{\Sigma}^{\gamma} - \mathbf{\Sigma}^{*\gamma}||_F \qquad \text{(Power-Euclidean)} \tag{11}$$

[Dryden *et al.*, Non-euclidean statistics for covariance matrices, with applications to diffusion tensor imaging (Annals of Applied Statistics, 2009]

Only Power-Euclidean dist. including Matrix Square Root $||\operatorname{Sqrt}(\mathbf{\Sigma}) - \operatorname{Sqrt}(\mathbf{\Sigma}^*)||_F$ and Cholesky $||\operatorname{Chol}(\mathbf{\Sigma}) - \operatorname{Chol}(\mathbf{\Sigma}^*)||_F$ 'like' semi-definite matrices and they reduce burstiness, although they were not designed for that purpose nor for image classification. We showed that connection first in our HAL paper, 2013.

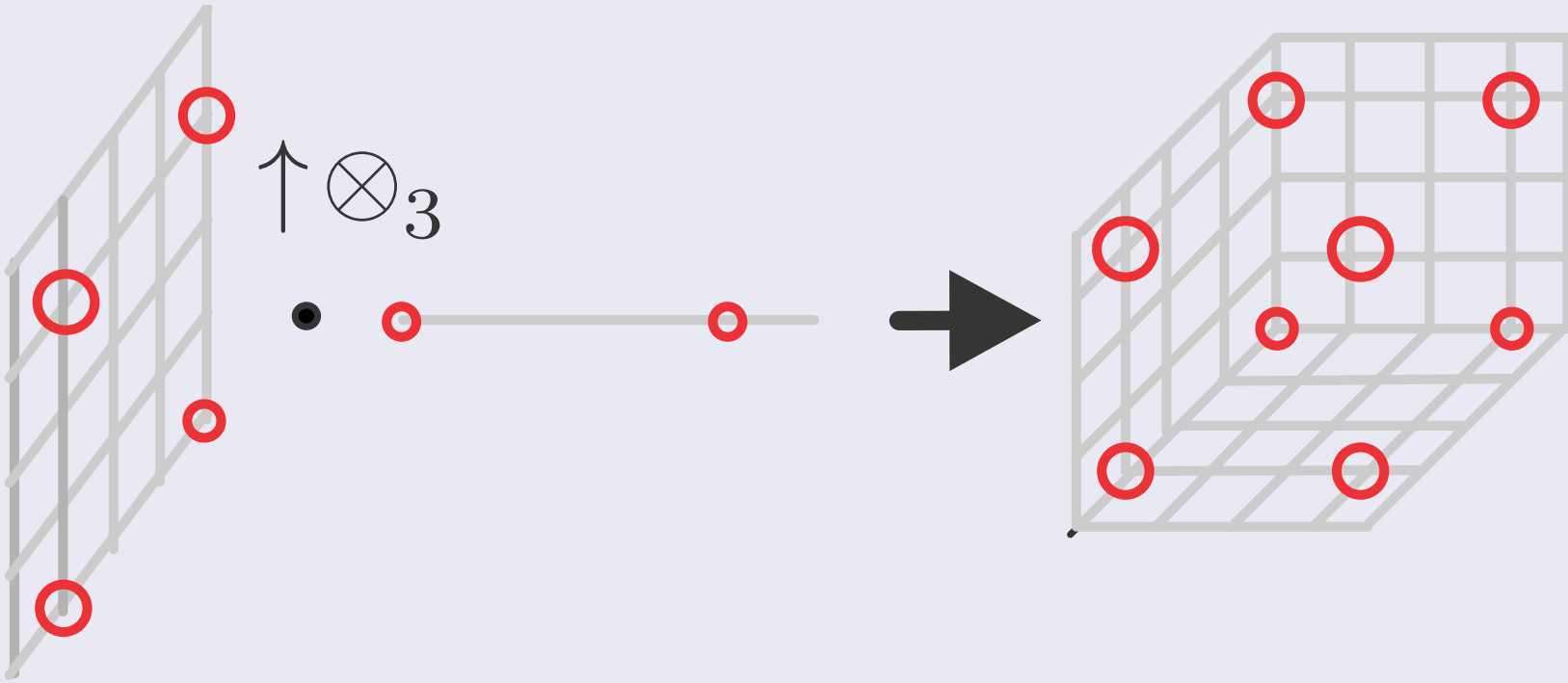
What are tensors?

- Scalars, vectors, matrices, third-order tensors such that $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$, and higher-orders such that $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_1 \times N_2 \times N_3 \times \ldots \times N_K}$.



Tensor outer product (simplified)

- Outer product of three vectors $\mathbf{u}$, $\mathbf{v}$, $\mathbf{z}$ can be written as $X_{mno} = u_m v_n z_o$ and $\mathcal{X} = \mathbf{u}\mathbf{v}^T \uparrow \otimes_3 \mathbf{z}$.



- Outer product of order r of a vector $\mathbf{u}$ with itself can be written as $X_{mno\ldots z} = u_m u_n u_o \cdot \ldots \cdot u_z$ and $\mathcal{X} = \uparrow \otimes_r \mathbf{u}$.

Why tensors at all?

- To compare images, we want to capture distribution of their patches/local descriptors/local receptive fields in CNN *etc.*
- The characteristic function $\varphi_x(\boldsymbol{\omega}) = \mathbb{E}_X\left(\exp(i\boldsymbol{\omega}^T\mathbf{x})\right)$ describes the probability density $f_X(\mathbf{x})$ of an image (patches $\mathbf{x} \sim X$).
- This gives us the following Taylor expansion:

$$\mathbb{E}_X\left(\sum_{r=0}^{\infty}\frac{i^j}{r!}\left\langle \mathbf{x},\boldsymbol{\omega}\right\rangle^r\right) \approx \frac{1}{N}\sum_{n=0}^{N}\sum_{N}^{\infty}\frac{i^r}{r!}\left\langle \uparrow\otimes_r\mathbf{x}_n, \uparrow\otimes_r\boldsymbol{\omega}\right\rangle, \tag{12}$$

$$\sum_{r=0}^{\infty}\frac{i^r}{r!}\left\langle \frac{1}{N}\sum_{n=0}^{N}\uparrow\otimes_r\mathbf{x}_n, \uparrow\otimes_r\boldsymbol{\omega}\right\rangle = \sum_{r=0}^{\infty}\left\langle \boldsymbol{\mathcal{X}}^{(r)}, \frac{i^r}{r!}\uparrow\otimes_r\boldsymbol{\omega}\right\rangle, \tag{13}$$

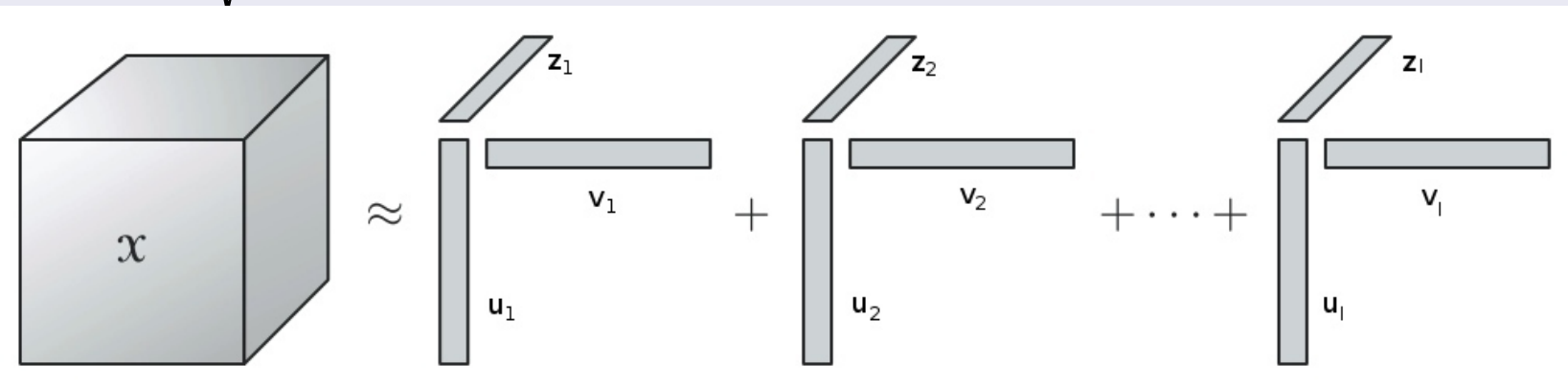
where our tensor descriptor $\boldsymbol{\mathcal{X}}^{(r)} = \frac{1}{N}\sum\limits_{n=0}^{N}\uparrow\otimes_r\mathbf{x}_n$.

SPD matrix+SVD

- For SPD matrices $\mathbf{X} \in \mathcal{S}_{++}^{N}$, we have eigenvalue decomposition such that $\mathbf{X} = \boldsymbol{U}\boldsymbol{\lambda}\boldsymbol{U}^T = \sum_i \lambda_{ii}\mathbf{u}_i\mathbf{u}_i^T$ where $\boldsymbol{U}^T\boldsymbol{U} = \mathbb{I}$ and $\lambda_{ii} > 0$.
- For any matrices $\mathbf{X} \in \mathbb{R}^{N_1 \times N_2}$, we have singular value decomposition such that $\mathbf{X} = \boldsymbol{U}\boldsymbol{\lambda}\boldsymbol{V}^T = \sum_i \lambda_{ii}\mathbf{u}_i\mathbf{v}_i^T$ where $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{V}^T\boldsymbol{V} = \mathbb{I}$.

Tensor properties

- For higher-order tensors, there exist many decompositions, *e.g.* CP (CANDECOMP/PARAFAC) given as $\boldsymbol{\mathcal{X}} = \sum_i^I \lambda_i \mathbf{u}_i \mathbf{v}_i^T \uparrow \otimes_3 \mathbf{z}_i$ and generally $\boldsymbol{U}^T\boldsymbol{U} \neq \boldsymbol{V}^T\boldsymbol{V} \neq \boldsymbol{Z}^T\boldsymbol{Z} \neq \mathbb{I}$.
- There exist also many definitions of tensor rank, notably the smallest possible I in the decomposition above such that the equality holds.
- Also, I is upper bounded, that is: $I \leq \min(N_1 N_2, N_1 N_3, N_2 N_3)$.
- For third-order, the Frobenius norm is given by $\|\boldsymbol{\mathcal{X}}\| = \sqrt{\sum_{m,n,o} X_{m,n,o}^2}$.
- For third-order, the dot-product is given by $\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sqrt{\sum_{m,n,o} X_{m,n,o} \cdot Y_{m,n,o}}$.

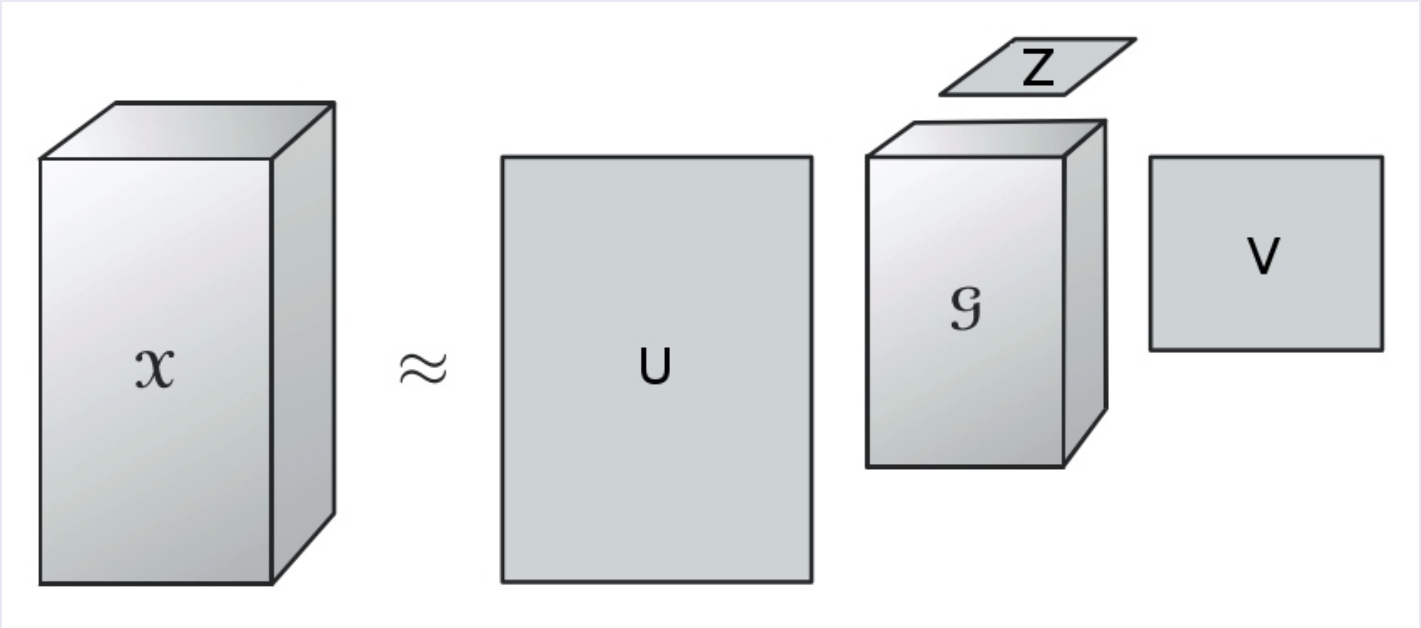


Tensor properties

- Another popular decomposition called Higher Order Singular Value Decomposition is given as $\boldsymbol{\mathcal{X}} = \sum_{m,n,o} \mathcal{G}_{mno} \mathbf{u}_m \mathbf{v}_n^T \uparrow \otimes_3 \mathbf{z}_o$ and $\boldsymbol{U}^T \boldsymbol{U} = \boldsymbol{V}^T \boldsymbol{V} = \boldsymbol{Z}^T \boldsymbol{Z} = \mathbb{I}$.
- $\boldsymbol{\mathcal{G}}$ is known as so-called core tensor (equivalent of singular value matrix in SVD but non-diagonal values are typically non-zero too).
- $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{Z}$ are known as factor matrices. They are obtained by flattening tensor in each mode and regular left-hand side singular vectors of SVD form $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{Z}$, respectively.
- An important concept is the so-called *n-mode product*. Assume $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_1 \times \ldots \times N_K}$ and matrix $\boldsymbol{U} \in \mathbb{R}^{J \times N_I}$, then: $(\boldsymbol{\mathcal{X}} \times_n \boldsymbol{U})_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_K} = \sum_{i_n=1}^{N_n} x_{i_1,\ldots,\imath_n,\ldots,i_K} \cdot u_{j,i_n}$.

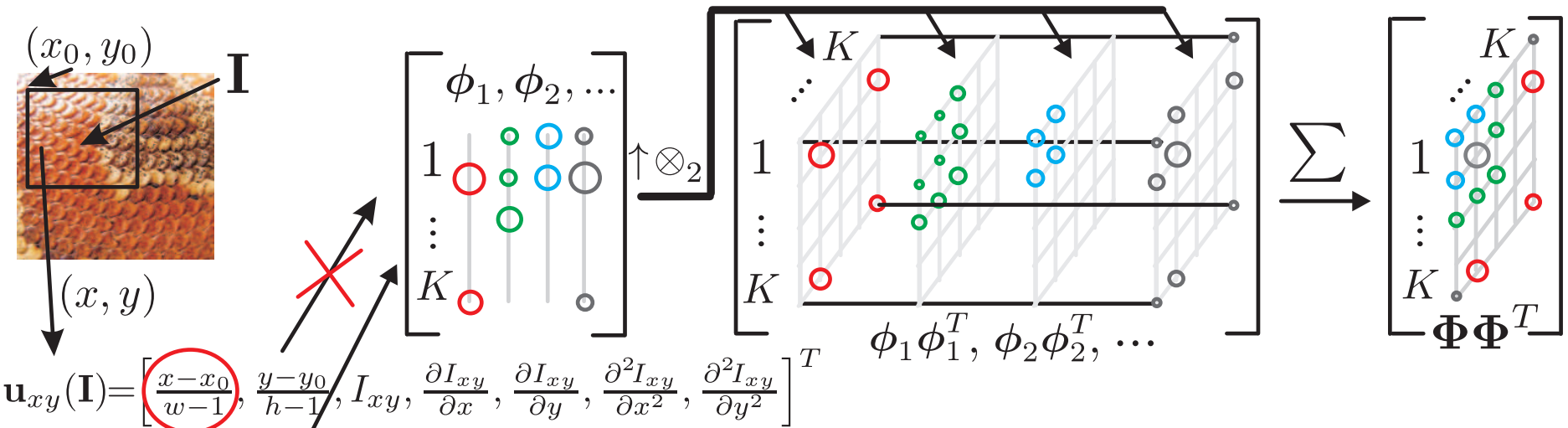
Tensor properties

- It follows that $\mathcal{X} = \mathcal{G} \times_1 \boldsymbol{U} \times_2 \boldsymbol{V} \times_3 \boldsymbol{Z}$ and
- $\mathcal{G} = \mathcal{X} \times_1 \boldsymbol{U}^{-1} \times_2 \boldsymbol{V}^{-1} \times_3 \boldsymbol{Z}^{-1} = \mathcal{X} \times_1 \boldsymbol{U}^T \times_2 \boldsymbol{V}^T \times_3 \boldsymbol{Z}^T$.



TOSST Texture Descriptors

- Region covariance descriptors (co-occurrences) use the outer-product of low-level feature vectors $\uparrow\otimes_2\mathbf{u} = \mathbf{u}\mathbf{u}^T$.
- Let us have non-linear co-occurrence descriptors.



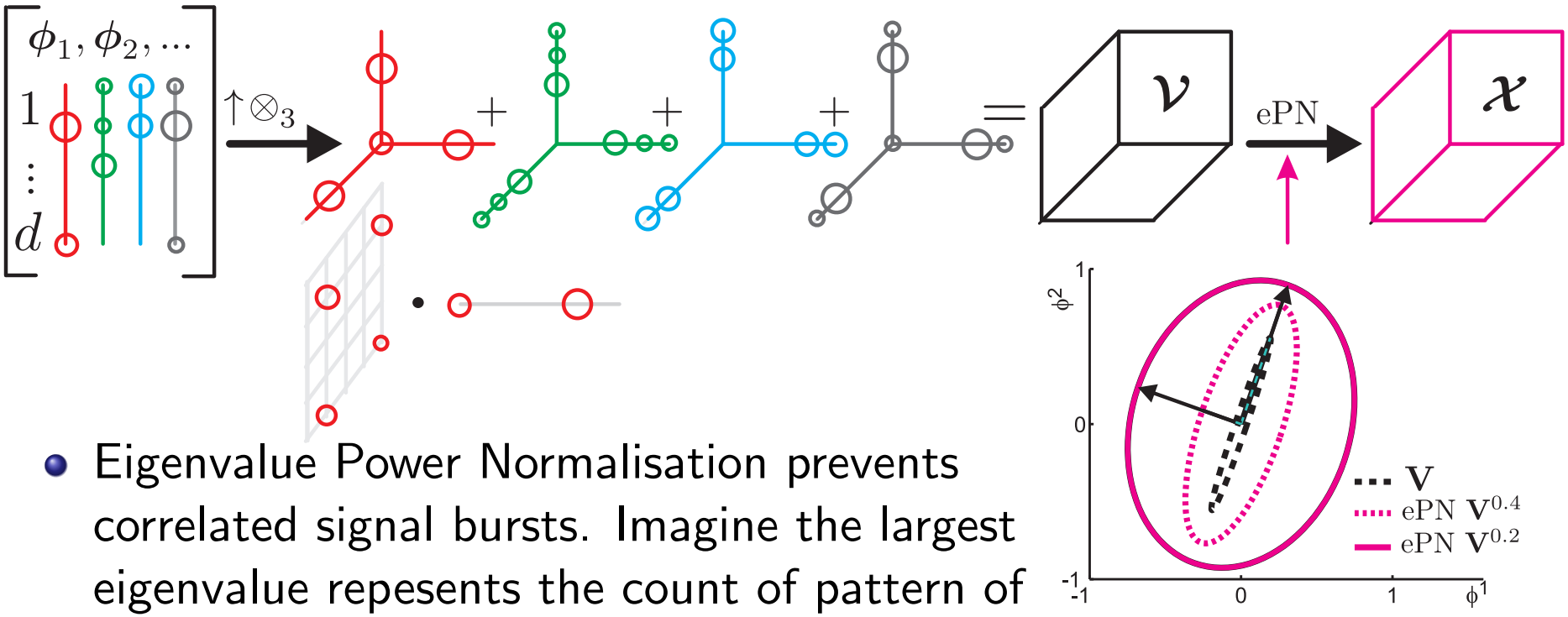
- Can we form more informative co-occurrences?

  Yes, we extend $\uparrow\otimes_2$ to third-order outer product $\uparrow\otimes_3$

  [Koniusz, Cherian, Sparse Coding for Third-order Super-symmetric Tensor Descriptors with Application to Texture Recognition (CVPR, 2016)]

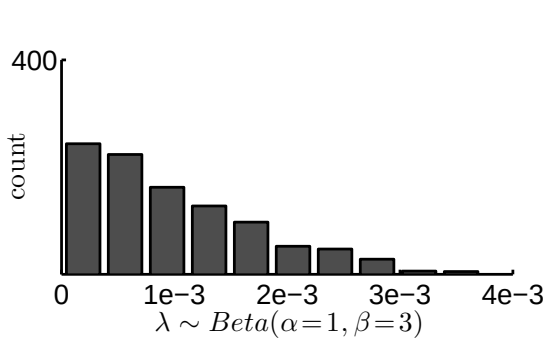
TOSST Texture Descriptors

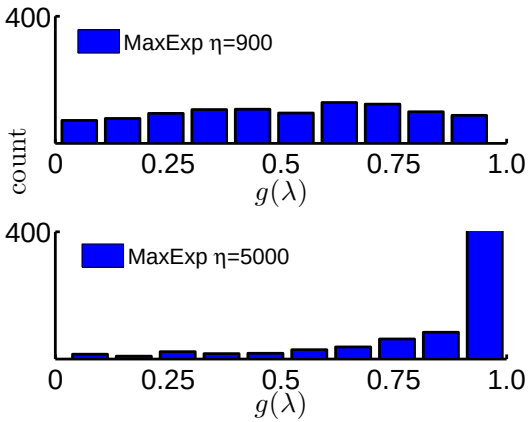
- Non-linear third-order descriptors +eigenvalue Power Normalization (ePN).

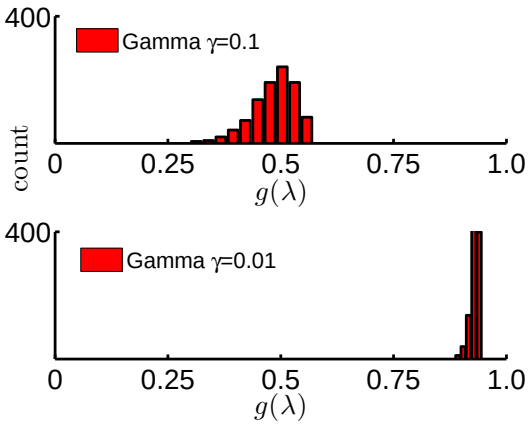


- Eigenvalue Power Normalisation prevents correlated signal bursts. Imagine the largest eigenvalue repesents the count of pattern of brick and the 2nd represents the tree bark. Surely, the amount of brick and bark patterns should not affect the prediction. But it does!
- Higher-order models can be derived analytically.

Push-forward distribution of PN







Initial spectral dist. Push-forward (MaxExp) Push-forward (Gamma)

- Thus, MaxExp and Gamma do a similar job, but we will show that MaxExp has very fast non-SVD based backprop rules.
- MaxExp is even faster than the matrix square root via Newton-Schulz iterations... So why the obsession with the matrix square root?

Higher-order Occurrence Pooling: derivation

- Assume a kernel, e.g. RBF and its linearisation given by: $ker\left(\mathbf{u}, \bar{\mathbf{u}}\right) \approx \left\langle \phi, \bar{\phi} \right\rangle$.
- Assume the dot product $\left\langle \phi, \bar{\phi} \right\rangle$ on a pair of features and polynomial kernel: $\left\langle \phi, \bar{\phi} \right\rangle^r$, $r \geq 2$.
- Define a sum kernel between two sets of features $\boldsymbol{U} = \{\mathbf{u}_n\}_{n \in \mathcal{N}}$ and $\bar{\boldsymbol{U}} = \{\bar{\mathbf{u}}_{\bar{n}}\}_{\bar{n} \in \bar{\mathcal{N}}}$ for two images/regions/sequences (anything you like):

$$
\begin{aligned}
Ker\left(\boldsymbol{U}, \bar{\boldsymbol{U}}\right) &= \frac{1}{|\mathcal{N}|} \sum_{n \in \mathcal{N}} \frac{1}{|\bar{\mathcal{N}}|} \sum_{\bar{n} \in \bar{\mathcal{N}}} ker\left(\mathbf{u}_n, \bar{\mathbf{u}}_{\bar{n}}\right)^r \\
&\approx \frac{1}{|\mathcal{N}|} \sum_{n \in \mathcal{N}} \frac{1}{|\bar{\mathcal{N}}|} \sum_{\bar{n} \in \bar{\mathcal{N}}} \left\langle \phi_n, \bar{\phi}_{\bar{n}} \right\rangle^r \\
&= \frac{1}{|\mathcal{N}|} \sum_{n \in \mathcal{N}} \frac{1}{|\bar{\mathcal{N}}|} \sum_{\bar{n} \in \bar{\mathcal{N}}} \left(\sum_{k=1}^{K} \phi_{kn} \bar{\phi}_{k\bar{n}} \right)^r
\end{aligned} \tag{14}
$$

Higher-order Occurrence Pooling: derivation

- The rightmost summation can be re-expressed as a dot product of two outer-products of order r on ϕ:

$$\left(\sum_{k=1}^{K}\phi_{kn}\bar{\phi}_{k\bar{n}}\right)^{r}=\sum_{k^{(1)}=1}^{K}\cdots\sum_{k^{(r)}=1}^{K}\phi_{k^{(1)}}\bar{\phi}_{k^{(1)}}\cdot\ldots\cdot\phi_{k^{(r)}}\bar{\phi}_{k^{(r)}}=\left\langle\otimes_{r}\boldsymbol{\phi}_{n},\otimes_{r}\bar{\boldsymbol{\phi}}_{\bar{n}}\right\rangle_{F} \tag{15}$$

- Now, the problem is further simplified:

$$Ker\left(\boldsymbol{U},\bar{\boldsymbol{U}}\right)\approx Ker'\left(\boldsymbol{\Phi},\bar{\boldsymbol{\Phi}}\right)=\frac{1}{|\mathcal{N}|}\sum_{n\in\mathcal{N}}\frac{1}{|\bar{\mathcal{N}}|}\sum_{\bar{n}\in\bar{\mathcal{N}}}\left\langle\otimes_{r}\boldsymbol{\phi}_{n},\otimes_{r}\bar{\boldsymbol{\phi}}_{\bar{n}}\right\rangle_{F}$$

$$=\left\langle\frac{1}{|\mathcal{N}|}\sum_{n\in\mathcal{N}}\otimes_{r}\boldsymbol{\phi}_{n},\frac{1}{|\bar{\mathcal{N}}|}\sum_{\bar{n}\in\bar{\mathcal{N}}}\otimes_{r}\bar{\boldsymbol{\phi}}_{\bar{n}}\right\rangle_{F}=\left\langle\underset{n\in\mathcal{N}}{\mathrm{Avg}}\left(\otimes_{r}\boldsymbol{\phi}_{n}\right),\underset{\bar{n}\in\bar{\mathcal{N}}}{\mathrm{Avg}}\left(\otimes_{r}\bar{\boldsymbol{\phi}}_{\bar{n}}\right)\right\rangle_{F}$$

- We introduce operator $\mathcal{G}$ (similarity for matrices/tensors, e.g. ePN):

$$Ker^{*}\left(\boldsymbol{\Phi},\bar{\boldsymbol{\Phi}}\right)=\left\langle\mathcal{G}\left(\frac{1}{|\mathcal{N}|}\sum_{n\in\mathcal{N}}\otimes_{r}\boldsymbol{\phi}_{n}\right),\mathcal{G}\left(\frac{1}{|\bar{\mathcal{N}}|}\sum_{\bar{n}\in\bar{\mathcal{N}}}\otimes_{r}\bar{\boldsymbol{\phi}}_{\bar{n}}\right)\right\rangle_{F} \tag{16}$$

Sparse Coding for Third-order Tensor Descriptors

- However, $\mathcal{X}$ is cubic w.r.t. size of features.
  We propose Sparse Coding for Third-order Tensor Descriptors.
- We can learn a dictionary to encode TOSST:

$$\underset{\substack{\mathcal{B}_1,\ldots,\mathcal{B}_K \\ \boldsymbol{\alpha}^1,\ldots,\boldsymbol{\alpha}^N}}{\arg\min} \sum_{n=1}^{N} \left\| \mathcal{X}_n - \sum_{k=1}^{K} \mathcal{B}_k \alpha_k^n \right\|_F^2 + \lambda \left\| \boldsymbol{\alpha}^n \right\|_1. \tag{17}$$

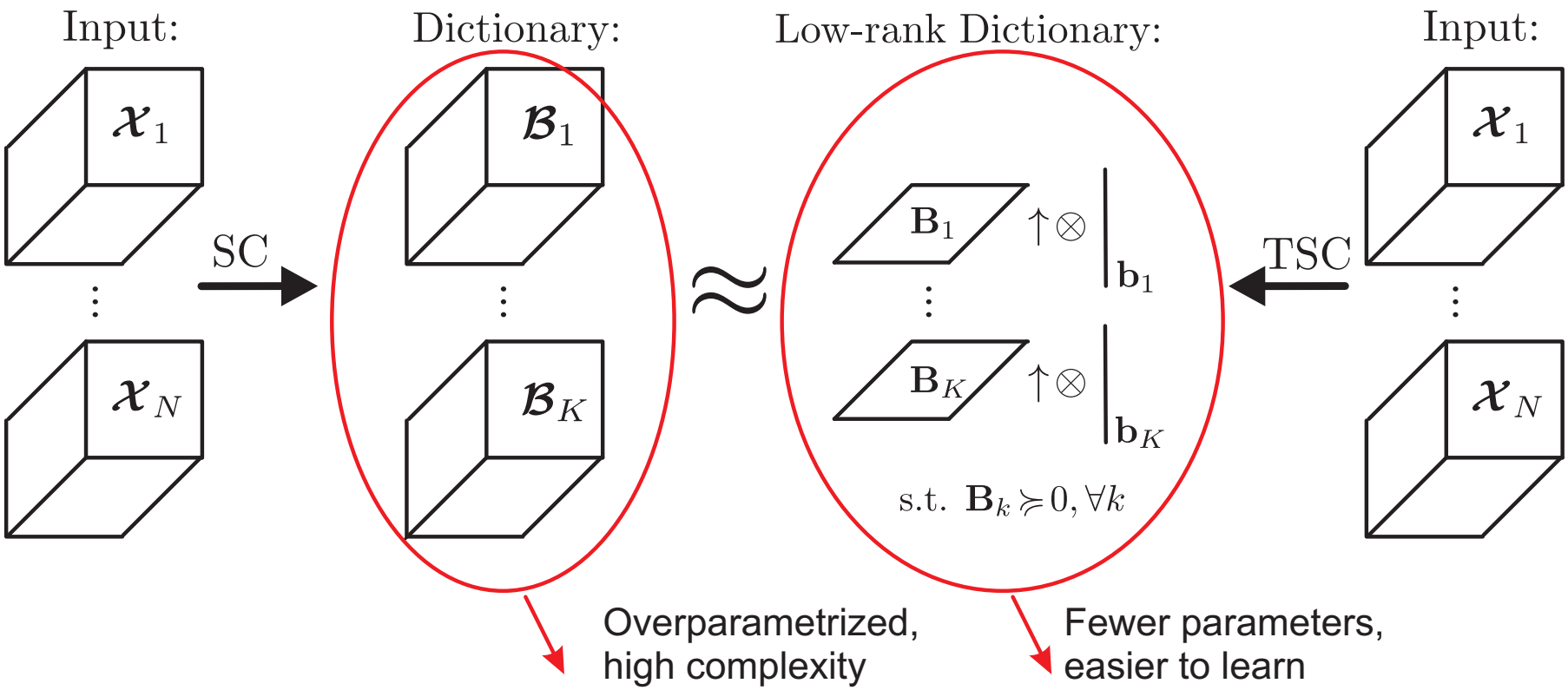
- However, $\mathcal{B}$ have three modes (overparametrised model), so we learn instead low-rank dictionary:

$$\underset{\substack{\mathbf{B}_1,\ldots,\mathbf{B}_K \\ \mathbf{b}_1,\ldots,\mathbf{b}_K \\ \boldsymbol{\alpha}^1,\ldots,\boldsymbol{\alpha}^N}}{\arg\min} \sum_{n=1}^{N} \left\| \mathcal{X}_n - \sum_{k=1}^{K} \left(\mathbf{B}_k \uparrow \otimes \mathbf{b}_k \right) \alpha_k^n \right\|_F^2 + \lambda \left\| \boldsymbol{\alpha}^n \right\|_1. \tag{18}$$

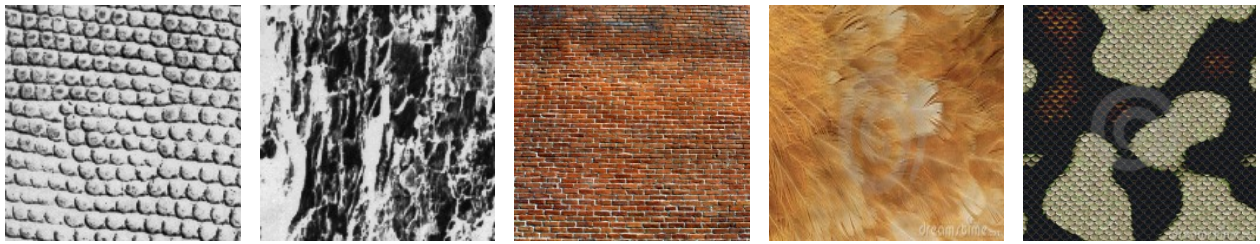
- Resulting sparse codes $\boldsymbol{\alpha}$ are pooled and used for SVM training.

Sparse Coding for Third-order Tensor Descriptors

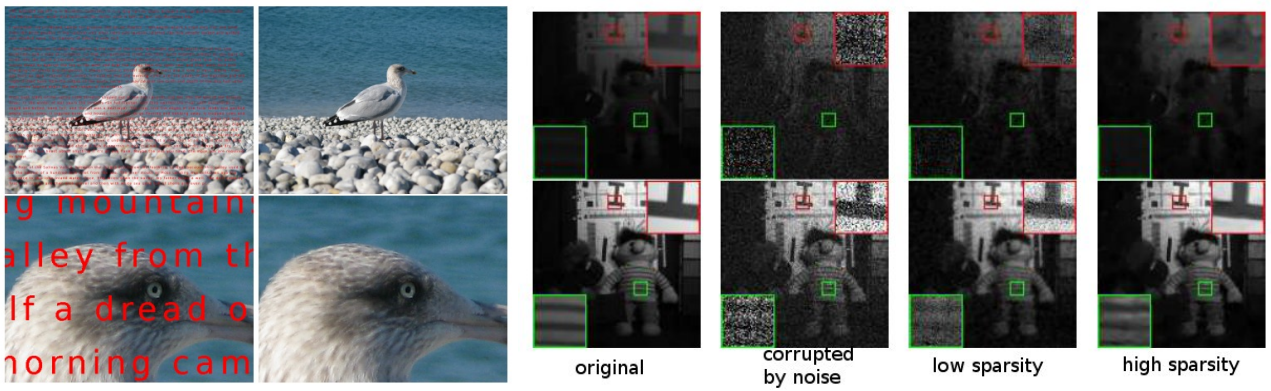
- We use training set of TOSST descriptors $\boldsymbol{\mathcal{X}}_1, ..., \boldsymbol{\mathcal{X}}_N$.
- We learn low-rank dictionary atoms $\mathbf{B}_1 \uparrow\otimes \mathbf{b}_1, ..., \mathbf{B}_K \uparrow\otimes \mathbf{b}_K$ (outer product of matrices with vectors).
- They approximate full-rank tensor atoms $\boldsymbol{\mathcal{B}}_1, ..., \boldsymbol{\mathcal{B}}_K$.



Results

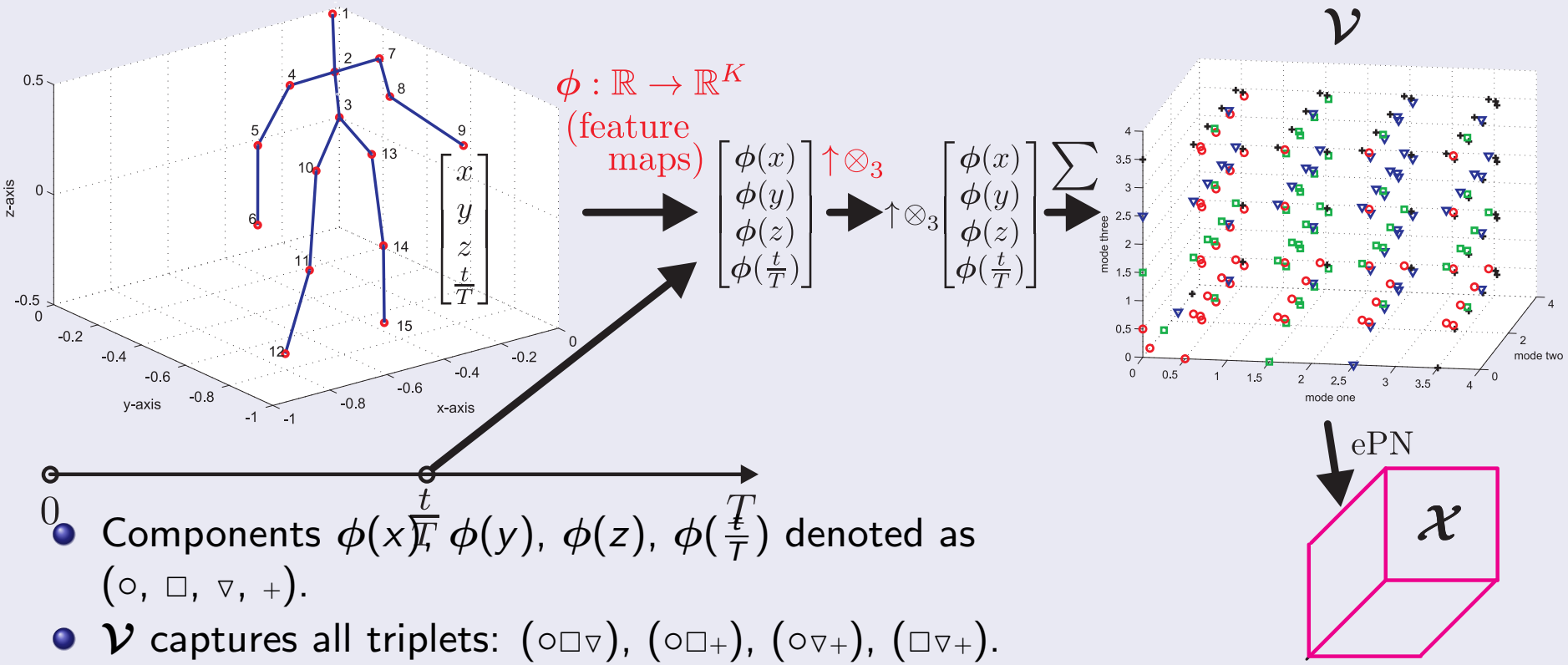


- Brodatz textures; 99.9% accuracy (the state of the art); others score $\sim 98.72\%$.
- UIUC materials recognition; 58.0% accuracy.
- PASCAL VOC07 descriptor compression: 61.2% mAP ($25K$ signature) vs. 61.3% mAP ($176K$ signature).
- Image/video denoising, *e.g.* $\arg\min_{\mathcal{G}} ||\mathcal{X} - \hat{\mathcal{X}}||_F^2 + \Omega(\mathcal{X} - \hat{\mathcal{X}})$ s.t. $\hat{\mathcal{X}} = \mathcal{G} \times_1 \boldsymbol{U} \times_2 \boldsymbol{V} \times_3 \mathbf{Z}$:



Action Recognition from 3D Skeletons

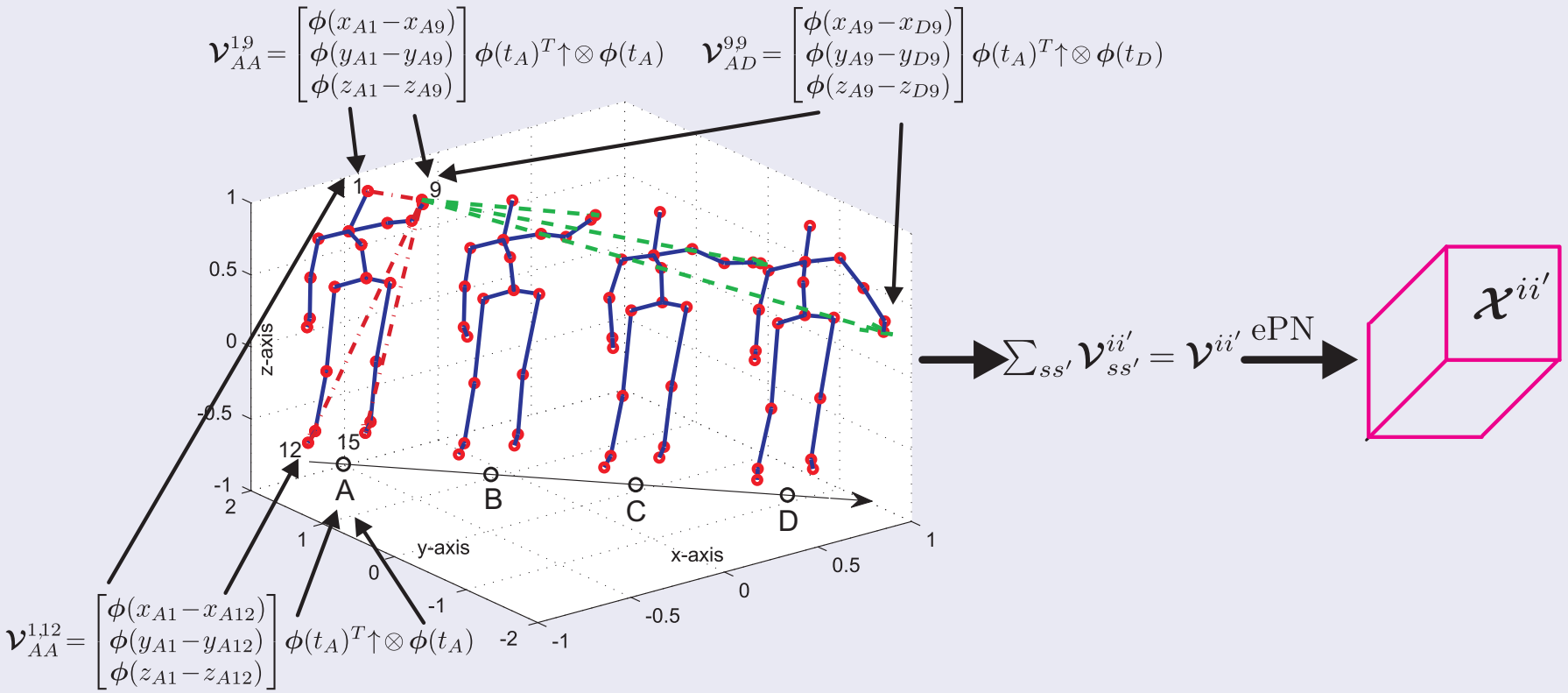
Sequence Compatibility Kernel



- Components $\phi(x)$, $\phi(y)$, $\phi(z)$, $\phi(\frac{t}{T})$ denoted as $(\circ, \square, \triangledown, +)$.
- $\mathcal{V}$ captures all triplets: $(\circ\square\triangledown)$, $(\circ\square+)$, $(\circ\triangledown+)$, $(\square\triangledown+)$.
- ePN evens out counts of these co-occurrences.
- Tensors $\mathcal{X}$ are the samples for training SVM.
  [Koniusz *et al.*, Tensor Representations via Kernel Linearlization for Action Recognition from 3D Skeletons (ECCV, 2016)]

Action Recognition from 3D Skeletons

Dynamics Compatibility Kernel



- Enumerate all unique joint displacement vectors $\mathbf{x}_{it} - \mathbf{x}_{jt'}, i \leq j, t \leq t'$.
- Embed displacements into RKHS and linearise to obtain $\phi(\mathbf{x}_{it} - \mathbf{x}_{jt'})$.
- Embed start-/end-times into RKHS, linearise to obtain $\phi(\frac{t}{T})$, $\phi(\frac{t'}{T})$.
- Take outer products $\phi(\mathbf{x}_{it} - \mathbf{x}_{jt'})\phi(\frac{t}{T})\uparrow\otimes\phi(\frac{t'}{T})$, aggregate+ePN.

Natural Inner Product on Gaussians

- Gaussian kernel between $\mathbf{u} \in \mathbb{R}^{d'}$ and $\bar{\mathbf{u}} \in \mathbb{R}^{d'}$ can be rewritten as:

$$G_\sigma(\mathbf{u}-\bar{\mathbf{u}}) = e^{-\|\mathbf{u}-\bar{\mathbf{u}}\|_2^2/2\sigma^2} = \left(\frac{2}{\pi\sigma^2}\right)^{\frac{d'}{2}} \int_{\boldsymbol{\zeta}\in\mathbb{R}^{d'}} G_{\sigma/\sqrt{2}}(\mathbf{u}-\boldsymbol{\zeta})\, G_{\sigma/\sqrt{2}}(\bar{\mathbf{u}}-\boldsymbol{\zeta})\, \mathrm{d}\boldsymbol{\zeta}. \tag{19}$$

- Finite approximation by $\boldsymbol{\zeta}_1, ..., \boldsymbol{\zeta}_Z$ pivots is given by:

$$\boldsymbol{\phi}(\mathbf{u}) = \left[G_{\sigma/\sqrt{2}}(\mathbf{u}-\boldsymbol{\zeta}_1), ..., G_{\sigma/\sqrt{2}}(\mathbf{u}-\boldsymbol{\zeta}_Z)\right]^T, \tag{20}$$

$$\text{and } G_\sigma(\mathbf{u}-\bar{\mathbf{u}}) \approx \left\langle \frac{\boldsymbol{\phi}(\mathbf{u})}{\|\boldsymbol{\phi}(\mathbf{u})\|_2}, \frac{\boldsymbol{\phi}(\bar{\mathbf{u}})}{\|\boldsymbol{\phi}(\bar{\mathbf{u}})\|_2} \right\rangle. \tag{21}$$

- As few as 6 pivots yield $\leq$0.8% approximation error.

Eigenvalue Power Normalisation

- Four simple steps in MATLAB:

$$(\boldsymbol{\mathcal{E}}; \boldsymbol{A}_1, ..., \boldsymbol{A}_r) = \text{HOSVD}(\boldsymbol{\mathcal{V}}) \tag{22}$$

$$\hat{\boldsymbol{\mathcal{E}}} = \text{Sgn}(\boldsymbol{\mathcal{E}}) \, |\boldsymbol{\mathcal{E}}|^{\gamma} \tag{23}$$

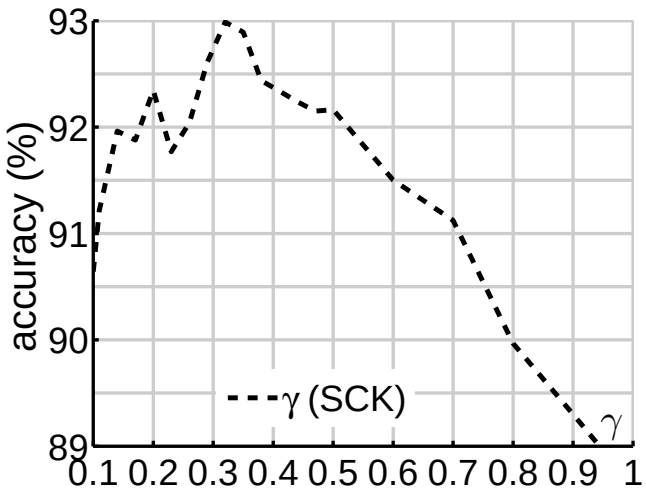
$$\hat{\boldsymbol{\mathcal{V}}} = ((\hat{\boldsymbol{\mathcal{E}}} \otimes_1 \boldsymbol{A}_1) ...) \otimes_r \boldsymbol{A}_r \tag{24}$$

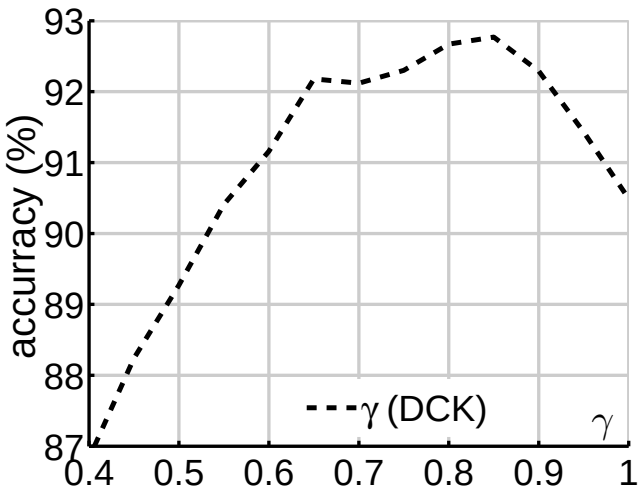
$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{V}}) = \text{Sgn}(\hat{\boldsymbol{\mathcal{V}}}) \, |\hat{\boldsymbol{\mathcal{V}}}|^{\gamma^*} \tag{25}$$

- Perform Higher Order SVD (equiv. of SVD for more than 2 modes).
- Obtain the core tensor $\boldsymbol{\mathcal{E}}$ (equivalent of singular values).
- Power-normalise this spectrum (values $\boldsymbol{\mathcal{E}}$ can be negative).
- Assemble back tensor, if needed, perform additionally standard PN.
- We have now mathematical interpretation and derivations.

Action Recognition from 3D Skeletons: results

- Eigenvalue Power Normalisation w.r.t. γ:





Action Recognition from 3D Skeletons: results

- Florence3D-Action (state-of-the-art):

	SCK	DCK		SCK+DCK
accuracy	92.98%	93.03%	92.77%	**95.47%**
size	26565	9450	16920	43485

Bag-of-Poses 82.00%	$SE(3)$ 90.88%

- UTKinect-Action (state-of-the-art):

	SCK	DCK	SCK+DCK
accuracy	96.08%	97.69%	**98.39%**
size	40480	16920	57400

3D joints hist. 90.92%	$SE(3)$ 97.08%

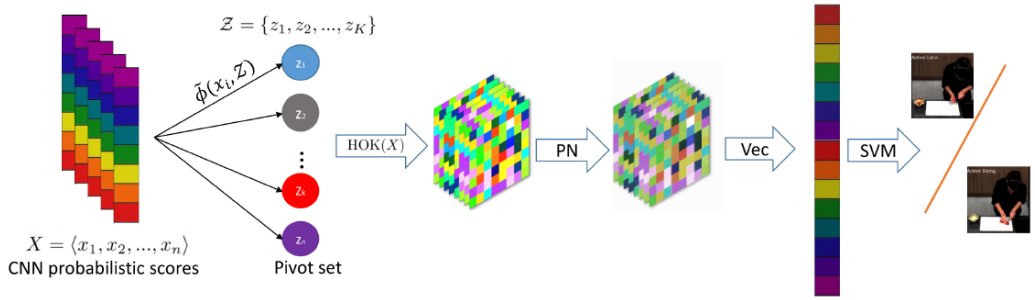
- MSR-Action3D:

	SCK+DCK	$SE(3)$
accuracy, standard protocol	**92.8%**	89.48%
accuracy, specific classes/subjectss	**94.8%**	92.46%
size	57400	-

- NTU: 69% (TO) vs. 62% (SO)

Action Recognition from 3D Skeletons: results

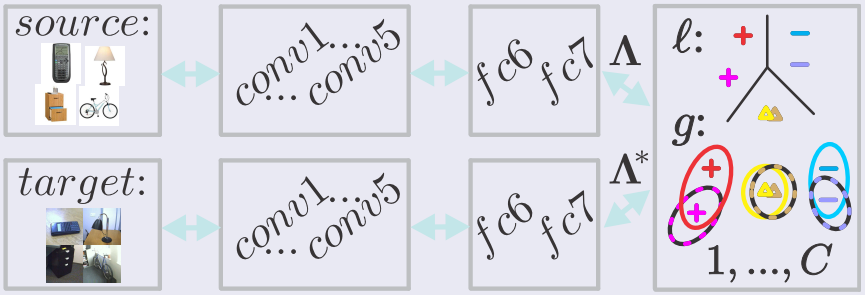
- Also, we extended SCK to run over frame-wise scores from two-stream CNN networks (HOK kernel)



- MPII cooking activities (fine-grained): 73.1%
  [Cherian, Koniusz, Gould, Higher-order Poling of CNN Features via Kernel Linearization for Action Recognition (WACV, 2017)]
- NEW RESULTS. With a new SCK+ kernel, I get:
- MPII cooking activities (fine-grained): $\geq$80.4%
- Florence3D: $\sim$ 97.0% UTKinect: $\geq$99.0%, MSR-Action3D: $\sim$ 97.0
- Large-scale NTU (3D body-joints): $\geq$ 73.0% (TO) vs. $\sim$ 64.0% (SO)
- Large-scale NTU (RGB+3D body-joints): $\geq$ 91.0% accuracy
- Large-scale HMDB51 (RGB+optical-flow): $\sim$87.2% accuracy

Domain Adaptation by Mixture of Alignments of Second- or Higher-Order Scatter Tensors

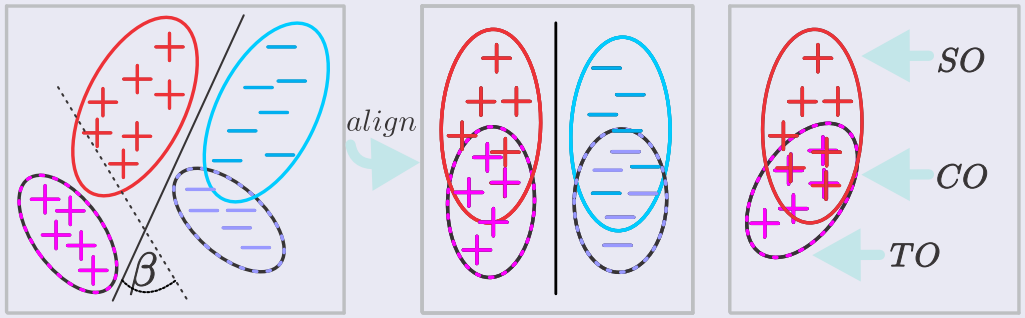
- We focus on the supervised domain adaptation
- Given a common set of labels for the source and target data, we use rich source data (∼30 images per class) to train a robust classifier for the scarce target data (∼3 images per class)
- We utilize two CNN streams $\mathbf{\Lambda}$ and $\mathbf{\Lambda}^*$, one per source and target data, combined at the *fc* level.



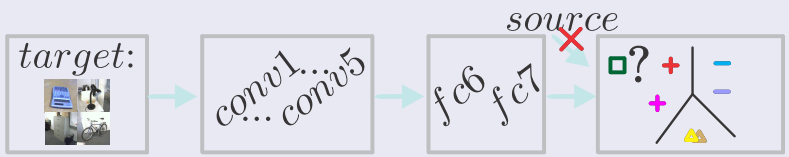
- The main idea is to establish so-called commonality between class-wise statistics

  [Koniusz *et al.*, Domain Adaptation by Mixture of Alignments of Second- or Higher-Order Scatter Tensors (CVPR, 2017)].

Domain Adaptation by Mixture of Alignments of Second- or Higher-Order Scatter Tensors



- Solid/dashed ellipses indicate the source/target domain distributions
- The two hyperplane lines that separate $(+)$ from $(-)$ on the target data indicate large uncertainty (denoted as β) in the optimal orientation for the target problem
- For every class $1, ..., C$, we align within-class scatters from the source and target data to a desired degree by learning weights of alignment
- In the testing stage, we cut-off the source network:



Domain Adaptation: Formulation

- We solve a trade-off between the classifier loss ℓ and the alignment loss g which acts on the scatter tensors $\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{X}}^*$ and related to them means $\boldsymbol{\mu}$, $\boldsymbol{\mu}^*$:

$$\underset{\substack{\mathbf{W},\mathbf{b},\boldsymbol{\Theta},\boldsymbol{\Theta}^* \\ \text{s. t. } ||\boldsymbol{\phi}_n||_2^2 \leq \tau, \\ ||\boldsymbol{\phi}^*_{n'}||_2^2 \leq \tau, \\ \forall n \in \mathcal{I}_N, n' \in \mathcal{I}^*_N}}{\arg\min} \ell(\mathbf{W},\mathbf{b},\boldsymbol{\Lambda} \cup \boldsymbol{\Lambda}^*) + \lambda ||\mathbf{W}||_F^2 + \underbrace{\frac{\sigma_1}{C} \sum_{c \in \mathcal{I}_C} ||\boldsymbol{\mathcal{X}}_c - \boldsymbol{\mathcal{X}}^*_c||_F^2 + \frac{\sigma_2}{C} \sum_{c \in \mathcal{I}_C} ||\boldsymbol{\mu}_c - \boldsymbol{\mu}^*_c||_2^2}_{g(\boldsymbol{\Phi},\boldsymbol{\Phi}^*)}.$$

- We minimize over the CNN parameters of the source and target streams $\boldsymbol{\Theta}$, $\boldsymbol{\Theta}^*$, hyperplane $\mathbf{W}$ and bias $\mathbf{b}$
- We use feature vectors from *fc* in the source network stream, one per image, and associated with them labels. This forms pairs $\boldsymbol{\Lambda} \equiv \{(\boldsymbol{\phi}_n, y_n)\}_{n \in \mathcal{I}_N}$, where $\boldsymbol{\phi}_n \in \mathbb{R}^d$ and $y_n \in \mathcal{I}_C$, $\forall n \in \mathcal{I}_N$
- For the target data, we define pairs $\boldsymbol{\Lambda}^* \equiv \{(\boldsymbol{\phi}^*_n, y^*_n)\}_{n \in \mathcal{I}^*_N}$, where $\boldsymbol{\phi}^* \in \mathbb{R}^d$ and $y^*_n \in \mathcal{I}_C$, $\forall n \in \mathcal{I}^*_N$
- Moreover, class-specific sets of feature vectors are given as $\boldsymbol{\Phi}_c \equiv \{\boldsymbol{\phi}^c_n\}_{n \in \mathcal{I}_{N_c}}$ and $\boldsymbol{\Phi}^*_c \equiv \{\boldsymbol{\phi}^{c*}_n\}_{n \in \mathcal{I}_{N^*_c}}$, $\forall c \in \mathcal{I}_C$. Then, $\boldsymbol{\Phi} \equiv (\boldsymbol{\Phi}_1, ..., \boldsymbol{\Phi}_C)$ and $\boldsymbol{\Phi}^* \equiv (\boldsymbol{\Phi}^*_1, ..., \boldsymbol{\Phi}^*_C)$

Domain Adaptation: From Tensors to Kernels

- We express the costly Frobenius norm between tensors of order r as a tractable kernelized distance which uses kernels $K^r_{nn'} = \langle \mathbf{x}_n - \boldsymbol{\mu}, \mathbf{x}_{n'} - \boldsymbol{\mu} \rangle^r$, $\bar{K}^r_{nn'} = \langle \mathbf{y}_n - \boldsymbol{\mu}^*, \mathbf{y}_{n'} - \boldsymbol{\mu}^* \rangle^r$ and $\bar{\bar{K}}^r_{nn'} = \langle \mathbf{x}_n - \boldsymbol{\mu}, \mathbf{y}_{n'} - \boldsymbol{\mu}^* \rangle^r$:

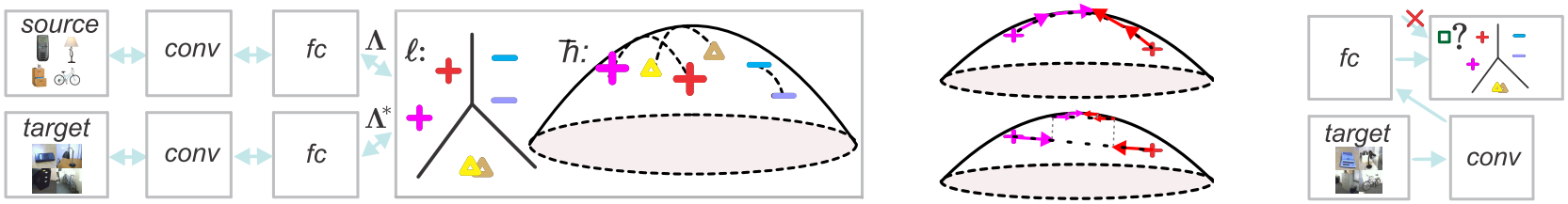
$$||\boldsymbol{\mathcal{X}}^{(r)} - \boldsymbol{\mathcal{X}}^{*(r)}||_F^2 = \frac{1}{N^2} \mathbb{1}^T \mathbf{K}^r \mathbb{1} + \frac{1}{N^{*2}} \mathbb{1}^T \bar{\mathbf{K}}^r \mathbb{1} - \frac{2}{NN^*} \mathbb{1}^T \bar{\bar{\mathbf{K}}}^r \mathbb{1}.$$

- Results on the Office dataset ($\mathcal{A} \rightarrow \mathcal{D}$, VGG streams):

	sp1	sp2	sp3	sp4	sp5	sp6	sp7	sp8	sp9	sp10	aver. acc.
$S+T$	90.6	88.9	89.4	92.4	90.1	87.2	91.1	88.2	90.9	89.4	89.83±1.4
$So+To+Fo+\zeta$	93.1	93.1	92.0	92.7	93.3	89.9	94.1	91.9	94.0	93.4	**92.73±1.1**

Domain Adaptation: Rimannian distances

- We combine the source and target CNN streams:



(a) (b) (c)

DA pipeline:

(a) Source/target streams $\mathbf{\Lambda}$ and $\mathbf{\Lambda}^*$ merge at the classifier level.
(b) Loss $\hbar$ aligns covariances on the manifold of $\mathcal{S}_{++}$ matrices.
(c) At the test time, we use the target stream and the trained classifier.

- For alignment of covariances, the Euclidean distance is suboptimal in the light of Riemannian geometry.

  [Koniusz *et al.*, Museum Exhibit Identification Challenge for the Supervised Domain Adaptation and Beyond (ECCV, 2018)].

Domain Adaptation: Rimannian distances

- For alignment of covariances/SPD matrices, the Euclidean distance is suboptimal in the light of Riemannian geometry.

Dist./Ref.	$d^2(\boldsymbol{\Sigma}, \boldsymbol{\Sigma}^*)$	Invar.	Tr. Ineq.	Geo.	d if $\mathcal{S}_+$	$\nabla_{\boldsymbol{\Sigma}}$ if $\mathcal{S}_+$	$\frac{\partial d^2(\boldsymbol{\Sigma}, \boldsymbol{\Sigma}^*)}{\partial \boldsymbol{\Sigma}}$
Frobenius	$\|\|\boldsymbol{\Sigma} - \boldsymbol{\Sigma}^*\|\|_F^2$	rot.	yes	no	fin.	fin.	$2(\boldsymbol{\Sigma} - \boldsymbol{\Sigma}^*)$
AIRM	$\|\| \log(\boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\Sigma}^* \boldsymbol{\Sigma}^{-\frac{1}{2}}) \|\|_F^2$	aff./inv.	yes	yes	∞	∞	$-2\boldsymbol{\Sigma}^{-\frac{1}{2}} \log(\boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\Sigma}^* \boldsymbol{\Sigma}^{-\frac{1}{2}}) \boldsymbol{\Sigma}^{-\frac{1}{2}}$
JBLD	$\log \left\lvert \frac{\boldsymbol{\Sigma} + \boldsymbol{\Sigma}^*}{2} \right\rvert - \frac{1}{2} \log \lvert \boldsymbol{\Sigma} \boldsymbol{\Sigma}^* \rvert$	aff./inv.	no	no	∞	∞	$(\boldsymbol{\Sigma} + \boldsymbol{\Sigma}^*)^{-1} - \frac{1}{2} \boldsymbol{\Sigma}^{-1}$

We use Affine Inv. Riemannian Metric (AIRM) and Jensen-Bregman LogDet Divergence (JBLD).

Domain Adaptation: Rimannian distances

- For GPU/CPU, SVD of large matrices ($d \geq 2048$) in CUDA BLAS is extremely slow.
- Idea: we exploit the low-rank nature of our covariance matrices + low number of datapoints (RKHS-friendly setting).
- For typical $N \approx 30$, $N^* \approx 3$, we get 33×33 dim. covariances rather than 4096×4096.

Domain Adaptation: Rimannian distances

- For GPU/CPU, SVD of large matrices ($d \geq 2048$) in CUDA BLAS is extremely slow.
- Idea: we exploit the low-rank nature of our covariance matrices + low number of datapoints (RKHS-friendly setting).
- For typical $N \approx 30$, $N^* \approx 3$, we get 33×33 dim. covariances rather than 4096×4096.
- For each class $c \in \mathcal{I}_C$, we choose $\mathbf{X} = \mathbf{Z} = [\mathbf{\Phi}_c, \mathbf{\Phi}_c^*]$.
- From the Nyström projection, we obtain: $\mathbf{\Pi}(\mathbf{X}) = (\mathbf{Z}^T\mathbf{Z})^{-0.5}\mathbf{Z}^T\mathbf{X} = \cancel{\mathbf{Z}}\mathbf{X} = (\mathbf{Z}^T\mathbf{Z})^{0.5} = (\mathbf{X}^T\mathbf{X})^{0.5}$.
- Then $\mathbf{\Pi}(\mathbf{\Phi}) = [\mathbf{y}_1, ..., \mathbf{y}_N]$ and $\mathbf{\Pi}(\mathbf{\Phi}^*) = [\mathbf{y}_{N+1}, ..., \mathbf{y}_{N+N*}]$.

Domain Adaptation: Rimannian distances

- For GPU/CPU, SVD of large matrices ($d \geq 2048$) in CUDA BLAS is extremely slow.
- Idea: we exploit the low-rank nature of our covariance matrices + low number of datapoints (RKHS-friendly setting).
- For typical $N \approx 30$, $N^* \approx 3$, we get 33×33 dim. covariances rather than 4096×4096.
- For each class $c \in \mathcal{I}_C$, we choose $\mathbf{X} = \mathbf{Z} = [\mathbf{\Phi}_c, \mathbf{\Phi}_c^*]$.
- From the Nyström projection, we obtain: $\mathbf{\Pi}(\mathbf{X}) = (\mathbf{Z}^T\mathbf{Z})^{-0.5}\mathbf{Z}^T\mathbf{X} = \mathbf{Z}\mathbf{X} = (\mathbf{Z}^T\mathbf{Z})^{0.5} = (\mathbf{X}^T\mathbf{X})^{0.5}$.
- Then $\mathbf{\Pi}(\mathbf{\Phi}) = [\mathbf{y}_1, ..., \mathbf{y}_N]$ and $\mathbf{\Pi}(\mathbf{\Phi}^*) = [\mathbf{y}_{N+1}, ..., \mathbf{y}_{N+N*}]$.
- $\mathbf{\Pi}(\mathbf{X})$ is isometric w.r.t. AIRM/JBLD, that is $d_g^2(\mathbf{\Sigma}(\mathbf{\Phi}), \mathbf{\Sigma}(\mathbf{\Phi}^*)) = d_g^2(\mathbf{\Sigma}(\mathbf{\Pi}(\mathbf{\Phi})), \mathbf{\Sigma}(\mathbf{\Pi}(\mathbf{\Phi}^*)))$ (*)

Domain Adaptation: Rimannian distances

- For GPU/CPU, SVD of large matrices ($d \geq 2048$) in CUDA BLAS is extremely slow.
- Idea: we exploit the low-rank nature of our covariance matrices + low number of datapoints (RKHS-friendly setting).
- For typical $N \approx 30$, $N^* \approx 3$, we get 33×33 dim. covariances rather than 4096×4096.
- For each class $c \in \mathcal{I}_C$, we choose $\mathbf{X} = \mathbf{Z} = [\mathbf{\Phi}_c, \mathbf{\Phi}_c^*]$.
- From the Nyström projection, we obtain: $\mathbf{\Pi}(\mathbf{X}) = (\mathbf{Z}^T\mathbf{Z})^{-0.5}\mathbf{Z}^T\mathbf{X} = \cancel{\mathbf{Z}}\mathbf{X} = (\mathbf{Z}^T\mathbf{Z})^{0.5} = (\mathbf{X}^T\mathbf{X})^{0.5}$.
- Then $\mathbf{\Pi}(\mathbf{\Phi}) = [\mathbf{y}_1, ..., \mathbf{y}_N]$ and $\mathbf{\Pi}(\mathbf{\Phi}^*) = [\mathbf{y}_{N+1}, ..., \mathbf{y}_{N+N*}]$.
- $\mathbf{\Pi}(\mathbf{X})$ is isometric w.r.t. AIRM/JBLD, that is $d_g^2(\mathbf{\Sigma}(\mathbf{\Phi}), \mathbf{\Sigma}(\mathbf{\Phi}^*)) = d_g^2(\mathbf{\Sigma}(\mathbf{\Pi}(\mathbf{\Phi})), \mathbf{\Sigma}(\mathbf{\Pi}(\mathbf{\Phi}^*)))$ (*)
- $\cancel{\mathbf{Z}}(\mathbf{X})$ can be treated as a constant in differentiation $\frac{\partial \mathbf{\Pi}(\mathbf{X})}{\partial X_{mn}} = \frac{\partial \cancel{\mathbf{Z}}(\mathbf{X})\mathbf{X}}{\partial X_{mn}} = \cancel{\mathbf{Z}}(\mathbf{X})\frac{\partial \mathbf{X}}{\partial X_{mn}} = \cancel{\mathbf{Z}}(\mathbf{X})\mathbf{J}_{mn}$ (*)
- $\cancel{\mathbf{Z}}$ is a composite rotation (*) and the Euclidean, JBLD and AIRM distances are rotation-invariant (*), hence isometry (*)

Domain Adaptation: Museum Exhibit Identification Challenge (New Dataset)

- In light of the Office detaset reaching ∼90%, we have proposed a new challenging dataset
- Open MIC has 866 different types of museum exhibits
- ∼7600 source images (mobile phone, well aligned)
- ∼7600 target images (wearable cameras, in-the-wild, various photometric and geometric distortions: blur, motion, zoom, glare, clipping, clutter, rotations, *etc*)

	So	*JBLD*	*AIRM*
sp1	55.8	**57.7**	57.2
sp2	58.9	**58.9**	58.9
sp3	69.6	**71.4**	71.4
sp4	53.8	**57.7**	57.7
sp5	58.3	**60.4**	60.4
acc.	59.3	**61.2**	61.1

AIRM vs. JBLD.

Domain Adaptation: Museum Exhibit Identification Challenge (New Dataset)

Source:

Target:

http://users.cecs.anu.edu.au/~koniusz/openmic-dataset/

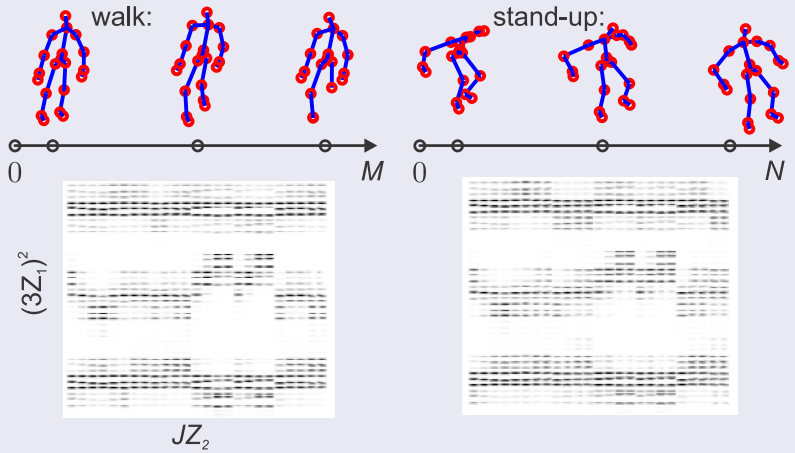
We challenge you to obtain 60% on 1-shot 20-way few-shot learning, 75% on 1-shot 5-way few-shot learning, or 50% in unsupervised domain adaptation :) Can your algorithm generalize without memorizing millions of images from ImageNet? After all, few-shot learning is about learning from few examples not from 2M images...

Domain Adaptation: Action Recognition

- Let Π_A and Π_B be two sequences, each with J joints, and M and N frames, respectively
- Let $\mathbf{x}_{is} \in \mathbb{R}^3$ and $\mathbf{y}_{jt} \in \mathbb{R}^3$ correspond to coordinates of joints
- We define an SCK kernel between sequences Π_A and Π_B as:

$$K(\Pi_A, \Pi_B) = \frac{1}{MN} \sum_{i \in \mathcal{I}_J} \sum_{s \in \mathcal{I}_M} \sum_{t \in \mathcal{I}_N} K_{\sigma_1}(\mathbf{x}_{is} - \mathbf{y}_{it})^2 \, G_{\sigma_2}\left(\frac{s}{M} - \frac{t}{N}\right), \tag{26}$$

- Linearization of this kernel results in the following feature maps:

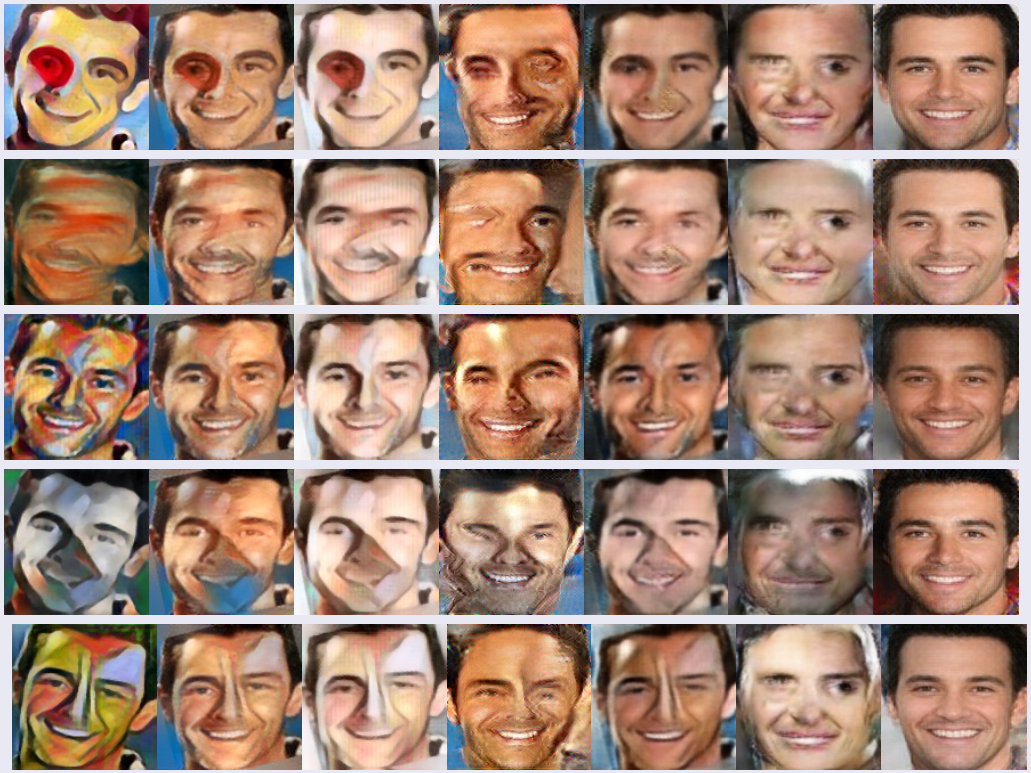


- On NTU→SBUKinect, we get 91.13→94.36% increase
- On NTU→UTK, we get 96.5%→98.9%
  [Tas, Koniusz, CNN-based Action Recognition and Supervised Domain Adaptation on 3D Body Skeletons via Kernel Feature Maps (BMVC, 2018)]

Face Recovery from Portraits

- Gatys used covariance alignment to transfer the style to contents images. [Gatys *et al.*, Image Style Transfer Using Convolutional Neural Networks (CVPR, 2016)]
- We use the Log-Euclidean distance to cluster most distinct styles for training a destylization pipeline. [Shiri, Yu, Porikli, Hartley, Koniusz, Identity-preserving Face Recovery from Stylized Portraits (IJCV, 2019)]

RF

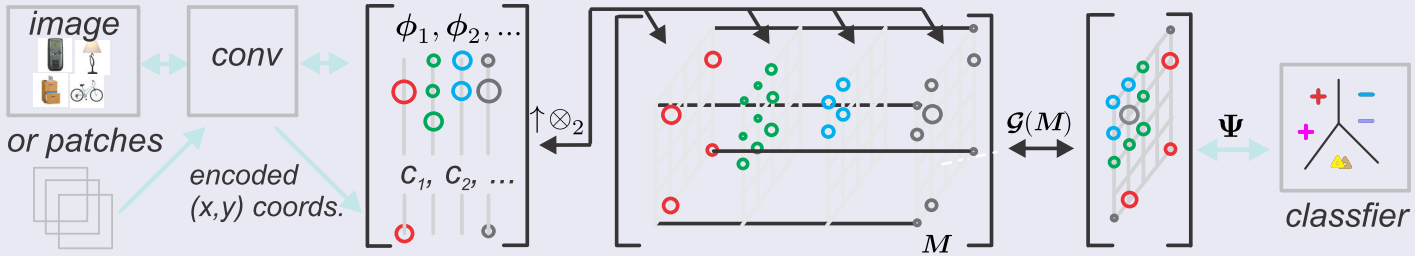


SF A B C D E Ours

- RF: Ground-truth face, SF: Stylised face, Ours: Recovered face
- A: Gatys, B: Johnson, C: Li and Wand's (MGAN), D: Isola (pix2pix), E: Zhu (CycleGAN)

A Deeper Look at Power Normalisations

- We look at power normalisations in end-to-end setting



- We derive element-wise and spectral pooling operators

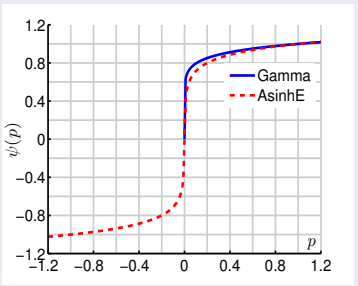
Pooling function	$\psi(p)$ if $p<0$	$\psi'(p)$ if $p=0$	$\psi(p)$	$\psi'(p)$
Gamma	inv.	∞	p^γ	$\gamma p^{\gamma-1}$
MaxExp	inv.	fin.	$1-(1-p)^\eta$	$\eta(1-p)^{\eta-1}$
AsinhE	ok	fin.	$\text{Asinh}(\gamma' p)$	$\frac{\gamma'}{\sqrt{1+\gamma'^2 p^2}}$
SigmE	ok	fin.	$\frac{2}{1+e^{-\eta' p}}-1$	$\frac{2\eta' e^{-\eta' p}}{(1+e^{-\eta' p})^2}$

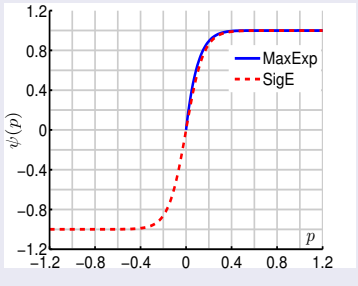
	Gamma	*MaxExp*	*AsinhE*	*SigmE*
$\mathcal{G}(\boldsymbol{M})$	$\boldsymbol{M}^\gamma$	$\mathbb{I}-(\mathbb{I}-\frac{\boldsymbol{M}}{\text{Tr}(\boldsymbol{M})+\lambda})^\eta$	$\log\left(\gamma'\boldsymbol{M}+(\mathbb{I}+\gamma'^2\boldsymbol{M}^2)^{\frac{1}{2}}\right)$	$2\left(\mathbb{I}+e^{\frac{-\eta'\boldsymbol{M}}{\text{Tr}(\boldsymbol{M})+\lambda}}\right)^{-1}-\mathbb{I}$
der.	Closed/SVD	Closed/SVD	SVD	SVD

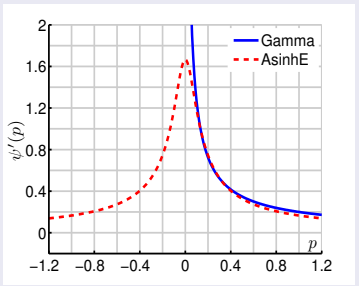
[Koniusz *et al.*, A Deeper Look at Power Normalisations (CVPR, 2018)]

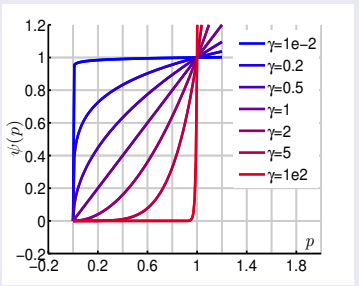
Deeper Look at Power Normalisations

- We analyse power norms which...turn out to be sigmoids (neural nets)









- We show that the probability of at least one co-occurrence event $(\phi_n \cap \phi'_n = 1)$ in ϕ_n and ϕ'_n simultaneously in N trials is $1-(1-p)^N$
- Our proof uses a Multinomial distribution model with four events for (ϕ_n) and (ϕ'_n) and events $(\phi_n \cap \phi'_n = 1)$, $(\phi_n = 1, \phi'_n = 0)$, $(\phi_n = 0, \phi'_n = 1)$ and $(\phi_n \cup \phi'_n = 0)$. The probability of at least one co-occurrence $(\phi_n \cap \phi'_n = 1)$ in N trials becomes:

$$\sum_{n=1}^{N} \sum_{n'=0}^{N-n} \sum_{n''=0}^{N-n-n'} \binom{N}{n, n', n'', N-n-n'-n''} p^n q^{n'} s^{n''} (1-p-q-s)^{N-n-n'-n''} = 1-(1-p)^N$$

- Many papers, *e.g.* RootSIFT, Fisher Vectors, bi-linear pooling, CNNs use the square root normalisation but do not seem to understand its role that well. Take the bin 'clipping'in SIFT: that is a Power Normalisation (AxMin) too.

Deeper Look at Power Norm. (Fast Spectral MaxExp)

- Backprop through Spectral MaxExp given as $\mathbb{I}-(\mathbb{I}-\boldsymbol{M})^{\eta}$ is very fast (no need to backpropagate through SVD).
- It can be achieved faster than Newton-Schulz iterations for Gamma $\boldsymbol{M}^{0.5}$ while it enjoys a tunable parameter η.
- Without the loss of generality, assume $\boldsymbol{M}$ is trace-normalized. Then, the derivative $\frac{\partial \boldsymbol{\mathcal{G}}(\boldsymbol{M})}{\partial\ M_{kl}}$ of spec. MaxExp has the closed form:

$$\frac{\partial \boldsymbol{\mathcal{G}}(\boldsymbol{M})}{\partial\ M_{kl}} = -\sum_{n=0}^{\eta-1} (\mathbb{I}-\boldsymbol{M})^{n} \mathbf{J}_{kl} (\mathbb{I}-\boldsymbol{M})^{\eta-1-n}. \tag{27}$$

- Now, let $\ell(\boldsymbol{\Psi}, \mathbf{W})$ be some classification loss (or any layer with param. $\mathbf{W}$) where $\boldsymbol{\Psi} \in \mathcal{S}_{+}^{d}$ (or $\mathcal{S}_{++}$) are our feature maps, that is $\boldsymbol{\Psi} = \boldsymbol{\mathcal{G}}(\boldsymbol{M})$. Then, we obtain a more versatile equation:

$$\sum_{k,l} \frac{\partial \ell(\boldsymbol{\Psi}, \mathbf{W})}{\partial \boldsymbol{\Psi}_{kl}} \frac{\partial \boldsymbol{\Psi}_{kl}}{\partial \boldsymbol{M}} = -\sum_{n=0}^{\eta-1} (\mathbb{I}-\boldsymbol{M})^{n} \frac{\partial \ell(\boldsymbol{\Psi}, \mathbf{W})}{\partial \boldsymbol{\Psi}} (\mathbb{I}-\boldsymbol{M})^{\eta-1-n}. \tag{28}$$

- Finally, the trace-norm. can be reversed: $\left(\mathbb{I}-(\mathbb{I}-\frac{\boldsymbol{M}}{\mathrm{Tr}(\boldsymbol{M})+\lambda})^{\eta}\right) \cdot (\mathrm{Tr}(\boldsymbol{M})+\lambda)^{\gamma}$ where $\gamma = 0.5$. Why? Mind the energy (norms) of pre-trained model.

Deeper Look at Power Norm. (Fast Spectral MaxExp)

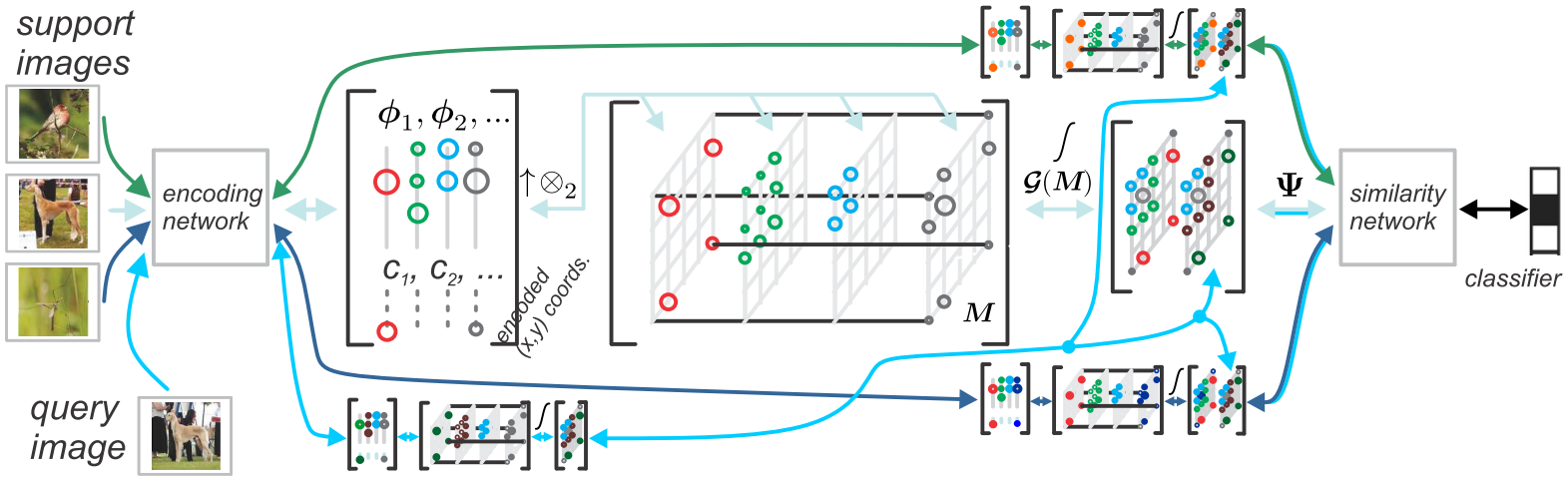
- Computing $(\mathbb{I}-\boldsymbol{M})^1, ..., (\mathbb{I}-\boldsymbol{M})^{\eta-1}$ for $\eta \in \mathbb{N}^+$ has a lightweight complexity $\mathcal{O}(m\log(\eta-1))$, where m is the cost of a matrix-matrix multiplication, and $\log(\eta-1)$ is the number of multiplications in exponentiation by squaring.
- The final complexity is $\mathcal{O}(m\eta)$ which scales linearly with η.
- In comparison, an approx. der. of the matrix square root via Newton-Schulz iterations has $\mathcal{O}(mk)$ complexity, where $k \approx 20$ is set by authors of [Lin, Maji, Improved Bilinear Pooling with CNNs (BMVC, 2017)]
- In head-to-head comparisons, we require $\log(\eta-1)+2\eta/2 \approx 56.6$ matrix-matrix multiplications for $\eta=50$ (typically $20 \leq \eta \leq 80$). Newton-Schulz iter. require $4k=80$ matrix-matrix mult. for $k=20$.
- Memory-wise, MaxExp and Newton-Schulz iter. need to store the chain of $\eta-1$ and $2k$ matrices, resp.
- In contrast, the cost of SVD is $\mathcal{O}(d^\omega)$, where $2<\omega<2.376$, and the implementation of SVD in CUDA BLAS is suboptimal.

Deeper Look at Power Normalisations

Results

- Flower102: 97.28% (spectral MaxExp), 96.78% (element-wise MaxExp), 95.74% (element-wise Gamma), 94.70% (bilinear), other methods $\sim$95.3%
- FMD: 85.5% (element-wise MaxExp) vs. 82.3% (other methods)
- MIT67: 86.3% (element-wise MaxExp) vs. 84.3% (Fourier Features)
- Food101: 87.8% (spectral MaxExp) vs. 85.5% (CNN kernel pooling)
- In conclusion, Fast Spectral MaxExp [Koniusz *et al.*, A Deeper Look at Power Normalisations (CVPR, 2018)] is even faster than the matrix square root via the Newton-Schulz iterations in [Lin, Maji, Improved Bilinear Pooling with CNNs (BMVC, 2017)].
- A new work on gamma-democratic higher-order pooling – similar to matrix spectral operators, a bit lower accuracy but even faster [Lin, Maji, Koniusz, Second-order Democratic Aggregation (ECCV, 2018)]

Few-shot Learning



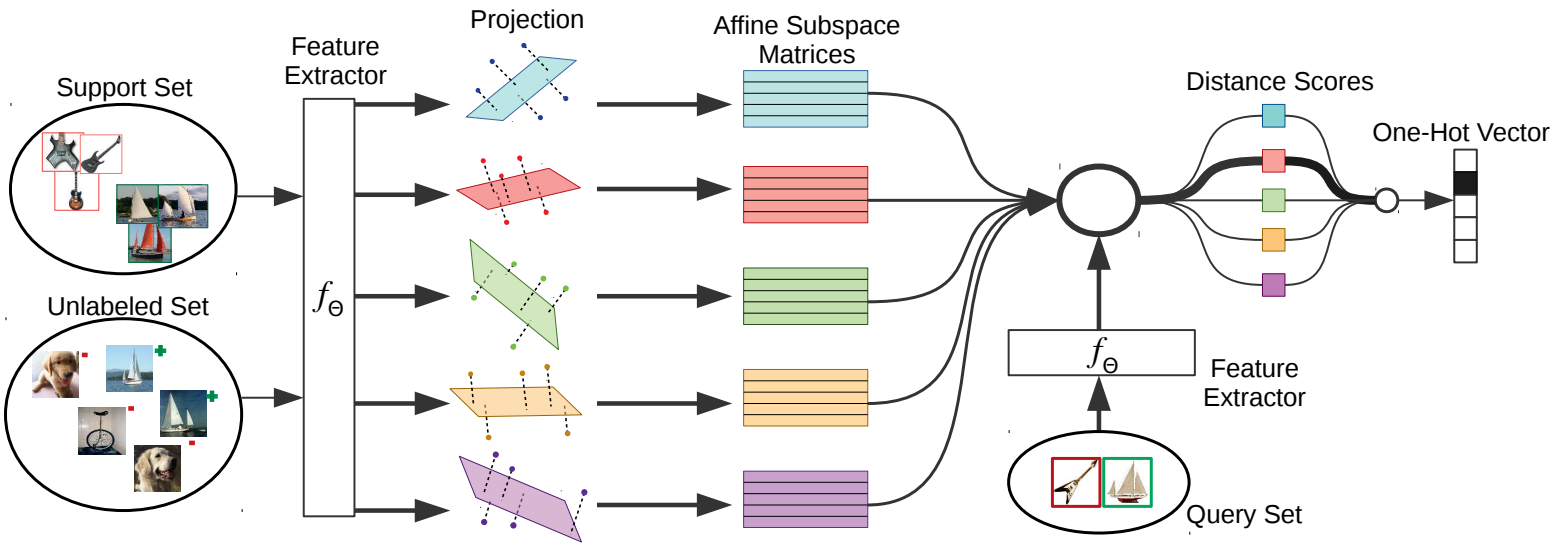
- Second-order pooling works really nicely in relation learning (decreasing burst for pairs/groups is even more important).

  [Zhang, Koniusz, Power Normalizing Second-order Similarity Network for Few-shot Learnings (WACV, 2019)]

  [Zhang, Zhang, Koniusz, Few-Shot Learning via Saliency-guided Hallucination of Samples (CVPR, 2019)]

Open MIC dataset	L	$p1 \rightarrow p2$	$p1 \rightarrow p3$	$p1 \rightarrow p4$	$p2 \rightarrow p1$	$p2 \rightarrow p3$	$p2 \rightarrow p4$
Relation Net	20	40.1±0.5	30.4±0.5	41.4±0.5	23.5±0.4	26.4±0.5	38.6±0.5
SoSN		61.0±0.5	42.3±0.5	60.2±0.5	35.7±0.5	37.0±0.5	54.8±0.5
SoSN+SigmE		**61.5**±0.6	**42.5**±0.5	**61.0**±0.5	**36.1**±0.5	**38.3**±0.5	**56.3**±0.5
SoSN+SigmE+224x224		**63.6**±0.5	**48.7**±0.6	**65.6**±0.5	**42.6**±0.5	**43.9**±0.5	**61.8**±0.5

Few-shot Learning



- Subspaces are also second-order representations.

  [Simon, Koniusz, Harandi, Projective Subspace Networks For Few-Shot Learning (2018)]

Open MIC dataset	**5-way 1-shot**				
	$p1 \rightarrow p2$	$p2 \rightarrow p3$	$p3 \rightarrow p4$	$p4 \rightarrow p1$	Avg
Matching Nets	$69.40 \pm 0.9\%$	$57.30 \pm 1.0\%$	$76.35 \pm 1.0\%$	$53.68 \pm 0.9\%$	64.18
PN	$66.33 \pm 0.9\%$	$52.03 \pm 1.1\%$	$74.28 \pm 0.9\%$	$54.30 \pm 0.9\%$	61.74
SoSN	$\mathbf{78.00 \pm 0.9\%}$	$\mathbf{60.10 \pm 1.1\%}$	$75.50 \pm 1.0\%$	$57.80 \pm 1.1\%$	67.85
PSN	$72.92 \pm 0.9\%$	$57.60 \pm 0.9\%$	$\mathbf{77.59 \pm 0.9\%}$	$61.29 \pm 1.1\%$	67.35
PSN$^+$	$74.24 \pm 0.9\%$	$59.20 \pm 0.9\%$	$\mathbf{78.25 \pm 0.9\%}$	$\mathbf{61.48 \pm 0.9\%}$	**68.29**

Thank You