Tensor Representations for Action Recognition

Piotr Koniusz Lei Wang Anoop Cherian

Abstract—Human actions in video sequences are characterized by the complex interplay between spatial features and their temporal dynamics. In this paper, we propose novel tensor representations for compactly capturing such higher-order relationships between visual features for the task of action recognition. We propose two tensor-based feature representations, viz. (i) *sequence compatibility kernel* (SCK) and (ii) *dynamics compatibility kernel* (DCK). SCK builds on the spatio-temporal correlations between features, whereas DCK explicitly models the action dynamics of a sequence. We also explore generalization of SCK, coined SCK$\oplus$, that operates on subsequences to capture the local-global interplay of correlations, which can incorporate multi-modal inputs *e.g.*, skeleton 3D body-joints and per-frame classifier scores obtained from deep learning models trained on videos. We introduce linearization of these kernels that lead to compact and fast descriptors. We provide experiments on (i) 3D skeleton action sequences, (ii) fine-grained video sequences, and (iii) standard non-fine-grained videos. As our final representations are tensors that capture higher-order relationships of features, they relate to co-occurrences for robust fine-grained recognition [1, 2]. We use higher-order tensors and so-called Eigenvalue Power Normalization (EPN) which have been long speculated to perform spectral detection of higher-order occurrences [3, 4], thus detecting fine-grained relationships of features rather than merely count features in action sequences. We prove that a tensor of order r, built from Z_* dimensional features, coupled with EPN indeed detects if at least one higher-order occurrence is 'projected' into one of its $\binom{Z_*}{r}$ subspaces of dim. r represented by the tensor, thus forming a Tensor Power Normalization metric endowed with $\binom{Z_*}{r}$ such 'detectors'.

Index Terms—CNN, 3D Skeletons, Action Recognition, Aggregation, Kernels, Higher-order Tensors, HOSVD, Power Normalization.

1 Introduction

Human action recognition is a central problem in computer vision with potential impact in surveillance, human-robot interaction, elderly assistance systems, *etc*. While there have been significant advancements in this area over the past few years, action recognition in unconstrained settings still remains a challenge. Some papers simplify the problem from using RGB cameras to the use of Microsoft Kinect or the OpenPose library [5] to localize human body-parts, produce moving 3D skeletons [6] and use them for recognition. However, skeletons can be noisy due to badly localized body-parts, self-occlusions, and sensor errors. Similarly, a popular strategy of classifying RGB frames into actions followed by average/max-pooling fails as only correlations of some features are informative [7, 8, 9]. Such observations motivate the need for higher-order reasoning on 3D skeletons/frame-wise CNN classifier scores taking action recognition toward fine-grained modeling.

Recent approaches which work with skeletons can be mainly divided into two perspectives, namely (i) generative models that assume the skeleton points are produced by a latent dynamic model [10] corrupted by noise and (ii) discriminative approaches that generate compact representations of sequences on which classifiers are trained [11]. Due to the huge configuration space of 3D actions and the unavailability of sufficient training data, discriminative approaches have been more successful. In this line of research, the main idea is to compactly represent the spatio-temporal evolution of 3D skeletons, and later train classifiers on these representations to recognize actions. Fortunately, there is a definitive structure to motions of 3D joints relative to each other due to the connectivity and length constraints of body-parts. Such constraints have been used with the Lie Algebra [12], positive definite matrices [13, 14], torus manifold [15], Hanklet representations [16], *etc*. While modeling actions with explicit manifold assumptions is useful, it is computationally costly.

However, action recognition from videos [17, 18, 19, 20] does not require elaborate skeletal models. A two-stream CNN framework [17] uses two streams to model RGB frames and optical flow. Tran *et al*. [18] use CNNs to learn spatio-temporal filters. Karpathy *et al*. [19] apply RGB and optical-flow fusion, whereas approach [20] combines CNNs with LSTM to model temporal flow. Wang *et al*. [21] apply a long-range temporal structure modeling. Tran *et al*. [22] study several forms of spatiotemporal convolutions. Recent works on fine-grained activity recognition use CNNs [23, 24] and the human pose estimation for high-level fine-grained reasoning [23, 25, 26, 27]. Finally, the recent I3D model [28] 'inflates' 2D CNN filters pretrained on ImageNet to spatio-temporal 3D filters yielding state-of-the-art results.

In contrast to these approaches, we present a novel representation of actions based on 3D skeleton sequences and the CNN classifier score sequences. We avoid assumptions about the data manifold by capturing higher-order statistics of the body-joints and the classifier score interactions per sequence. To this end, our scheme combines positive definite kernels and higher-order tensors, with the goal of obtaining rich and compact representations that benefit from the non-linearity of radial basis functions (RBF). Such a scheme captures higher-order data statistics [4], complex action dynamics [29, 30] and fine-grained relations [1, 2].

We present two representations for classification of 3D skeletons. Our first representation, *sequence compatibility kernel* (SCK), captures the spatio-temporal compatibility of body-joints between two sequences. To this end, we present an RBF kernel

- *P. Koniusz and L. Wang are with Data61/CSIRO (former NICTA) and the Australian National University, Canberra, Australia, ACT2601. E-mail: see http://claret.wikidot.com*
- *A. Cherian is with Mitsubishi Electric Research Labs (MERL), Cambridge, MA, USA.*

Manuscript submitted Dec-2018. Manuscript accepted by TPAMI on 24-Dec-2019.

formulation that jointly captures the spatial and temporal similarity of each body-pose (normalized with respect to the hip position) in a sequence against those in another. We show that tensors generated from third-order outer-products of the linearizations of these kernels are a simple yet powerful representation capturing higher-order statistics of body-parts.

Our second representation, termed *dynamics compatibility kernel* (DCK), represents spatio-temporal dynamics of each sequence explicitly. We present a novel RBF kernel formulation that captures the similarity between a pair of body-poses in a given sequence explicitly, and then compare it against such body-pose pairs in other sequences. Such spatio-temporal modeling could be expensive due to the volumetric nature of space and time. However, we show that using an appropriate kernel model can shrink the time-related variable into a small representation of constant size after kernel linearization. With this approach, we can model both spatial and temporal variations in the form of co-occurrences which could otherwise be prohibitive. We show empirically that SCK and DCK are complementary.

As SCK/DCK work on entire sequences, we formulate an SCK-like kernel over multiple length subsequences as some of subsequences capture the gist of performed actions better than full sequences. To show the versatility of the extended SCK, we apply it to capture spatio-temporal compatibility of frame-wise CNN classifier scores from videos (regular and fine-grained actions).

We present experiments on seven standard datasets, namely (i) UTKinect-Actions [31], (ii) Florence3D-Actions [32], (iii) MSR-Action3D [33] and (iv) HMDB-51[34] datasets as well as two fine-grained datasets (v) NTU RGB+D [35], (vi) MPII Cooking Activities [25] and (vii) Kinetics [36]. We use the first three datasets as a source of 3D body joint sequences (as well as Kinetics), NTU for both 3D body joint sequences, and videos with RGB frames and optical flow frames, and HMDB-51 and MPII Cooking Activities for videos with RGB and optical flow frames. We show that our extensions can still achieve state-of-the-art accuracy two years after SCK/DCK were proposed [29]. To summarize:

i. We design sequence and dynamics compatibility kernels that capture spatio-temporal evolution of 3D skeleton body-joints.
ii. We derive linearizations of these kernels by tensors.
iii. We extend these kernels to aggregation over multiple subsequences and CNN classifier scores.
iv. We conduct a novel theoretical analysis of Tensor Power Normalization which connects it to subspace methods. We are the first to conduct a theoretical analysis of higher-order pooling with Tensor Power Normalization in Section D, and use it for generic/fine-grained action recognition.

2 Related Work

In the first part of our paper, we focus on action recognition from an articulated set of connected body-joints that evolve in time [37]. A temporal evolution of the human skeleton is very informative for action recognition as shown by Johansson in his seminal experiment involving the moving lights display [38]. At the simplest level, the human body can be represented as a set of 3D points corresponding to body-joints such as elbow, wrist, knee, ankle, *etc*. Action dynamics has been modeled using the motion of such 3D points in [14, 39], using joint orientations with respect to a reference axis [40] and even relative body-joint positions [41, 42]. In contrast, we represent these 3D body-joints by kernels whose linearization results in higher-order tensors capturing complex statistics. We also note parts-based approaches that use connected body segments [12, 43, 44, 45]. For details, see a survey [11].

We also handle the temporal domain differently to other methods. 3D joint locations are modeled as temporal hierarchy of coefficients in [14]. Pairwise relative positions of joints were modeled in [41] and combined with a hierarchy of Fourier coefficients to capture temporal evolution of actions. In [42], the relative joint positions and their temporal displacements are modeled with respect to the initial frame. In [12], the displacements and angles between the body parts are represented as a collection of matrices belonging to SE(3), a special Euclidean group. The temporal domain is handled by the dynamic time warping and Fourier temporal pyramid matching. In contrast, we avoid expensive time warping by modeling the temporal domain with an RBF kernel invariant to local temporal shifts.

Our scheme also differs from works such as kernel descriptors [46] that sum gradient orientations over image patches, action recognition via kernelized covariances [47, 48, 49], and a time series kernel [50] which extracts spatio-temporal autocorrelations. In contrast, our scheme sums over several multiplicative and additive RBF kernels. We capture higher-order statistics by linearizing a polynomial kernel and avoid evaluating costly kernels directly.

Third-order tensors have been used to form spatio-temporal tensors on videos in [51]. Non-negative tensor factorization is used for image denoising [52], tensors are used for texture rendering [53] and for face recognition [54]. A survey of multi-linear algebraic methods for tensor subspace learning is available in [55]. These methods use a single tensor, whereas we use tensors as descriptors [3, 4, 56, 57]. However, we use third-order tensors for action recognition, which poses a set of new challenges.

For fine-grained action recognition, high-level sophisticated action reasoning [23, 25, 26, 27] is typically used together with pose estimation systems [58, 59]. However, these approaches scale poorly to millions of video frames. Human-object interactions in the videos are analyzed in [60]. Correlations between space-time features are proposed in [61].

Power Normalization approaches [2, 3, 4, 56, 62] speculate that Eigenvalue Power Normalization prevents so-called burstiness, thus performing spectral detection of higher-order occurrences of features [3, 4], which can be paraphrased as '*do a knife, a hand and a chopping board co-occur together?*' rather than '*how many knifes, hands and chopping boards appear in the scene?*'

Moreover, first-order pooling was successfully used for representing action recognition via hallucination [63]. Papers [2, 62] study second-order pooling, power normalizing functions and their taxonomy while fast pooling methods are proposed in [1, 62, 64].

Finally, second-order pooling was successfully used for few-shot action recognition [65], few-shot classification [66, 67], few-shot segmentation [68], modulating optimization [69], style transfer [70, 71, 72, 73] and action self-supervision [74]. Noteworthy are also graph convolutional networks [75, 76, 77] and embeddings [78] easily applicable to 3D skeleton action recognition.

3 Preliminaries

In this section, we review our notations and the necessary background on shift-invariant kernels and their linearizations.

3.1 Tensor Notations

Figure 1a illustrates the notion of tensors, their order and modes. Let $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ denote a third-order tensor. Using the Matlab

notation, we refer to the k-th slice of this tensor as $\boldsymbol{\mathcal{V}}_{:,:,k}$, which is a $d_1 \times d_2$ matrix. For a matrix $\boldsymbol{V} \in \mathbb{R}^{d_1 \times d_2}$ and a vector $\mathbf{v} \in \mathbb{R}^{d_3}$, the notation $\boldsymbol{\mathcal{V}} = \boldsymbol{V} \uparrow\otimes \mathbf{v}$ produces a tensor $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ whose k-th slice is given by $\boldsymbol{V} \cdot v_k$, v_k being the k-th coefficient of $\mathbf{v}$. Figure 1b illustrates such an outer-product. Symmetric third-order tensors of rank one are formed by the outer-product of a vector $\mathbf{v} \in \mathbb{R}^d$ in three modes, that is, a rank-one $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{d \times d \times d}$ is obtained from $\mathbf{v}$ as $\boldsymbol{\mathcal{V}} = (\uparrow\otimes_3 \mathbf{v} \triangleq (\mathbf{v}\mathbf{v}^T) \uparrow\otimes \mathbf{v})$ which yields $\mathcal{V}_{ijk} = v_i \cdot v_j \cdot v_k$, where $\mathcal{V}_{ijk}$ represents the ijk-th element of $\boldsymbol{\mathcal{V}}$. Matrices have two modes: the first and second mode correspond to the row and column indexes i and j, respectively. Order r tensors have r modes addressed by $\mathcal{V}_{i_1 \ldots i_r}$ where $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{d_1 \times \ldots \times d_k \times \ldots \times d_r}$ and k indicates the mode k. Concatenation of n tensors in mode k is simply stacking them along mode k, denoted as $[\boldsymbol{\mathcal{V}}_i]_{i \in \mathcal{I}_n}^{\oplus_k} \equiv \text{numpy.concatenate}((\boldsymbol{\mathcal{V}}_1, ..., \boldsymbol{\mathcal{V}}_n), \text{axis} = k-1))$. $\mathcal{I}_n$ is an index sequence $1, 2, ..., n$. We define the Frobenius norm $\|\boldsymbol{\mathcal{V}}\|_F = \sqrt{\sum_{i,j,k} \mathcal{V}_{ijk}^2}$ and the inner-product between $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ as $\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{ijk} \mathcal{X}_{ijk} \mathcal{Y}_{ijk}$. Also, $\boldsymbol{e}_z$ are spanning bases of $\mathbb{R}^Z$. Further basics on tensors and tensor algebra can be found in [79].

3.2 Kernel Linearization

Let $G_\sigma(\mathbf{u} - \bar{\mathbf{u}}) = \exp(-\|\mathbf{u} - \bar{\mathbf{u}}\|_2^2 / 2\sigma^2)$ denote a standard Gaussian RBF kernel centered at $\bar{\mathbf{u}}$ and having a bandwidth σ. Kernel linearization refers to rewriting this G_σ as an inner-product of two infinite-dimensional feature maps. To obtain these maps, we use a fast approximation method based on probability product kernels [80]. Specifically, we employ the inner product of d'-dimensional isotropic Gaussians given $u, u' \in \mathbb{R}^{d'}$. Thus, we have:

$$G_\sigma(\mathbf{u} - \bar{\mathbf{u}}) = \left(\frac{2}{\pi\sigma^2}\right)^{\frac{d'}{2}} \int_{\boldsymbol{\zeta} \in \mathbb{R}^{d'}} G_{\sigma/\sqrt{2}}(\mathbf{u} - \boldsymbol{\zeta})\, G_{\sigma/\sqrt{2}}(\bar{\mathbf{u}} - \boldsymbol{\zeta})\, \mathrm{d}\boldsymbol{\zeta}. \quad (1)$$

Eq. (1) is then approximated by replacing the integral with the sum over Z pivots $\boldsymbol{\zeta}_1, ..., \boldsymbol{\zeta}_Z$. Thus, we obtain a feature map $\boldsymbol{\phi}$:

$$\boldsymbol{\phi}(\mathbf{u}; \{\boldsymbol{\zeta}_i\}_{i \in \mathcal{I}_Z}) = \left[G_{\sigma/\sqrt{2}}(\mathbf{u} - \boldsymbol{\zeta}_1), ..., G_{\sigma/\sqrt{2}}(\mathbf{u} - \boldsymbol{\zeta}_Z)\right]^T, \quad (2)$$

$$\text{and } G_\sigma(\mathbf{u} - \bar{\mathbf{u}}) \approx \left\langle \sqrt{c}\boldsymbol{\phi}(\mathbf{u}), \sqrt{c}\boldsymbol{\phi}(\bar{\mathbf{u}}) \right\rangle, \quad (3)$$

where c is a const. Eq. (3) is the linearization of the RBF kernel. Eq. (2) is the feature map. $\{\boldsymbol{\zeta}_i\}_{i \in \mathcal{I}_Z}$ are pivots. As we use 1 dim. signals, we simply cover interval $[-1; 1]$ (or $[0; 1]$) with Z equally spaced pivots. For clarity, we drop $\{\boldsymbol{\zeta}_i\}_{i \in \mathcal{I}_Z}$ and write $\boldsymbol{\phi}(\mathbf{u})$, *etc.*

3.3 Equivalence between Polynomial Kernels and the Dot-product of Tensors [4]

For any two Z' dim. feature vectors $\boldsymbol{\phi}, \bar{\boldsymbol{\phi}} \in \mathbb{R}^{Z'}$, we have:

$$\langle \boldsymbol{\phi}, \bar{\boldsymbol{\phi}} \rangle^r = \sum_{i_1=1}^{Z'} ... \sum_{i_r=1}^{Z'} \phi_{i_1} \bar{\phi}_{i_1} \cdot ... \cdot \phi_{i_r} \bar{\phi}_{i_r} = \left\langle \uparrow\otimes_r \boldsymbol{\phi}, \uparrow\otimes_r \bar{\boldsymbol{\phi}} \right\rangle, \quad (4)$$

where $\boldsymbol{\mathcal{X}} = (\uparrow\otimes_r \boldsymbol{\phi})$ is defined as $\mathcal{X}_{i_1 \ldots i_r} = \phi_{i_1} \cdot ... \cdot \phi_{i_r}$.

4 Proposed Approach

Below, we formulate the problem of action recognition from 3D skeleton sequences, which precedes an exposition of our two kernel formulations for describing actions, followed by their tensor reformulations through kernel linearization. We also introduce Eigenvalue Power Normalization and our improved kernels used for action recognition based on skeletons and/or classifier scores obtained from videos passed via CNNs.

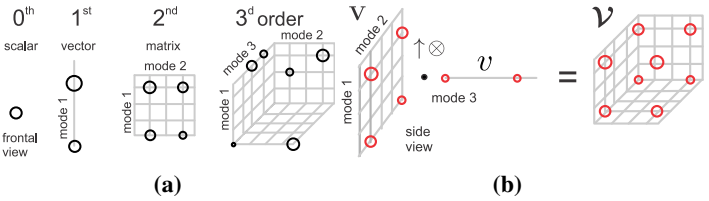


Fig. 1: Figure 1a illustrates the notion of tensors, their order and modes. Figure 1b illustrates the matrix-vector order outer-product.

4.1 Statistical Motivation

Before we outline our higher-order tensor representations, below we motivate the use of higher-order statistics. To compare skeleton sequences/videos, we want to capture distribution of local features/descriptors per sequence *e.g.*, body joints or receptive fields in CNN. The characteristic function $\varphi_\Phi(\boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{\phi} \sim \Phi}\left(\exp(i\boldsymbol{\omega}^T\boldsymbol{\phi})\right)$ describes the probability density $f_\Phi(\boldsymbol{\phi})$ of a skeleton sequence/video (local features/descriptors $\boldsymbol{\phi} \sim \Phi$).

Taylor expansion of the characteristic function per sequence is:

$$\mathbb{E}_{\boldsymbol{\phi} \sim \Phi}\left(\sum_{r=0}^{\infty} \frac{i^r}{r!} \langle \boldsymbol{\phi}, \boldsymbol{\omega} \rangle^r\right) \approx \frac{1}{N} \sum_{n=0}^{N} \sum_{r=0}^{\infty} \frac{i^r}{r!} \langle \uparrow\otimes_r \boldsymbol{\phi}_n, \uparrow\otimes_r \boldsymbol{\omega} \rangle \quad (5)$$

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

Symbol $\boldsymbol{\mathcal{X}}^{(r)} = \frac{1}{N} \sum_{n=0}^{N} \uparrow\otimes_r \boldsymbol{\phi}_n$ defines a tensor descriptor while i is the imaginary number. In principle, with infinite data and infinite moments, one can fully capture $f_\Phi(\boldsymbol{\phi})$ which is intractable. In practice, third-order moments work well in what follows while second-order moments are somewhat insufficient.

4.2 Problem Formulation

Suppose we are given a set of 3D human pose skeleton sequences, each pose consisting of J body-keypoints. Further, to simplify our notations, we assume each sequence consists of N skeletons, one per frame[1]. We define such a pose sequence Π as:

$$\Pi = \left\{\mathbf{x}_{is} \in \mathbb{R}^3, i \in \mathcal{I}_J, s \in \mathcal{I}_N\right\}. \quad (6)$$

Further, let each such a sequence Π be associated with one of K action class labels $\ell \in \mathcal{I}_K$. Our goal is to use the skeleton sequence Π and generate an action descriptor for this sequence that can be used in a classifier for recognizing the action class. In what follows, we will present two such action descriptors, namely (i) sequence compatibility kernel and (ii) dynamics compatibility kernel, which are formulated using kernel linearization and tensor algebra theories. We present both these kernel formulations next.

4.3 Sequence Compatibility Kernel

As alluded to earlier, the main idea of this kernel is to measure the compatibility between two action sequences in terms of the similarity between their skeletons and their temporal order. To this end, we assume each skeleton is centered with respect to

[1]We assume that all sequences have N frames for simplification of presentation. Our formulations are applicable to sequences of arbitrary lengths *e.g.*, M and N. Thus, we apply in practice $G_{\sigma_3}(\frac{s}{M} - \frac{t}{N})$ in Eq. (7).

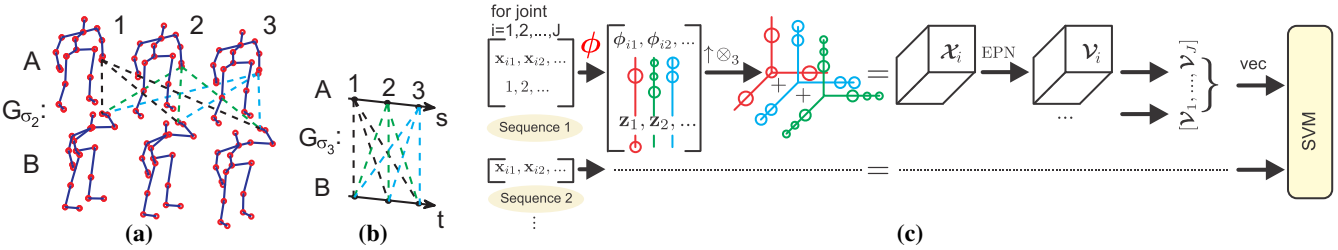


Fig. 2: Figures 2a and 2b show how SCK works – kernel G_{σ_2} compares exhaustively *e.g.* hand-related joint i for every frame in sequence A with every frame in sequence B. Kernel G_{σ_3} compares exhaustively the frame indexes. Figure 2c shows this burden is avoided by linearization – third-order statistics on feature maps $\boldsymbol{\phi}(\mathbf{x}_{is})$ and $\mathbf{z}(s/N)$ for joint i are captured in tensor $\boldsymbol{\mathcal{X}}_i$ and whitened by EPN to obtain $\boldsymbol{\mathcal{V}}_i$ which are concatenated over $i=1,...,J$ to represent a sequence. The final sequence tensors are vectorized per video by 'vec' and fed to an SVM.

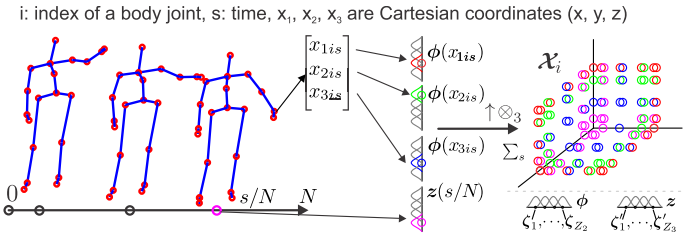


Fig. 3: Order r statistics from Eq. (7) can be understood by studying the linearization in Eq. (10). For a given joint i at time s/N (normalized frame number), we embed a 3D joint coordinate $\mathbf{x}_{is}$ (all centered w.r.t. hip) via function $\boldsymbol{\phi}(\cdot)$ into a non-linear Hilbert space representing an RBF kernel according to Eq. (2). Similarly, we embed the time s/N via function $\mathbf{z}(\cdot)$ (also by Eq. (2)). Finally, $\otimes_r$ performs the third-order outer-product on concatenated embeddings aggregated next over frames s (note $\sum_s$). The interpretation: the Gaussians 'soft-divide' the the Cartesian coordinate system along x, y, z direction, resp., and time s/N. Thus, triplets (x,y,z), $(x,y,s/N)$, $(x,z,s/N)$ and $(y,z,s/N)$ assigned into such a 'soft-divided' space capture locally three-way occurrences. They factor out one spatial (or time) variable at a time (note invariance to such a variable).

one of the body-joints (say, hip). Suppose we are given two such sequences Π_A and Π_B, each with J joints, and N frames. Further, let $\mathbf{x}_{is}\in\mathbb{R}^3$ and $\mathbf{y}_{jt}\in\mathbb{R}^3$ correspond to the body-joint coordinates of Π_A and Π_B, respectively.

We define our *sequence compatibility kernel* (SCK) between Π_A and Π_B as[1]:

$$K_S(\Pi_A,\Pi_B)=\frac{1}{\Lambda}\sum_{(i,s)\in\mathcal{J}}\sum_{(j,t)\in\mathcal{J}} G_{\sigma_1}(i-j)\Big(\beta_1 G_{\sigma_2}(\mathbf{x}_{is}-\mathbf{y}_{jt})+\beta_2\, G_{\sigma_3}(\frac{s-t}{N})\Big)^r. \tag{7}$$

Symbol Λ is a normalization constant and $\mathcal{J}=\mathcal{I}_J\times\mathcal{I}_N$. As is clear, this kernel involves three different compatibility subkernels, namely (i) G_{σ_1}, capturing the compatibility between joint-types i and j, (ii) G_{σ_2}, capturing the compatibility between joint locations $\mathbf{x}$ and $\mathbf{y}$, and (iii) G_{σ_3}, measuring the temporal alignment of two poses in two sequences. We also introduce weighting factors $\beta_1,\beta_2\geq 0$ that adjust the importance of the body-joint compatibility against the temporal alignment, where $\beta_1+\beta_2=1$. Figures 2a and 2b illustrate how this kernel works. It might come as a surprise that we use kernel G_{σ_1}. Note that our skeletons may be noisy and there is a possibility that some keypoints are detected incorrectly (for example, elbows and wrists). Thus, this kernel allows incorporating a degree of uncertainty into the alignment of such joints. To simplify our formulation, in this paper, we will assume that such errors are absent from our skeletons, and thus $G_{\sigma_1}(i-j)=\delta(i-j)$. Furthermore, standard deviations σ_2 and σ_3 control the joint-coordinate selectivity and temporal shift-invariance, respectively. That is, for $\sigma_3\to 0$, two sequences will have to match perfectly in the temporal sense. For $\sigma_3\to\infty$, the algorithm is invariant to any permutations of the frames. As will be clear in the sequel, parameter r determines the order of statistics of our kernel (we use $r=3$).

Next, we present linearization of our kernel using the method from Sections 3.2, 3.3, and Eq. (3), so that kernel $G_{\sigma_2}(\mathbf{x}-\mathbf{y})\approx\boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{y})$ (see note[2]) while $G_{\sigma_3}(\frac{s-t}{N})\approx\mathbf{z}(s/N)^T\mathbf{z}(t/N)$ (see note[3]). With these approximations and simplification to G_{σ_1} described above, we rewrite our sequence compatibility kernel as:

$$K_S(\Pi_A,\Pi_B)\approx \frac{1}{\Lambda}\sum_{i\in\mathcal{I}_J}\sum_{s\in\mathcal{I}_N}\sum_{t\in\mathcal{I}_N}\left(\begin{bmatrix}\sqrt{\beta_1}\,\boldsymbol{\phi}(\mathbf{x}_{is}),\ \text{(see note}^2\text{)}\\ \sqrt{\beta_2}\,\mathbf{z}(s/N),\ \text{(see note}^3\text{)}\end{bmatrix}^T\cdot\begin{bmatrix}\sqrt{\beta_1}\boldsymbol{\phi}(\mathbf{y}_{it})\\ \sqrt{\beta_2}\mathbf{z}(t/N)\end{bmatrix}\right)^r \tag{8}$$

$$=\frac{1}{\Lambda}\sum_{i\in\mathcal{I}_J}\sum_{s\in\mathcal{I}_N}\sum_{t\in\mathcal{I}_N}\left\langle\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\,\boldsymbol{\phi}(\mathbf{x}_{is})\\ \sqrt{\beta_2}\,\mathbf{z}(s/N)\end{bmatrix},\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\boldsymbol{\phi}(\mathbf{y}_{it})\\ \sqrt{\beta_2}\mathbf{z}(t/N)\end{bmatrix}\right\rangle \tag{9}$$

$$=\sum_{i\in\mathcal{I}_J}\left\langle\frac{1}{\sqrt{\Lambda}}\sum_{s\in\mathcal{I}_N}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\,\boldsymbol{\phi}(\mathbf{x}_{is})\\ \sqrt{\beta_2}\mathbf{z}(s/N)\end{bmatrix},\frac{1}{\sqrt{\Lambda}}\sum_{t\in\mathcal{I}_N}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\boldsymbol{\phi}(\mathbf{y}_{it})\\ \sqrt{\beta_2}\mathbf{z}(t/N)\end{bmatrix}\right\rangle. \tag{10}$$

Expansion of Eq. (8) into Eq. (9) simply follows the notion of equivalence between the polynomial kernels and tensor outer-products as detailed in Eq. (4). Similarly, the summations in Eq. (9) can be absorbed into the dot-product in Eq. (10) because the inner-product is a linear operation in each of its arguments *e.g.*, $\langle\mathbf{v}_1+\mathbf{v}_2,\bar{\mathbf{v}}\rangle=\langle\mathbf{v}_1,\bar{\mathbf{v}}\rangle+\langle\mathbf{v}_2,\bar{\mathbf{v}}\rangle$. The physical meaning of the above equation is detailed in Figure 3. While the first-, second- and third-order outer-products are connected to the sample mean, covariance and co-skewness of features, our tensors are not mere counts of features, as explained next. As is clear, (10) expresses $K_S(\Pi_A,\Pi_B)$ as a sum of inner-products on third-order tensors ($r=3$), as shown in Figure 2c. While, using the dot-product as the inner-product is an option, other alternatives for tensors of order $r\geq 2$ can act on their spectrum, leading to better representations.

[2]In practice, Cartesian coordinates of joints $\mathbf{x},\mathbf{y}\in\mathbb{R}^3$ are fed into a kernel. Thus, in place of kernel G_{σ_2}, we use the sum kernel $G'_{\sigma_2}(\mathbf{x}-\mathbf{y})=G_{\sigma_2}(x_1-y_1)+G_{\sigma_2}(x_2-y_2)+G_{\sigma_2}(x_3-y_3)$ whose approximation is given as: $G'_{\sigma_2}(\mathbf{x}-\mathbf{y})\approx[\boldsymbol{\phi}(x_1;\{\boldsymbol{\zeta}_i\}_{i\in\mathcal{I}_{Z_2}});\boldsymbol{\phi}(x_2;\{\boldsymbol{\zeta}_i\}_{i\in\mathcal{I}_{Z_2}});\boldsymbol{\phi}(x_3;\{\boldsymbol{\zeta}_i\}_{i\in\mathcal{I}_{Z_2}})]^T[\boldsymbol{\phi}(y_1;\{\boldsymbol{\zeta}_i\}_{i\in\mathcal{I}_{Z_2}});\boldsymbol{\phi}(y_2;\{\boldsymbol{\zeta}_i\}_{i\in\mathcal{I}_{Z_2}});\boldsymbol{\phi}(y_3;\{\boldsymbol{\zeta}_i\}_{i\in\mathcal{I}_{Z_2}})]$ but for simplicity we refer to it in our formulations by its generic form $G_{\sigma_2}(\mathbf{x}-\mathbf{y})\approx\boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{y})$ because we can define $\phi(\mathbf{x})=[\phi(x_1);\phi(x_2);\phi(x_3)]$.

[3]Feature maps $\mathbf{z}(\cdot)\equiv\boldsymbol{\phi}(\cdot)$ from Eq. (2). We simply write $\mathbf{z}$ rather than $\boldsymbol{\phi}$ to denote these feat. maps as they encode the time/frame number (*c.f.* the body joints). Note that $\mathbf{z}(\cdot;\{\boldsymbol{\zeta}'_i\}_{i\in\mathcal{I}_{Z_3}})$ uses Z_3 pivots $\{\boldsymbol{\zeta}'_i\}_{i\in\mathcal{I}_{Z_3}}$ (see Figure 3).

An example is the so-called *burstiness* [81], which is a commonly encountered property that a given feature appears more/less often in a sequence than a statistically independent model predicts. Robust descriptors must be invariant w.r.t. the length of actions *e.g.*, a prolonged *hand waving* represents the same action as a short *hand wave*. Eigenvalue Power Normalization (EPN) [4] suppresses burstiness by acting on higher-order statistics (see Fig. 2c). By incorporating EPN, we generalize (10) as:

$$K^*_S(\Pi_A,\Pi_B)=\sum_{i\in\mathcal{I}_J}\left\langle \mathcal{G}\left(\frac{1}{\sqrt{\Lambda}}\sum_{s\in\mathcal{I}_N}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\boldsymbol{\phi}(\mathbf{x}_{is})\\ \sqrt{\beta_2}\mathbf{z}(s/N)\end{bmatrix}\right),\right.$$
$$\left.\mathcal{G}\left(\frac{1}{\sqrt{\Lambda}}\sum_{t\in\mathcal{I}_N}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\boldsymbol{\phi}(\mathbf{y}_{it})\\ \sqrt{\beta_2}\mathbf{z}(t/N)\end{bmatrix}\right)\right\rangle, \quad (11)$$

where the operator $\mathcal{G}$ performs EPN by applying power normalization to the spectrum of the third-order tensor (by taking the higher-order SVD). Note that in general $K^*_S(\Pi_A,\Pi_B)\not\approx K_S(\Pi_A,\Pi_B)$ as $\mathcal{G}$ is intended to manipulate the spectrum of $\boldsymbol{\mathcal{X}}$.

The final representation for linearized SCK becomes:

$$\boldsymbol{\mathcal{V}}_i=\mathcal{G}(\boldsymbol{\mathcal{X}}_i),\ \text{where}\ \boldsymbol{\mathcal{X}}_i=\frac{1}{\sqrt{\Lambda}}\sum_{s\in\mathcal{I}_N}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1}\,\boldsymbol{\phi}(\mathbf{x}_{is})\\ \sqrt{\beta_2}\mathbf{z}(s/N)\end{bmatrix}. \quad (12)$$

We replace the sum over the body-joint indexes in (11) by concatenating $\boldsymbol{\mathcal{V}}_i$ in (12) along the fourth tensor mode, thus defining $\boldsymbol{\mathcal{V}}=\left[\boldsymbol{\mathcal{V}}_i\right]_{i\in\mathcal{I}_J}^{\oplus_4}$. Suppose $\boldsymbol{\mathcal{V}}_A$ and $\boldsymbol{\mathcal{V}}_B$ are the corresponding fourth order tensors for Π_A and Π_B respectively. Then, we obtain:

$$K^*_S(\Pi_A,\Pi_B)=\langle\boldsymbol{\mathcal{V}}_A,\boldsymbol{\mathcal{V}}_B\rangle\,. \quad (13)$$

Note that tensors $\boldsymbol{\mathcal{X}}$ have the following properties: (i) super-symmetry $\boldsymbol{\mathcal{X}}_{i,j,k}=\boldsymbol{\mathcal{X}}_{\pi(i,j,k)}$ for indexes i,j,k and their permutation given by π, $\forall\pi$, and (ii) positive semi-definiteness of every slice, that is, $\boldsymbol{\mathcal{X}}_{:,:,s}\in\mathcal{S}^d_+$, for $s\in\mathcal{I}_d$. Thus, we use only the upper-simplices of $\boldsymbol{\mathcal{V}}_i$ which consist of $\binom{d+r-1}{r}$ coefficients (which is the total size of our final representation times the number of body-joints) rather than d^r, where d is the side-dimension of $\boldsymbol{\mathcal{V}}_i$ *i.e.*, $d=3Z_2+Z_3$ (see notes[2,3]), and Z_2 and Z_3 are the numbers of pivots used in the approximation of G_{σ_2} and G_{σ_3} (see notes[2,3]).

Next, we pass tensors $\boldsymbol{\mathcal{X}}$ via (i) slice-wise EPN (sEPN) operator or (ii) HOSVD-based tensor whitening EPN (tEPN) [4]. sEPN is faster but tEPN uses the entire tensor spectrum, thus being more accurate. The slice-wise EPN uses the Power-Euclidean dist. for rising matrices, slices of tensor tensor $\boldsymbol{\mathcal{X}}$, to the power of γ. Power norm. and re-stacking slices along the third mode yields:

$$\mathcal{G}(\boldsymbol{\mathcal{X}})=[\boldsymbol{\mathcal{X}}^{\gamma}_{:,:,s}]^{\oplus_3}_{s\in\mathcal{I}_d},\ \text{for}\ 0<\gamma\leq 1. \quad (14)$$

We note that $\mathcal{G}(\boldsymbol{\mathcal{X}})$ preserves listed earlier properties of tensors $\boldsymbol{\mathcal{X}}$ and it forms our final tensors $\boldsymbol{\mathcal{V}}$ for the action sequence.

The HOSVD-based tensor whitening EPN, proposed in [4], is defined by the following operator $\mathcal{G}$:

$$(\boldsymbol{\mathcal{E}};\boldsymbol{A}_1,...,\boldsymbol{A}_r)=\text{HOSVD}(\boldsymbol{\mathcal{X}}), \quad (15)$$
$$\hat{\boldsymbol{\mathcal{E}}}=\text{Sgn}(\boldsymbol{\mathcal{E}})\,|\boldsymbol{\mathcal{E}}|^{\gamma}, \qquad \left(\text{generally}\ \hat{\boldsymbol{\mathcal{E}}}=\hat{\mathcal{G}}(\boldsymbol{\mathcal{E}})\right) \quad (16)$$
$$\hat{\boldsymbol{\mathcal{V}}}=((\hat{\boldsymbol{\mathcal{E}}}\times_1\boldsymbol{A}_1)\,...)\times_r\boldsymbol{A}_r, \quad \left(\text{think}\ \hat{\boldsymbol{\mathcal{V}}}=\boldsymbol{\mathcal{X}}^{\frac{1}{2}}\right) \quad (17)$$
$$\mathcal{G}(\boldsymbol{\mathcal{X}})=\text{Sgn}(\hat{\boldsymbol{\mathcal{V}}})\,|\hat{\boldsymbol{\mathcal{V}}}|^{\gamma^*}. \quad (18)$$

In the above equations, we distinguish the core tensor $\boldsymbol{\mathcal{E}}$, its power-normalized variant $\hat{\boldsymbol{\mathcal{E}}}$ with factor weights evened out by rising them to the power $0<\gamma\leq 1$, singular vector matrices $\boldsymbol{A}_1,...,\boldsymbol{A}_r$ and operation $\times_r$ which is the so-called tensor-product in mode r.

As our tensors $\boldsymbol{\mathcal{X}}$ are super-symmetric, we note that $\boldsymbol{A}_1=\boldsymbol{A}_2=...=\boldsymbol{A}_r$. However, the kernel which is proposed in Section 4.4 leads to a non-symmetric tensor representation. We refer the reader to paper [4] for the detailed description of the above steps.

Eq. (16) has a more general form $\hat{\boldsymbol{\mathcal{E}}}=\hat{\mathcal{G}}(\boldsymbol{\mathcal{E}})$, where $\hat{\mathcal{G}}$ can be any power normalizing function [2]. In Sec. D, we derive the exact interpretation of Eq. (15-18) for $\hat{\mathcal{G}}=\text{Sgn}(\boldsymbol{\mathcal{E}})\,(1-(1-|\boldsymbol{\mathcal{E}}|)^{\bar{N}})$ for which $\text{Sgn}(\boldsymbol{\mathcal{E}})\,|\boldsymbol{\mathcal{E}}|^{\gamma}$ is an approximation [2]. We prove in Sec. D that EPN performs in fact a spectral detection of higher-order occurrences of features, the base of fine-grained systems [1, 2]. Figure 9 illustrates details of such a spectral detection.

4.4 Dynamics Compatibility Kernel

The SCK kernel that we described above captures the inter-sequence alignment, whereas the intra-sequence spatio-temporal dynamics is lost. Thus, we propose a novel *dynamics compatibility kernel* (DCK). In what follows, we use the absolute coordinates of the joints in our kernel and follow notations from the prev. section.

DCK for two action sequences Π_A and Π_B is defined as:

$$K_D(\Pi_A,\Pi_B)=$$
$$\frac{1}{\Lambda}\sum_{\substack{(i,s)\in\mathcal{J},\\(i',s')\in\mathcal{J},\\ i'\neq i,s'\neq s}}\sum_{\substack{(j,t)\in\mathcal{J},\\(j',t')\in\mathcal{J},\\ j'\neq j,t'\neq t}} G'_{\sigma'_1}(i-j,i'-j')\,G_{\sigma'_2}((\mathbf{x}_{is}-\mathbf{x}_{i's'})-(\mathbf{y}_{jt}-\mathbf{y}_{j't'}))\cdot$$
$$\cdot\,G'_{\sigma'_3}(\frac{s-t}{N},\frac{s'-t'}{N})\,G'_{\sigma'_4}(s-s',t-t'). \quad (19)$$

In contrast to SCK in (7), the DCK kernel uses the intra-sequence joint differences, thus capturing the dynamics, which is then compared against dynamics of other sequences.

Figures 4a-4c depict schematically how DCK captures co-occurrences. As in SCK, the first kernel, $G'_{\sigma'_1}$, captures the sensor uncertainty in body-keypoint detection, and is assumed to be a delta function in this paper. The second kernel, $G_{\sigma'_2}$, models the spatio-temporal co-occurrences of the body-joints. Temporal alignment kernels, expressed as $G'_{\sigma'_3}(\boldsymbol{\alpha},\boldsymbol{\beta})=G_{\sigma'_3}(\boldsymbol{\alpha})G_{\sigma'_3}(\boldsymbol{\beta})$, encode temporal start- and end-points from (s,s') and (t,t'). Finally, $G_{\sigma'_4}$ limits contributions of dynamics between temporal points if they are distant from each other, *i.e.* if $s'\gg s$ or $t'\gg t$ and σ'_4 is small. Similarly to SCK, the standard deviations σ'_2 and σ'_3 control the selectivity over spatio-temporal dynamics of body-joints and their temporal shift-invariance for the start and end points, resp. As discussed for SCK, the practical extensions from footnotes[1,2,3] also apply to DCK *e.g.*, the definition of $\mathbf{z}$, the pivot numbers Z_2 and Z_3 for $G_{\sigma'_2}$ and $G_{\sigma'_3}$ kernels.

Based on the above formulations, Section A shows that the linearization of DCK admits the form:

$$K_D(\Pi_A,\Pi_B)\approx \quad (20)$$
$$\sum_{\substack{i\in\mathcal{I}_J,\\ i'\in\mathcal{I}_J:\\ i'\neq i}}\left\langle\frac{1}{\sqrt{\Lambda}}\sum_{\substack{s\in\mathcal{I}_N,\\ s\in\mathcal{I}_N:\\ s'\neq s}}G_{\sigma'_4}(s-s')\left(\boldsymbol{\phi}(\mathbf{x}_{is}-\mathbf{x}_{i's'})\cdot\mathbf{z}(\frac{s}{N})^T\right)\uparrow\otimes\mathbf{z}(\frac{s'}{N}),\right.$$
$$\left.\frac{1}{\sqrt{\Lambda}}\sum_{\substack{t\in\mathcal{I}_N,\\ t\in\mathcal{I}_N:\\ t'\neq t}}G_{\sigma'_4}(t-t')\left(\boldsymbol{\phi}(\mathbf{y}_{it}-\mathbf{y}_{i't'})\cdot\mathbf{z}(\frac{t}{N})^T\right)\uparrow\otimes\mathbf{z}(\frac{t'}{N})\right\rangle.$$

Equation (20) expresses $K_D(\Pi_A,\Pi_B)$ as a sum over inner-products on third-order non-symmetric tensors (c.f. Section 4.3 where the proposed kernel results in an inner-product between super-symmetric tensors). However, we can decompose each of

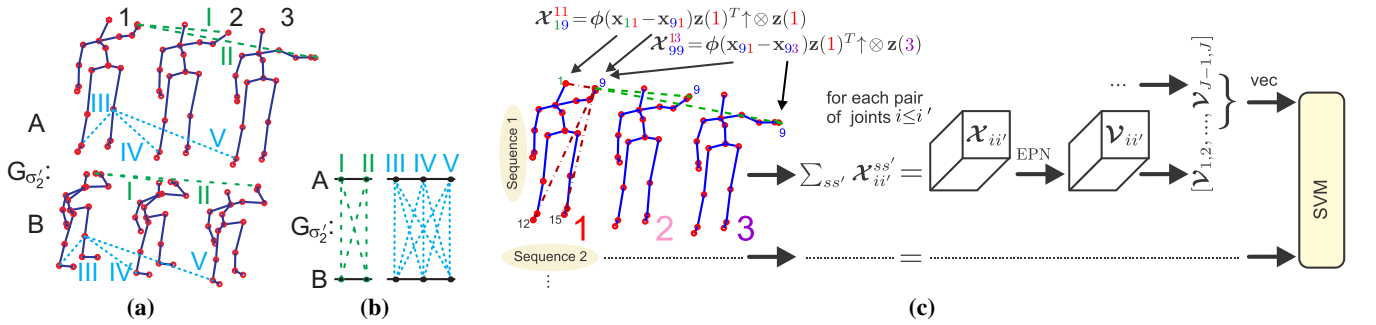


Fig. 4: Figure 4a shows that kernel $G_{\sigma_2'}$ in DCK captures spatio-temporal dynamics by measuring displacement vectors from any given body-joint to remaining joints spatially- and temporally-wise (*i.e.* see dashed lines). Figure 4b shows that comparisons performed by $G_{\sigma_2'}$ for any selected two joints are performed all-against-all temporally-wise which is computationally expensive. Figure 4c shows the encoding steps in the proposed linearization which is fastn. We collect all $\boldsymbol{\mathcal{X}}_{ii'}$ for joints $i \le i'$, whiten them by EPN to obtain $\boldsymbol{\mathcal{V}}_{ii'}$, concatenate, vectorize them per video with 'vec' and fed to an SVM. We introduced color-coded body joints/frame numbers to show how we assemble a single $\boldsymbol{\mathcal{X}}_{ii'}$.

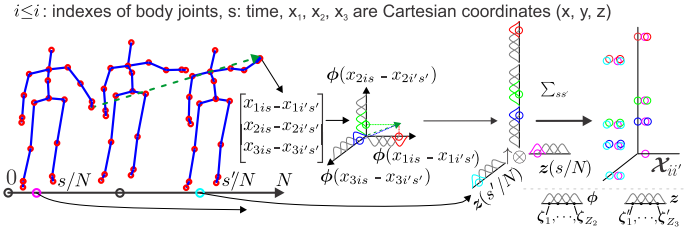


Fig. 5: Third-order statistics from Eq. (19) can be understood by studying the linearization in Eq. (20). For a given pair of joints $i \le i'$ at times s/N and s'/N (normalized frame numbers), we embed displacement vectors $\mathbf{x}_{is} - \mathbf{x}_{i's'}$ of 3D joint coordinates $\mathbf{x}_{is}$and $\mathbf{x}_{i's'}$ via function $\boldsymbol{\phi}(\cdot)$ into a non-linear Hilbert space representing an RBF kernel according to Eq. (2). Similarly, we embed the starting and ending times s/N and s'/N via function $\mathbf{z}(\cdot)$ (also by Eq. (2)). Finally, $\otimes$ performs the third-order outer-product on concatenated displacement and time embeddings aggregated next over frames s and s' (note $\sum_{ss'}$). The interpretation: the Gaussians 'soft-divide' the Cartesian coordinate system along x, y, z direction, resp., as well as time direction (s/N and s'/N). We project displacements along x, y, z directions of Cartesian coordinates and assign each projection to Gaussians. Thus, triplets $([x;y;z],s,s')$ assigned into such a 'soft-divided' space capture locally displacements of pairs of joints on the time grid (3-way soft-histogram). For DCK $\oplus$ in Section 4.6 we use velocity vectors $\frac{\mathbf{x}_{is} - \mathbf{x}_{i's'}}{\max(1,|s'-s|)}$ (*c.f.* displacement vectors) with short- and long-term estimates depending on $s'-s$ (3-way soft-histogram of short- and long-term speeds).

these tensors with a variant of EPN, which involves Higher Order Singular Value Decomposition (HOSVD), into factors stored in the so-called core tensor, and equalize the contributions of these factors to prevent bursts in the spatio-temporal co-occurrence dynamics of actions. For example, consider that a long *hand wave* versus a short *hand wave* yield different temporal statistics, that is, the prolonged action results in bursts. However, the final representation described below becomes invariant to bursts.

The final representation for linearized DCK with a non-linear operator $\boldsymbol{\mathcal{G}}$ introduced into Eq. (20) to prevent burstiness becomes: $\boldsymbol{\mathcal{V}}_{ii'} = \boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}_{ii'})$, where (21)

$$\boldsymbol{\mathcal{X}}_{ii'} = \frac{1}{\sqrt{\Lambda}} \sum_{s,s' \in \mathcal{I}_N : s \not= s} G_{\sigma_4'}(s-s') \left(\boldsymbol{\phi}(\mathbf{x}_{is} - \mathbf{x}_{i's'}) \cdot \mathbf{z}\left(\frac{s}{N}\right)^T \right) \uparrow \otimes \mathbf{z}\left(\frac{s'}{N}\right).$$

The summation over pairs of body-joint indexes in (20) is equivalent to the concatenation of $\boldsymbol{\mathcal{V}}_{ii'}$ from (21) along the fourth mode. Thus, we obtain tensor representations $\left[\boldsymbol{\mathcal{V}}_{ii'}\right]^{\oplus_4}_{i>i':\, i,i' \in \mathcal{I}_J}$ for sequence Π_A and $\left[\bar{\boldsymbol{\mathcal{V}}}_{ii'}\right]^{\oplus_4}_{i>i':\, i,i' \in \mathcal{I}_J}$ for sequence Π_B.

The physical meaning of Eq. (21) is detailed in Figure 5. The dot-product can be now applied between these representations to compare them. Tensors $\boldsymbol{\mathcal{X}}$ in (21) are non-symmetric. Thus, for the operator $\boldsymbol{\mathcal{G}}$, we choose the HOSVD-based tensor whitening EPN, that is, tEPN defined in Eq. (15-18).

4.5 Sequence Compatibility Kernel 'Plus' (SCK $\oplus$)

Below, we extend the SCK formulation from Section 4.3 to aggregate over multiple subsequences extracted from the input sequence. Intuitively, this process is an equivalent of extracting local descriptors from images to attain so-called shift-invariance to the object location. As it is unlikely that relevant motion patterns stretch throughout a sequence, a specific pattern associated with some action classes may appear in one/few subsequences. Moreover, in what follows next, we will allow the aggregation to run over multiple modalities $q \in \mathcal{I}_Q$ *e.g.*, we use 3D body-joints and/or frame-wise CNN classification scores from RGB videos and/or optical flow. Thus, we can define our multimodal pose sequence Π as:

$$\Pi = \left\{ \mathbf{x}_{is}^{(q)} \in \mathbb{R}^{W_q}, i \in \mathcal{I}_J, s \in \mathcal{I}_M, q \in \mathcal{I}_Q \right\}, \qquad (22)$$

where $W_1 = 3$, J is the total number of body-joints, W_q for $q > 1$ equals the size of modality q other than body-joints. Note that if modality $q > 1$ is global rather than per-joint specified, we can replicate it *e.g.*, $\mathbf{x}_{1s}^{(q)} = ... = \mathbf{x}_{Js}^{(q)}$.

SCK $\oplus$ on a pair of sequences Π_A and Π_B of length M and N is defined as: (23)

$$K_{S\oplus}(\Pi_A, \Pi_B) = \frac{1}{\Lambda} \sum_{i \in \mathcal{I}_J} \sum_{\substack{\tau \in \mathcal{P}_A \\ \tau' \in \mathcal{P}_B}} \sum_{\substack{u \in \mathcal{U}_\tau \\ u' \in \mathcal{U}_{\tau'}}} \sum_{\substack{s \in \mathcal{S}_\tau \\ t \in \mathcal{S}_{\tau'}}} \Bigg(\sum_{q \in \mathcal{I}_Q} \beta_1^{(q)} G_{\sigma_2^{(q)}} \left(\mathbf{x}_{i,u+s}^{(q)} - \mathbf{y}_{i,u'+t}^{(q)} \right) + \beta_2 G_{\sigma_3}(f(s,\mathcal{S}_\tau) - f(t,\mathcal{S}_{\tau'})) + \beta_3 G_{\sigma_4}(f(u,\mathcal{U}_\tau^A) - f(u',\mathcal{U}_{\tau'}^B)) + \beta_4 G_{\sigma_5}(f(\tau,\mathcal{P}_A) - f(\tau',\mathcal{P}_B)) \Bigg)^r.$$

Symbols $\mathcal{P}_A$ and $\mathcal{P}_B$ denote subsequence lengths, $\mathcal{P}_A = \mathcal{P}_B = \mathcal{P}$ is a possible assertion to make, so that *i.e.* $\mathcal{P} = \{8, 10, 12, ..., 20\}$. Moreover, $\mathcal{U}_\tau^A$ and $\mathcal{U}_\tau^B$ are sets of all positions in sequences π_A and π_B for subsequences of lengths τ and τ', respectively, *i.e.*, if $N = 100$ and $\tau = 20$ then $\mathcal{U}_{20}^A = \{1, 3, 5, ..., 79\}$ is an example of a possible choice. Furthermore, $\mathcal{S}_\tau$ and $\mathcal{S}_{\tau'}$ are sets of all sampling positions in subsequences of lengths τ and τ', *i.e.*, if $\tau = 20$ then $\mathcal{S}_{20} = \{0, 1, 2, ..., 19\}$ is an example of a possible choice. We define a function $f(s, \mathcal{S}) = \frac{s - \mathcal{S}^{min}}{\mathcal{S}^{max} - \pi^{min}}$ which performs normalization on s w.r.t. set π, and $\mathcal{S}^{min}$ and $\mathcal{S}^{max}$ denote the smallest and largest element of set $\mathcal{S}$, respectively.

Moreover, normalizations $f(u,\mathcal{U})$ and $f(\tau,\mathcal{P})$ are defined by analogy, $\Lambda=\Lambda_A\cdot\Lambda_B=(|\mathcal{I}_J|\cdot|\mathcal{P}_A|\cdot|\mathcal{U}_\tau^A|\cdot|\mathcal{S}_\tau|)\cdot(|\mathcal{I}_J|\cdot|\mathcal{P}_B|\cdot|\mathcal{U}_\tau^B|\cdot|\mathcal{S}_{\tau'}|)$. For simplicity, we do not model the within-sequence similarity between the body joints in contrast to Eq. (7), thus we skip G_{σ_1}. Kernels $G_{\sigma_2^{(i)}}$ capture the compatibility between body-joint locations $\mathbf{x}$ and $\mathbf{y}$ in a subsequence. Kernel G_{σ_3} measures the temporal alignment of two pose snippets in the given two subsequences. Kernel G_{σ_4} measures the temporal alignment of two subsequences in two sequences. Lastly, G_{σ_5} measures the match of two subsequence lengths. Weight factors $\beta_1^{(q)}\geq 0$ adjust the importance of each modality $q\in\mathcal{I}_Q$. Weight $\beta_2\geq 0$ is the importance of the temporal alignment of snippets within subsequences. Weight $\beta_3\geq 0$ is the importance of the temporal alignment of subsequences within sequences. Weight $\beta_4\geq 0$ is the importance of the match of two subsequence lengths. We let $\sum_q\beta_1^{(q)}+\beta_2+\beta_3+\beta_4=1$. Parameters $\sigma_2^{(q)}$ in $G_{\sigma_2^{(q)}}$ and $\beta_1^{(q)}$ are set per modality *e.g.*, for the 3D body-joints we chose $G_{\sigma_2^{(1)}}$ to be an RBF kernel, for frame-wise class predictions obtained from CNNs applied on (i) RGB and (ii) optical flow frames we choose $G_{\sigma_2^{(2)}}$ and $G_{\sigma_2^{(3)}}$ to be linear kernels (with no parameters). As previously, r denotes the order of captured statistics *i.e.*, $r=3$.

Below, we present the process of linearization of our kernel which follows the reasoning from Section 3.2 and Eq. (3). However, we feel it is interesting to show how various kernel components translate to various statistics encoded by the tensor:

i. $G_{\sigma_2^{(q)}}(\mathbf{x}-\mathbf{y})\approx\boldsymbol{\phi}^{(q)}(\mathbf{x})^T\boldsymbol{\phi}^{(q)}(\mathbf{y})$ (see note[2]) and, in order to reflect the choice of par. $\sigma_2^{(q)}$ for index q, we write $\boldsymbol{\phi}^{(q)}$,
ii. $G_{\sigma_3}(f(s,\mathcal{S}_\tau)-f(t,\mathcal{S}_{\tau'}))\approx\mathbf{z}'(f(s,\mathcal{S}_\tau)^T\mathbf{z}'(f(t,\mathcal{S}_{\tau'}))$,
iii. $G_{\sigma_4}(f(u,\mathcal{U}_\tau)-f(u',\mathcal{U}_{\tau'}))\approx\mathbf{z}''(f(u,\mathcal{U}_\tau)^T\mathbf{z}''(f(u',\mathcal{U}_{\tau'}))$,
iv. $G_{\sigma_5}(f(\tau,\mathcal{P}_A)-f(\tau',\mathcal{P}_B))\approx\mathbf{z}'''(f(\tau,\mathcal{P}_A)^T\mathbf{z}'''(f(\tau',\mathcal{P}_B))$.

With these approximations at hand, we rewrite our sequence compatibility kernel 'plus' as:

$$K_{S^\oplus}(\Pi_A,\Pi_B)\approx \frac{1}{\Lambda}\sum_{i\in\mathcal{I}_J}\sum_{\substack{\tau\in\mathcal{P}_A\\ \tau'\in\mathcal{P}_B}}\sum_{\substack{u\in\mathcal{U}_\tau\\ u'\in\mathcal{U}_{\tau'}}}\sum_{\substack{s\in\mathcal{S}_\tau\\ t\in\mathcal{S}_{\tau'}}}\left(\begin{bmatrix}\sqrt{\beta_1^{(1)}}\,\boldsymbol{\phi}(\mathbf{x}_{i,u+s}^{(1)})\\ \dots\\ \sqrt{\beta_1^{(Q)}}\,\boldsymbol{\phi}(\mathbf{x}_{i,u+s}^{(Q)})\\ \sqrt{\beta_2}\,\mathbf{z}'(f(s,\mathcal{S}_\tau))\\ \sqrt{\beta_3}\,\mathbf{z}''(f(u,\mathcal{U}_\tau))\\ \sqrt{\beta_4}\,\mathbf{z}'''(f(\tau,\mathcal{P}_A))\end{bmatrix}^T\cdot\begin{bmatrix}\sqrt{\beta_1^{(1)}}\boldsymbol{\phi}(\mathbf{y}_{i,u'+t}^{(1)})\\ \dots\\ \sqrt{\beta_1^{(Q)}}\boldsymbol{\phi}(\mathbf{y}_{i,u'+t}^{(Q)})\\ \sqrt{\beta_2}\mathbf{z}'(f(t,\mathcal{S}_{\tau'}))\\ \sqrt{\beta_3}\mathbf{z}''(f(u',\mathcal{U}_{\tau'}))\\ \sqrt{\beta_4}\mathbf{z}'''(f(\tau',\mathcal{P}_B))\end{bmatrix}\right)^r= \tag{24}$$

$$\sum_{i\in\mathcal{I}_J}\left\langle\boldsymbol{\mathcal{G}}\left(\frac{1}{\Lambda_A}\sum_{\tau\in\mathcal{P}_A}\sum_{u\in\mathcal{U}_\tau}\sum_{s\in\mathcal{S}_\tau}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1^{(1)}}\,\boldsymbol{\phi}(\mathbf{x}_{i,u+s}^{(1)})\\ \dots\\ \sqrt{\beta_1^{(Q)}}\,\boldsymbol{\phi}(\mathbf{x}_{i,u+s}^{(Q)})\\ \sqrt{\beta_2}\,\mathbf{z}'(f(s,\mathcal{S}_\tau))\\ \sqrt{\beta_3}\,\mathbf{z}''(f(u,\mathcal{U}_\tau))\\ \sqrt{\beta_4}\,\mathbf{z}'''(f(\tau,\mathcal{P}_A))\end{bmatrix}\right),\ \boldsymbol{\mathcal{G}}\left(\frac{1}{\Lambda_B}\sum_{\tau'\in\mathcal{P}_B}\sum_{u'\in\mathcal{U}_{\tau'}}\sum_{t\in\mathcal{S}_{\tau'}}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1^{(1)}}\boldsymbol{\phi}(\mathbf{y}_{i,u'+t}^{(1)})\\ \dots\\ \sqrt{\beta_1^{(Q)}}\boldsymbol{\phi}(\mathbf{y}_{i,u'+t}^{(Q)})\\ \sqrt{\beta_2}\mathbf{z}'(f(t,\mathcal{S}_{\tau'}))\\ \sqrt{\beta_3}\mathbf{z}''(f(u',\mathcal{U}_{\tau'}))\\ \sqrt{\beta_4}\mathbf{z}'''(f(\tau',\mathcal{P}_B))\end{bmatrix}\right)\right\rangle \tag{25}$$

In the above equation, we set $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}})=\boldsymbol{\mathcal{X}}$ for Eq. (25) to be equivalent to Eq. (24). However, similarly to considerations in Section 4.3, a commonly encountered adversity in aggregated representations, the *burstiness*, requires some suppression. To this end, we let operator $\boldsymbol{\mathcal{G}}$ in Eq. (25) perform tEPN on the spectrum of the third-order tensor.

The final representation for linearized SCK $\oplus$ becomes:

$$\boldsymbol{\nu}_i=\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}_i),\ \text{where}\ \boldsymbol{\mathcal{X}}_i=\frac{1}{\Lambda_A}\sum_{\tau\in\mathcal{P}_A}\sum_{u\in\mathcal{U}_\tau}\sum_{s\in\mathcal{S}_\tau}\uparrow\otimes_r\begin{bmatrix}\sqrt{\beta_1^{(1)}}\,\boldsymbol{\phi}(\mathbf{x}_{i,u+s}^{(1)})\\ \dots\\ \sqrt{\beta_1^{(Q)}}\,\boldsymbol{\phi}(\mathbf{x}_{i,u+s}^{(Q)})\\ \sqrt{\beta_2}\,\mathbf{z}'(f(s,\mathcal{S}_\tau))\\ \sqrt{\beta_3}\,\mathbf{z}''(f(u,\mathcal{U}_\tau))\\ \sqrt{\beta_4}\,\mathbf{z}'''(f(\tau,\mathcal{P}_A))\end{bmatrix}. \tag{26}$$

We can further replace the summation over the body-joint indexes in (25) by concatenating $\boldsymbol{\nu}_i$ in (26) along the fourth tensor mode, thus defining $\boldsymbol{\nu}=\left[\boldsymbol{\nu}_i\right]_{i\in\mathcal{I}_J}^{\oplus_4}$. Suppose $\boldsymbol{\nu}_A$ and $\boldsymbol{\nu}_B$ are the corresponding fourth order tensors for Π_A and Π_B, then we have:

$$K_{S^\oplus}^*(\Pi_A,\Pi_B)=\langle\boldsymbol{\nu}_A,\boldsymbol{\nu}_B\rangle\,. \tag{27}$$

Note that in general $K_{S^\oplus}^*(\Pi_A,\Pi_B)\not\approx K_{S^\oplus}(\Pi_A,\Pi_B)$ as $\boldsymbol{\mathcal{G}}$ manipulates the spectrum of $\boldsymbol{\mathcal{X}}$. Finally, for our final representation, we use only the upper-simplices of $\boldsymbol{\nu}_i$ which consist of $\binom{d+r-1}{r}$ coefficients each, rather than d^r, where d is the side-dimension of $\boldsymbol{\nu}_i$ *i.e.*, $d=3Z_2^{(1)}+\ldots+Z_2^{(Q)}+Z_3+Z_4+Z_5$ (see notes[2,3]), and $Z_2^{(1)},\ldots,Z_2^{(Q)}$ and Z_3,Z_4,Z_5 are the numbers of pivots used in the approximation of $G_{\sigma_2^{(1)}},\ldots,G_{\sigma_2^{(Q)}}$ and $G_{\sigma_3},G_{\sigma_4},G_{\sigma_5}$ (see notes[2,3]).

4.6 Dynamics Compatibility Kernel 'Plus' (DCK $\oplus$)

Below, we apply the aggregation over subsequences to our DCK kernel. We follow the same steps as for SCK $\oplus$ (Section 4.5) except that our subsequences for DCK $\oplus$ have a fixed length. For a pair of sequences Π_A and Π_B of length M and N, we have:

$$K_{D^\oplus}(\Pi_A,\Pi_B)= \frac{1}{\Lambda'}\sum_{u,u'\in\mathcal{U}_\tau}K_D(\Pi'_{A,\tau,u},\Pi'_{B,\tau,u'})G_{\sigma_4}(f(u,\mathcal{U}_\tau^A)-f(u',\mathcal{U}_\tau^B)), \tag{28}$$

where τ is a length of subsequences. $K_D(\Pi'_{A,\tau,u},\Pi_{B',\tau,u'})$ is defined in Eq. (19). However, we use velocity vectors $\frac{\mathbf{x}_{is}-\mathbf{x}_{i's'}}{\max(1,|s'-s|)}$ (*c.f.* displacement vectors in DCK) with short- and long-term estimates depending on s'–s. Figure 5 provides an interpretation of this kernel. $K_D(\Pi'_{A,\tau,u},\Pi_{B',\tau,u'})$ is evaluated over subsequences $\Pi'_{A,\tau,u}$ and $\Pi'_{B,\tau,u'}$ sampled from Π_A and Π_B according to sets of sampling coordinates $\mathcal{S}_{\tau,u}=\{\mathcal{S}_\tau\}+u$ and $\mathcal{S}_{\tau,u'}=\{\mathcal{S}_\tau\}+u'$ of length τ which are shifted by locations u and u' according to $\mathcal{U}_\tau$. Lastly, $\Lambda'=|\mathcal{U}_\tau^A|\cdot|\mathcal{U}_\tau^B|$. The remaining symbols follow definitions in Section 4.5. Kernel in Eq. (28) is then linearized in the similar manner to Eq. (19) which results in linearization similar to Eq. (21) but containing an additional mode corresponding to linearization of kernel G_{σ_4}. We skip this derivation for brevity.

5 Experiments

Below, we present experiments on our models on seven popular datasets. For datasets based on 3D skeletons, we use (i) the UTKinect-Action [31], (ii) Florence3D-Action [32], (iii) MSR-Action3D [33], and (iv) Kinetics [36] (where stated). For datasets based on RGB frames, we use (v) the fine-grained MPII Cooking Activities [25] and (vi) HMDB-51 [34] datasets. For experiments on the 3D skeletons fused with RGB frames, we use (vii) large scale NTU-RGBD [35] dataset. We also evaluate the influence of various hyper-parameters, such as the number of pivots Z used for linearizing the body-joint and temporal kernels, and the impact of

Eigenvalue Power Normalization (we vary the factor equalization). We evaluate our older SCK and DCK kernels, and their newer counterparts SCK$\oplus$ and DCK$\oplus$. For skeletons, we feed them directly to our kernel representations while RGB-based datasets are firstly encoded by the two-stream CNN [17] or the I3D [28].

5.1 Datasets

UTKinect-Action [31] consists of 10 actions performed twice by 10 different subjects, and has 199 action sequences. The dataset provides 3D coordinate annotations of 20 body-joints for every frame. The dataset was captured with a stationary Kinect sensor and contains significant viewpoint and intra-class variations.

Florence3D-Action [32] dataset consists of 9 actions performed 2–3$\times$ by 10 different subjects and it has 215 action sequences. 3D coordinate annotations of 15 body-joints captured with a Kinect sensor are provided. Significant intra-class variations are present *i.e.*, the same action articulated with the left/right hand, and actions like *drinking/performing a phone call* can be seen as fine-grained.

MSR-Action3D [33] dataset is comprised of 20 actions performed 2–3$\times$ by 10 different subjects and it has 567 action sequences. 3D coordinates of 20 body-joints captured by a Kinect-like depth sensor are provided. MSR-Action3D has strong inter-class similarity.

In the above datasets, we use the cross-subject test setting (unless stated otherwise), in which half of the subjects are used for training and the remaining half for testing. Similarly, we divide the training set into two halves for the purpose of training/validation.

NTU-RGBD [35] is by far the largest 3D skeleton-based video action recognition dataset. It has 56880 video sequences across 60 classes, 40 subjects, and 80 views. The videos have on average 70 frames and consist of people performing various actions. Each frame is annotated with 25 human skeletal keypoints (some videos have multiple subjects). Two evaluation protocols are used for this dataset, namely, cross-subject and cross-view evaluation. This dataset can be considered as having many fine-grained classes *e.g.*, *make a phone call*, *playing with phone*, *punching other person*, *pushing other person*, *pat on back of other person*, *etc*.

MPII Cooking Activities [25] dataset consists of high-resolution videos of cooking activities/people cooking various dishes. There are 64 distinct activities spread across 3748 video clips and one background activity (1861 clips). Activities include coarse actions *e.g.*, *opening refrigerator*, and fine-grained actions *e.g.*, *peel*, *slice*, *cut apart* (see Figure 6). This dataset is challenging due to (i) unbalanced action classes, (ii) significant intra-class differences (each subject cooks according to their own style). We use the mean Average Precision (mAP) over 7-fold cross-validation.

HMDB-51 [34] dataset is a popular video benchmark for human action recognition, consisting of 6766 Internet videos over 51 classes. Each video has about 20–1000 frames. We report the average classification accuracy on standard three-fold splits.

Kinetics [36] contains $\sim$300000 clips from YouTube which cover 400 human action classes, ranging from daily activities, sports scenes, to complex interactions. Each clip is $\sim$10 seconds long.

5.2 Experimental Setup

For our experiments, we distinguish four configurations: (i) for UTKinect-Action, Florence3D-Action and MSR-Action3D that provide 3D body-joints, we feed sequences of 3D body-joints to our kernel(s), (ii) for MPII Cooking Activities, HMDB-51 and NTU-RGBD that provide RGB frames, we train a two-stream ResNet-152 model (as in [17]) taking RGB frames (in the spatial stream) and a stack of optical flow frames (in the temporal stream) as input to obtain classification scores per frame per stream which are then passed to our kernel, (iii) for NTU-RGBD which contains both 3D body-joints and RGB frames, we investigate both such inputs separately as well as their combination, and (iv) for Kinetics, we use skeletons and combine ST-GCN with SCK.

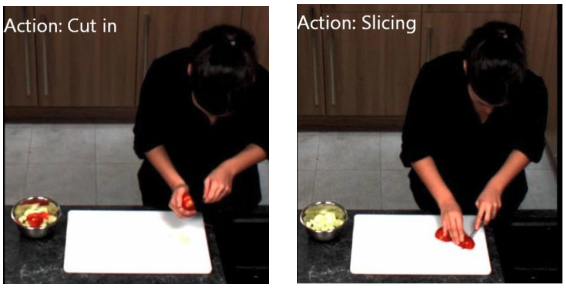
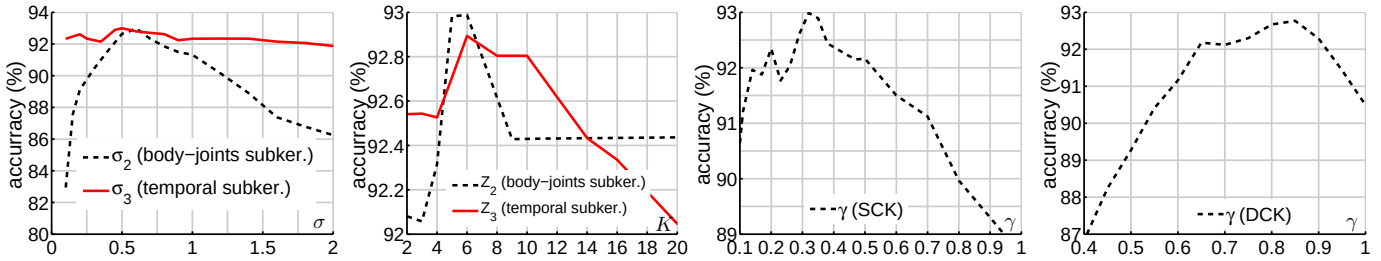


Fig. 6: Fine-grained action instances (MPII Cooking Activities [25]) from two different action categories: *cut-in* (left) and *slicing* (right).

For the sequence compatibility kernel on sequences of 3D body-joints, we first normalized all body-keypoints with respect to the hip joints across frames, as indicated in Section 4.3. We also normalized lengths of all body-parts w.r.t. to a reference skeleton. This setup follows pre-processing of [12]. For our dynamics compatibility kernel, we use unnormalized body-joints and assume that the displacements of body-joint coordinates across frames capture their temporal evolution implicitly. For the sequence compatibility kernel on classifier scores, we take the scores before they are passed through the logistic function and we apply a rectifier.

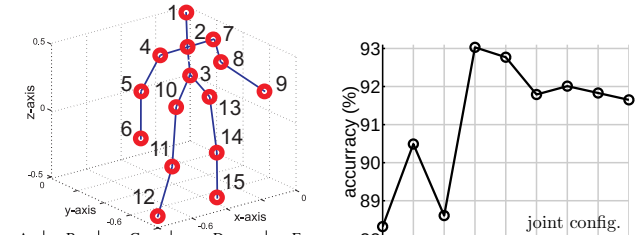
CNN Training. To extract features with CNN, we train a two-stream ResNet-152 model [17] taking RGB frames (in the spatial stream) and a stack of optical flow frames (in the temporal stream) from a given training split as input. For optical flow, we use the Large Displacement Optical Flow (LDOF) [82]. We use the classifier predictions from each stream as inputs to our kernels. The two streams of the CNN are trained separately on the respective input modalities against a softmax cross-entropy loss. We simply follow the standard training protocols from [17]. For fine-tuning, we used a fixed learning rate of $1e$–4 and a momentum of 0.9. For the MPII Cooking Activities dataset, we used the sequences from the training set for training the CNNs (1992 sequences) and those from the validation set (615 sequences) to check for overfitting. For HMDB-51, we use three standard splits provided with the dataset. For NTU-RGBD dataset in the cross-subject evaluation, the training and testing sets have 40320 and 16560 samples, respectively. For NTU-RGBD dataset in the cross-view evaluation, the training and testing sets have 37920 and 18960 samples, respectively. We use 70% of the training set for training and 30% for validation. To train SVM, we simply vectorize our tensors and set $c=1e-2$.

To stay competitive w.r.t. the state of the art, we additionally use two newer backbones such as (i) Spatial Temporal Graph Convolutional Network (ST-GCN) [75] and (ii) Two-Stream Inflated 3D ConvNet (I3D) [28]. For ST-GCN, we train it on skeletal sequences from NTU and Kinetics [36] datasets following the standard protocols. For Kinetics, we follow approach [36] and use skeletons extracted with OpenPose [5]. Finally, we combine our vectorized tensors from SCK or SCK$\oplus$ with the output of the last layer of ST-GCN preceding the classifier, and feed such a representation into the cross-entropy loss. As SCK is a shallow approach, we expect it to be highly complementary with ST-



(a) (b) (c) (d)

Fig. 7: Figure 7a illustrates the classification accuracy on Florence3d-Action for the sequence compatibility kernel when varying radii σ_2 (body-joints subkernel) and σ_3 (temporal subkernel). Figure 7b evaluates behavior of SCK w.r.t. the number of pivots Z_2 and Z_3. Figure 7c demonstrates effectiveness of our slice-wise Eigenvalue Power Normalization in tackling burstiness by varying parameter γ. Figure 7d shows effectiveness of equalizing the factors in non-symmetric tensor representation by HOSVD Eigenvalue Power Normalization by varying γ.



A	B	C	D	E
6,9	1,6,9	6,9,12,15	4,6,7,9,11,14	4,6,7,9,
F	G	H	I	11,12,
4-15	1,4-15	1,2,4-15	1-15	14,15

(a) (b)

Fig. 8: Figure 8a enumerates the body-joints in the Florence3D-Action dataset. The table below lists subsets A-I of the body-joints used to build representations eval. in Figure 8b, which shows the accuracy of our dynamics compatibility kernel w.r.t. these subsets.

GCN. For I3D network, we train it on subsequences extracted from HMDB51 and MPII. We use RGB and optical flow (LDOF) streams. We extract subsequences of length 48, 64, 80, 96 given strides 1, 2 and 3. Then, subsequences shorter than 64 are lapped. We put together all training subsequences of all lengths and all strides, and we train RGB and LDOF I3D networks separately with a learning rate $1e-4$ halved every 10 epochs.

IDT Features. On HMDB-51 and MPII Cooking Activities, we also report accuracy when our kernel is combined with dense trajectories [83] encoded by Fisher Vectors [84].

5.3 Sequence compatibility kernel.

In this section, we first present experiments evaluating the influence of parameters σ_2 and σ_3 of kernels G_{σ_2} and G_{σ_3} which control the degree of selectivity for the 3D body-joints and the temporal shift invariance, respectively. See Section 4.3 for a full definition of these parameters. Recall that kernels G_{σ_2} and G_{σ_3} are approximated via linearizations according to Eq. (1) and (3). The quality of these approximations and the size of our final tensor representations depend on the numbers Z_2 and Z_3 of pivots chosen. See Section 3.2, Figure 3 and notes[2],[3] for details on pivots. In our experiments, the pivots ζ are spaced uniformly within interval $[-1;1]$ and $[0;1]$ for kernels G_{σ_2} and G_{σ_3} respectively.

Figures 7a and 7b present the results of this experiment on the Florence3D-Action dataset. Figure 7a shows that the body-joint compatibility subkernel G_{σ_2} requires a choice of σ_2, which is not too strict as specific body-joints (*e.g.*, elbow) are expected to repeat across sequences in similar locations due to zero-centering w.r.t. hip. On the one hand, very small σ_2 leads to poor generalization. On the other hand, very large σ_2 allows big displacements of the corresponding body-joints between sequences which results in a poor discriminative power of this kernel. Furthermore, Figure 7a demonstrates that the range of σ_3 for the temporal subkernel for which we obtain very good performance is large. However, as σ_3 becomes very small or very large, extreme temporal selectivity or full temporal invariance, respectively, result in a loss of performance. For instance, $\sigma_3=4$ results in 91% accuracy only.

In Figure 7b, we show the performance of our SCK kernel with respect to the number of pivots used for linearization. For the body-joint compatibility subkernel G_{σ_2}, we see that $Z_2=5$ pivots are sufficient to obtain good performance of 92.98% accuracy. We have observed that this is consistent with the results on the validation set. Using more pivots, say $Z_2=20$, deteriorates the results slightly, suggesting overfitting. We make similar observations for the temporal subkernel G_{σ_3} which demonstrates a good performance for as few as $Z_3=2$ pivots. Such a small number of pivots suggests that linearizing 1D variables and generating higher-order co-occurrences, as described in Section 4.3, constitute on a simple, robust, and effective linearization strategy.

Furthermore, Figure 7c demonstrates the effectiveness of our slice-wise Eigenvalue Power Normalization described in Eq. (14). When $\gamma=1$, the EPN functionality is absent. This results in a drop of performance from 92.98% to 88.7% accuracy. This demonstrates that statistically unpredictable bursts of actions described by body-joints, such as long versus short *hand waving*, are indeed undesirable. It is clear that in such cases, EPN is very effective, as it deals with correlated bursts, *e.g.* co-occurring *hand wave* and associated with it elbow and neck motion. For more details regarding this concept, see [4]. For our further experiments, we choose $\sigma_2=0.6$, $\sigma_3=0.5$, $Z_2=5$, $Z_3=6$, and $\gamma=0.36$, as dictated by cross-validation.

5.4 Dynamics compatibility kernel.

Below, we evaluate the influence of parameters of the DCK kernel. Our experiments are based on the Florence3D-Action dataset. For ablations, we present results on the test set as results on the validation set match test results closely. As this kernel considers all spatio-temporal co-occurrences of body-joints, we firstly evaluate the impact of the joint subsets we select for generating DCK, as not all body-joints need to be used for capturing actions.

Figure 8a enumerates all body-joints that describe every 3D human skeleton on the Florence3D-Action dataset, whereas the table underneath lists the proposed body-joint subsets A-I which

	SCK	DCK		SCK+DCK
accuracy	92.98%	93.03%	92.77%	**95.23%**
size	26,565	9,450	16,920	43,485

	SCK ⊕	DCK ⊕	SCK ⊕ + DCK ⊕
accuracy	96.50%	96.41%	**97.45%**
size	60,900	37,800	98,700

Bag-of-Poses 82.00% [32]	Kendal Traj. 93.04% [85]
$SE(3)$ 90.88% [12]	Kernel+ResNet [86] 95.4%

TABLE 1: Evaluations of (*top*) SCK/DCK, (*middle*) our improved SCK ⊕ / DCK ⊕, (*bottom*) the state of the art on Florence3D-Action.

	SCK	DCK	SCK+DCK
accuracy	96.08%	97.5%	**98.2%**
size	40,480	16,920	57,400

	SCK ⊕	DCK ⊕	SCK ⊕ + DCK ⊕
accuracy	98.50%	98.12%	**99.2%**
size	81,200	67,680	148,880

3D joints. hist. 90.92% [31]	Kendal Traj. 97.39% [85]
$SE(3)$ 97.08% [12]	Second-order DA [86] 98.9%

TABLE 2: Evaluations of (*top*) SCK/DCK, (*middle*) our improved SCK ⊕ / DCK ⊕ and (*bottom*) the state of the art on UTKinect-Action.

we use for computations of DCK. In Figure 8b, we plot the performance of our DCK kernel for each subset. The plot shows that using two body-joints associated with the hands from Configuration-A in the DCK kernel construction, we attain 88.32% accuracy which highlights the informativeness of temporal dynamics.

Some body-joints may be noisy and thus detrimental to recognition, and should not be selected for experiments *e.g.*, Configuration-D, which includes six body-joints such as the knees, elbows and hands, yields 93.03%, which outperforms more complex configurations.

As Configuration-E includes eight body-joints such as the feet, knees, elbows and hands, we choose it for our further experiments as it represents a reasonable trade-off between performance and size of representations. This configuration scores 92.77% accuracy. We see that if we utilize all the body-joints according to Configuration-I, performance of 91.65% accuracy is still somewhat lower compared to 93.03% accuracy for Configuration-D highlighting the issue of noisy body-joints.

In Figure 7d, we show the accuracy on our DCK kernel when HOSVD factors underlying our non-symmetric tensors are equalized by varying the EPN parameter γ. For $\gamma = 1$, EPN is disabled, which leads to 90.49% accuracy only. For the optimal value of $\gamma = 0.85$, the accuracy rises to 92.77% which indicates the presence of the burstiness effect in temporal representations.

5.5 SCK and DCK vs. the state of the art.

Below, we compare the performance of our representations against the state of the art. Along with comparing SCK and DCK, we also explore the complementarity of these representations by combining them via the so-called late fusion, that is, a simple weighted concatenation of vectorized SCK and DCK.

On the Florence3D-Action dataset, we present our best results in Table 1. Note that the model parameters for the evaluation was selected by cross-validation. Linearizing a sequence compatibility kernel using these parameters resulted in a tensor representation

	SCK	DCK	SCK+DCK
acc., prot. [41]	90.72%	86.30%	**91.45%**
acc., prot. [33]	93.52%	91.71%	**93.96%**
size	40,480	16,920	57,400

	SCK ⊕	DCK ⊕	SCK ⊕ + DCK ⊕
acc., prot. [41]	97.50%	90.03%	**98.10%**
acc., prot. [33]	98.12%	94.28%	**98.62%**
size	81,200	67,680	148,880

accuracy, protocol [41]	accuracy, protocol [33]
Actionlets 88.20% [41]	$SE(3)$ 92.46% [12]
$SE(3)$ 89.48% [12]	Kendal Traj. 94.19% [85]
Kin. desc. 91.07% [87]	Ker-RP-RBF 96.9% [47]

TABLE 3: Results of (*top*) SCK/DCK, (*middle*) our improved SCK ⊕ / DCK ⊕ and (*bottom*) the state of the art on MSR-Action3D.

		cross-subject	cross-view
SCK ($r=2$)	on 3D body-joints	64.08%	65.24%
SCK (no EPN)	on 3D body-joints	65.37%	67.18%
SCK	on 3D body-joints	69.20%	70.55%
SCK ⊕	on 3D body-joints	**72.82%**	**74.10%**
SCK	on 3D body-joints +ST-GCN	82.61%	89.52%
SCK ⊕	on 3D body-joints +ST-GCN	**83.58%**	**90.84%**

	cross-subject	cross-view
Two-stream+AP (ResNet-50)	74.4%	83.3%
Two-stream+MP (ResNet-50)	65.8%	58.7%
SCK ⊕ on RGB frames (ResNet-152)	**90.11%**	**93.62%**
SCK ⊕ on 3D body-joints +RGB frames (ResNet-152)	**90.78%**	**94.15%**
SCK ⊕ on 3D body-joints +RGB frames+optical flow (ResNet-152)	**91.56%**	**94.75%**

	cross-subject	cross-view
Second-order DA [86] (ResNet-50)	75.35%	79.30%
Frames + CNN [88] (VGG-19)	75.73%	79.62%
Clips + CNN + MTLN [88] (VGG-19)	79.57%	84.83%
VA-LSTM [89]	79.4%	87.6%
ST-GCN [75]	81.5%	88.3%
DSP [90]	81.6%	88.7%
Multi-scale CNN [91] (ResNet-101)	84.6%	92.1%
Multi-scale CNN [91] (ResNet-152)	85.0%	92.3%
Deep Bilinear [92] (ResNet-101)	85.4%	90.7%

TABLE 4: Results on our SCK and the improved SCK ⊕ on (*top*) skeleton sequences and (*middle*) two-stream networks. We also indicate results on the baseline two-stream network with standard average pooling (*AP*) and maximum pooling (*MP*). We indicate backbones in parentheses. (*bottom*) The state of the art on NTU-RGBD.

of size $26,565$ dimensions[4], and produced an accuracy of 92.98% accuracy. As for DCK, our model used Configuration-E (described in Figure 8a) resulting in a representation of dimensionality $16,920$, and achieved a performance of 92%. However, somewhat better results were attained by Configuration-D, that is, 92.27% accuracy for size of $9,450$. Combining SCK and DCK in Configuration-E yields 95.23% accuracy, a 4.5% improvement over the state of the art on this dataset, as listed in Table 1, which highlights the complementary nature of SCK and DCK.

[4]This is the length of a vector per sequence after unfolding our tensor represent./removing duplicate coefficients from the symmetries in the tensor.

	SCK+ST-GCN	SCK ⊕+ST-GCN	ST-GCN
top-1	31.2%	**31.8%**	30.7%
top-5	53.7%	**54.9%**	52.8%

TABLE 5: SCK and SCK ⊕ combined with ST-GCN *vs*. ST-GCN [75] alone on Kinetics [36] skeletons extracted by OpenPose [5].

Action recognition results on the UTKinect-Action dataset are presented in Table 2. For our experiments on this dataset, we kept all the parameters the same as those used on the Florence3D dataset. SCK and DCK representations yielded 96.08% and 97.5% accuracy, respectively. Combining SCK and DCK yielded 98.2% accuracy outperforming marginally a more complex approach [12] based on the Lie group algebra, dynamic time warping and Fourier temporal pyramids.

In Table 3, we present our results on the MSR-Action3D dataset. Conforming to the prior literature, we use two evaluation protocols, that is, (i) the protocol described in actionlets [41], for which the authors utilize the entire dataset with its 20 classes during the training and evaluation, and (ii) approach of [33], for which the authors divide the data into three subsets and report the average in classification accuracy over these subsets. SCK yields the state-of-the-art accuracy of 90.72% and 93.52% for the two evaluation protocols, respectively. Combining SCK with DCK outperforms other approaches listed in the table and yields 91.45% and 93.96% accuracy for the two protocols, respectively.

5.6 SCK ⊕ and DCK ⊕ vs. the state of the art.

Our extended SCK ⊕ is trained with $3Z_2 = 15$, $Z_3 = Z_4 = 5$ and $Z_5 = 3$ while DCK ⊕ follows the same setting as DCK, except that we introduce quantity $Z_6 = 4$ which is the number of pivots encoding the subsequence position within the sequence, as dictated by Eq. (28). For the Florence3D-Action dataset, Table 1 shows that aggregating over subsequences across various scales results in SCK ⊕ outperforming SCK by ∼3.5%, DCK ⊕ outperforming DCK by ∼3.4% and the combined kernel SCK ⊕ + DCK ⊕ outperforming SCK+DCK by ∼2.2%. Table 2 shows the similar trend for the UTKinect-Action dataset, for which SCK ⊕ outperforms SCK by ∼2.4%, DCK ⊕ outperforms DCK by ∼0.6% and the combined kernel SCK ⊕ + DCK ⊕ outperforms SCK+DCK by ∼1.0%. Note that the results on UTKinect-Action should be considered as already saturated. Furthermore, Table 3 shows that on MSR-Action3D, SCK ⊕ outperforms SCK by ∼6.8%, DCK ⊕ outperforms DCK by ∼3.7% and the combined kernel SCK ⊕ + DCK ⊕ outperforms SCK+DCK by ∼7.5%.

Fine-grained Action Recognition. In what follows, we employ NTU-RGBD, a partially fine-grained dataset, and MPII Cooking Activities containing many fine-grained classes.

Our SCK ⊕ kernel is designed to capture specific subsequences of variable lengths. Kernels $G_{\sigma_2}, ..., G_{\sigma_5}$ from Section 4.5 capture higher-order statistics of joint locations in subsequences, the temporal alignment of pose snippets, the global alignment of subsequences, and the match of subsequence lengths. SCK ⊕ uses EPN in Eq. (15-18) which makes it act as a detector of spectral higher-order occurrences. Thus, SCK ⊕ addresses all hallmarks of modern fine-grained recognition systems: *it captures higher-order statistics* describing visual contents/objects and *discarding burstiness* [2] (co-occurrence detection).

Moreover, our SCK ⊕ kernel captures higher-order occurrences of features representing spatio-temporal evolution of skeletons (for 3D body-joints) and/or frame-based classifier scores (semantic information) by feeding them into kernels $G_{\sigma_2^{(1)}}, ..., G_{\sigma_2^{(Q)}}$ from Eq. (23) for Q modalities.

	-	+IDT	+sec-ord	+sec-ord +IDT
Two-stream+AP (VGG-19)	38.1%	-	-	-
Two-stream+AP (ResNet-152)	45.3%	-	-	-
Subsequences+AP (I3D)	52.7%	-	-	-
HOK [30] (VGG-16)	60.1%	-	69.1%	73.1%
SCK ⊕ (VGG-19)	70.1%	-	74.0%	76.1%
SCK ⊕ (ResNet-152)	71.4%	-	75.5%	77.4%
SCK ⊕ (I3D)	77.8%	**80.4%**	-	-

KRP-FS 70.0% [93] (VGG-19)	KRP-FS+IDT 76.1% [93] (VGG-19)
GRP 68.4% [8] (VGG-19)	GRP+IDT 75.5% [8] (VGG-19)

TABLE 6: Results (mAP%) for (*top*) our HOK [30] and improved SCK ⊕. We also indicate results on the baseline two-stream network with standard average pooling (*AP*). We indicate backbones in parentheses. (*bottom*) The state of the art on MPII Cooking Activities.

	sp1	*sp2*	*sp3*	mean acc.
Two-stream+AP (ResNet-152)	65.30%	62.20%	62.55%	63.35%
Two-stream+MP (ResNet-152)	61.38%	60.58%	60.06%	60.66%
SCK ⊕ (ResNet-152)	72.55%	70.85%	71.63%	71.67%
SCK ⊕ (ResNet-152)+IDT	74.20%	73.73%	73.40%	73.77%
SCK ⊕(r=2) (I3D)+IDT	85.61%	84.54%	85.25%	85.13%
SCK ⊕ (I3D)+IDT	86.31%	85.63%	86.41%	**86.11%**

DSP 72.4% [90] (ResNet-152)	ShuttleNet+MIF 71.08% [94]
DSP+IDT 74.3% [90] (ResNet-152)	I3D 80.2% [28]

TABLE 7: Evaluations of (*top*) our improved SCK ⊕. We also indicate results on baseline two-stream network with standard average pooling (*AP*) and maximum pooling (*MP*). We indicate backbones in parentheses. (*bottom*) The state of the art on HMDB-51.

Table 4 (top) shows that, our SCK ⊕ yields some ∼3.6% improvement over SCK and reaches 72.82% accuracy on the NTU-RGBD dataset in the cross- subject setting for the 3D body-joints as input. We expect that aggregating over subsequences can encode local fine-grained motion details essential for the good performance. Similar observations hold for the cross-view setting.

Table 4 (middle) shows that our SCK ⊕ attains 90.11% accuracy on the NTU-RGBD dataset in the cross- subject setting on the RGB frames (classifier scores) as input. With the 3D body-joints added, results increase to 90.78%. Lastly, adding optical flow as input to our SCK ⊕ yields 91.56%. This is ∼ 10.0% improvement over competing methods from Table 4 (bottom).

Table 6 shows that our SCK ⊕ yields some 1.4% mAP improvement over other state-of-the-art methods [8, 93] on the MPII Cooking Activities dataset. Further improvements are attained by combining SCK ⊕ with the second-order descriptor (*sec-ord*) [93] and the IDT representation, which yields 77.4% mAP. This compares favorably with other methods in the table. We also note that SCK ⊕ outperforms the HOK descriptor [30] which is a variant of SCK with a suboptimal linearization of an *fc* layer. Finally, applying SCK ⊕ over I3D-based subsequences yields state-of-the-art 80.4% mAP (we comment on the reasons below).

Video Classification. Table 4 confirms that the classifier scores extracted from CNNs rather than mere 3D body-joints are a more informative input for SCK ⊕. Thus, we perform additional evaluations on the HMDB-51 dataset. Table 7 (top) shows that

SCK$\oplus$ and SCK$\oplus$+IDT, trained with the two-stream ResNet-152 backbone, score 71.67 and 73.77% accuracy which is on a par with other best methods listed in Table 7 (bottom). Furthermore, applying the I3D backbone on SCK$\oplus$ yields state-of-the-art 86.11% accuracy. We believe that training I3D on subsequences of various lengths and strides, as detailed in Section 5.2 (bottom), is a more discriminative strategy than average-pooling of frame-wise features in standard two-stream networks. As SCK$\oplus$ is designed to combine subsequences of various lengths and strides rather than sequences, it captures informative higher-order occurrences of multiple complementary features, and also preserves a degree of individual statistics by factoring out one variable at a time *e.g.*, see the discussion in Figure 3.

Kinetics-400. Table 5 shows that our SCK and SCK$\oplus$ are complementary to powerful networks such as ST-GCN [75]. We work with Kinetics skeletons extracted with [5] and compare our method to the baseline ST-GCN [75]. We use the standard training/evaluation protocol (but we use skeletons rather than RGB or optical flow frames). As SCK and SCK$\oplus$ are shallow representations based on higher-order aggregation, it is unrealistic to expect them to outperform deep networks. However, SCK and SCK$\oplus$ capture very different statistics compared to deep networks, being highly complementary. Table 5 shows that we attain 1.1% and 2.1% gain over ST-GCN alone by concatenating both representations.

Signature Lengths. Section 5.6 indicates the number of pivots for SCK$\oplus$ on NTU (skeleton-based experiments) to amount to $d=3Z_2+Z_3+Z_4+Z_5=15+5+5+3=28$. The unique number of coefficients in the super-symmetric tensor of order r follows the formula $\binom{d+r-1}{r}$ discussed just below Eq. (11). As we obtain a tensor per joint, and we concatenate unique parts of tensors $j=1,...,J$, we have $\binom{d+r-1}{r}\cdot J$ coefficients in total in our representation. For SCK$\oplus$ on NTU with $J=25$ body joints, we obtain $4060\times 25=101500$ coefficients for SCK$\oplus$. For SCK and SCK ($r=2$) on NTU, we set $d=3Z_2+Z_3=24+5=29$ and obtain 112375 and 10875 coefficients, respectively. For Kinetics skeletons with $J=18$ body joints, OpenPose returns only two Cartesian coordinates, so we set $d=2Z_2+Z_3+Z_4+Z_5=20+5+5+3=33$ which yields $4545\times 18=117810$ coefficients.

For SCK$\oplus$ (NTU) on (i) RGB frames and (ii) RGB frames+optical flow, we obtain $d=Z_2+Z_3+Z_4+Z_5=60+5+5+3=73$ and $d=2Z_2+Z_3+Z_4+Z_5=73$ (for the latter case, we reduce the size of vector of classifier scores 2$\times$ by the PCA). As we do not use any body joints here, we obtain 67525 coefficients. When we concatenate these representations with the skeleton-based one, we obtain $67525+101500=169025$ coefficients per video.

On SCK$\oplus$ given MPII and HMDB-51 datasets, we obtain 171700 and 125580 coefficients after reducing the size of vectors of RGB frame-wise and optical flow classifier scores from 2×64 to 100 and from 2×51 to 90, respectively.

Parameters in SCK$\oplus$. The main parameters shared between SCK and SCK$\oplus$ are evaluated in Figures 7 and 8. The parameters for SCK$\oplus$ that we start with are indicated in Section 4.5 (bottom). To select the best parameters, we cross-validate one parameter at a time while keeping the rest fixed. For NTU, we aggregated over subsequence lengths (using the Matlab notation) of $14:1:110$, $14:2:110$, $14:4:110$ and $14:6:110$, and we obtained 73.10%, 72.82%, 72.41% and 71.65% accuracy, respectively. For subsequence lengths $30:2:110$ and $30:2:80$, we obtained 72.54% and 72.12% accuracy. These evaluations show that SCK$\oplus$ is not overly sensitive to its parameters. For smaller skeleton-based datasets, we aggregate subsequences in range $6:2:24$, whereas on HMDB-51 we use $6:8:62$, and for MPII we use $48:16:96$.

Processing Time. For SCK/DCK, processing a sequence with unoptimized MATLAB code on a single i5 core takes 0.2s and 1.2s, respectively. For SCK$\oplus$ / DCK$\oplus$, processing one sequence takes 0.5s and 3.0s. Training on full MSR-Action3D with the SCK+DCK takes about 13 min, whereas with the SCK$\oplus$ + DCK$\oplus$, it takes about 35 min. In comparison, extracting $SE(3)$ features [12] takes 5.3s per sequence, processing on the full MSR-Action3D dataset takes $\sim$50 min., whereas with post-processing (time warping and Fourier pyramids) it takes about 72 min. Thus, SCK+DCK is $\sim 5.4\times$ faster while SCK$\oplus$ + DCK$\oplus$ is $\sim 2\times$ faster. Section C contains the computational complexity analysis.

6 Conclusions

We have presented two kernel-based tensor representations, namely the sequence compatibility kernel (SCK) and dynamics compatibility kernel (DCK). SCK captures the higher-order correlations between 3D coordinates of the body-joints and their temporal variations. As SCK factors out the temporal variable, expensive Fourier temporal pyramid matching/dynamic time warping are not needed. Further, our DCK kernel captures the action dynamics by modeling the spatio-temporal co-occurrences of the body-joints.

Additionally, we have presented a highly effective extension of SCK, termed SCK$\oplus$, which aggregates over subsequences of multiple lengths, focusing on actions within subsequences. We have demonstrated that SCK$\oplus$ can aggregate over 3D body-joints and/or frame-wise classifier scores from CNNs to capture higher-order statistics between various features extracted from body-skeletons, classifier scores, and temporal positions.

Section D shows that (Tensor) Eigenvalue Power Normalization indeed acts as a spectrum-based metric with $\binom{Z_*}{r}$ subspace-based detectors of higher-order occurrence of datapoints of dim. Z_*, more specifically, detectors that capture asymmetry of projections into 'positive' and 'negative' parts of each subspace. As $\binom{Z_*}{3}\gg\binom{Z_*}{2}$, third-order tensors capture more dependencies than autocorrelation matrices, improving fine-grained systems.

Remaining Details/Derivations

A. Linearizing Dynamics Compatibility Kernel

In what follows, we derive the linearization of DCK. Let us recall that $G_\sigma(\mathbf{u}-\bar{\mathbf{u}})=\exp(-\|\mathbf{u}-\bar{\mathbf{u}}\|_2^2/2\sigma^2)$, $G'_\sigma(\boldsymbol{\alpha},\boldsymbol{\beta})=G_\sigma(\boldsymbol{\alpha})G_\sigma(\boldsymbol{\beta})$ and $G_\sigma(\mathbf{i}-\boldsymbol{j})=\delta(\mathbf{i}-\boldsymbol{j})$ if $\sigma\to 0$, therefore $\delta(\mathbf{0})=1$ and $\delta(\mathbf{u})=0$ if $\mathbf{u}\neq\mathbf{0}$. Moreover, $\Lambda=J^2$ is a normalization constant and $\mathcal{J}=\mathcal{I}_J\times\mathcal{I}_N$. We recall that kernel $G_{\sigma'_2}(\mathbf{x}-\mathbf{y})\approx\phi(\mathbf{x})^T\phi(\mathbf{y})$ while $G_{\sigma'_3}(\frac{s-t}{N})\approx\mathbf{z}(s/N)^T\mathbf{z}(t/N)$. Thus, we obtain Eq. (29) which expresses $K_D(\Pi_A,\Pi_B)$ as a sum over dot-products on third-order non-symmetric tensors. We introduce operator $\mathcal{G}$ into Eq. (29) to amend the dot-product with a distance which handles burstiness. We obtain a modified kernel in Eq. (30) based on which the following notation is introduced:

$$\boldsymbol{\mathcal{V}}_{ii'}=\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{X}}_{ii'}),\text{ where}\tag{31}$$

$$\boldsymbol{\mathcal{X}}_{ii'}=\frac{1}{\sqrt{\Lambda}}\sum_{\substack{s,s'\in\mathcal{I}_N:\\ s'\neq s}}G_{\sigma'_4}(s-s')\left(\boldsymbol{\phi}(\mathbf{x}_{is}-\mathbf{x}_{i's'})\cdot\mathbf{z}\big(\frac{s}{N}\big)^T\right)\uparrow\otimes\mathbf{z}\big(\frac{s'}{N}\big),$$

$$
\begin{aligned}
K_D(\Pi_A,\Pi_B) &= \frac{1}{\Lambda}\sum_{\substack{(i,s)\in\mathcal{J},\\(i',s')\in\mathcal{J},\\ i'\neq i, s'\neq s}}\ \sum_{\substack{(j,t)\in\mathcal{J},\\(j',t')\in\mathcal{J},\\ j'\neq j, t'\neq t}} G'_{\sigma'_1}(i-j,i'-j')\,G_{\sigma'_2}\left((\mathbf{x}_{is}-\mathbf{x}_{i's'})-(\mathbf{y}_{jt}-\mathbf{y}_{j't'})\right)G'_{\sigma'_3}(\frac{s-t}{N},\frac{s'-t'}{N})\cdot G'_{\sigma'_4}(s-s',t-t') \\
&= \frac{1}{\Lambda}\sum_{\substack{i,i'\in\mathcal{I}_J:\\ i'\neq i}}\ \sum_{\substack{s,s'\in\mathcal{I}_N:\\ s'\neq s}}\ \sum_{\substack{t,t':\\ t'\neq t}} G_{\sigma'_2}\left((\mathbf{x}_{is}-\mathbf{x}_{i's'})-(\mathbf{y}_{jt}-\mathbf{y}_{j't'})\right)G_{\sigma'_3}(\frac{s-t}{N})G_{\sigma'_3}(\frac{s'-t'}{N})\cdot G_{\sigma'_4}(s-s')\,G_{\sigma'_4}(t-t')\Big|_{\substack{j=i\\ j'=i'}} \\
&\approx \frac{1}{\Lambda}\sum_{\substack{i,i'\in\mathcal{I}_J:\\ i'\neq i}}\ \sum_{\substack{s,s'\in\mathcal{I}_N:\\ s'\neq s}}\ \sum_{\substack{t,t'\in\mathcal{I}_N:\\ t'\neq t}} \boldsymbol{\phi}\left(\mathbf{x}_{is}-\mathbf{x}_{i's'}\right)^T\boldsymbol{\phi}\left(\mathbf{y}_{it}-\mathbf{y}_{i't'}\right)\cdot\mathbf{z}(\frac{s}{N})^T\mathbf{z}(\frac{t}{N})\cdot\mathbf{z}(\frac{s'}{N})^T\mathbf{z}(\frac{t'}{N})\cdot G_{\sigma'_4}(s-s')\,G_{\sigma'_4}(t-t') \\
&= \frac{1}{\Lambda}\sum_{\substack{i,i'\in\mathcal{I}_J:\\ i'\neq i}}\ \sum_{\substack{s,s'\in\mathcal{I}_N:\\ s'\neq s}}\ \sum_{\substack{t,t'\in\mathcal{I}_N:\\ t'\neq t}} \left\langle G_{\sigma'_4}(s-s')\left(\boldsymbol{\phi}(\mathbf{x}_{is}-\mathbf{x}_{i's'})\cdot\mathbf{z}(\frac{s}{N})^T\right)\uparrow\otimes\mathbf{z}(\frac{s'}{N}),\, G_{\sigma'_4}(t-t')\left(\boldsymbol{\phi}(\mathbf{y}_{it}-\mathbf{y}_{i't'})\cdot\mathbf{z}(\frac{t}{N})^T\right)\uparrow\otimes\mathbf{z}(\frac{t'}{N})\right\rangle \\
&= \sum_{\substack{i,i'\in\mathcal{I}_J:\\ i'\neq i}}\left\langle \frac{1}{\sqrt{\Lambda}}\sum_{\substack{s,s'\in\mathcal{I}_N:\\ s'\neq s}} G_{\sigma'_4}(s-s')\left(\boldsymbol{\phi}(\mathbf{x}_{is}-\mathbf{x}_{i's'})\cdot\mathbf{z}(\frac{s}{N})^T\right)\uparrow\otimes\mathbf{z}(\frac{s'}{N}),\, \frac{1}{\sqrt{\Lambda}}\sum_{\substack{t,t'\in\mathcal{I}_N:\\ t'\neq t}} G_{\sigma'_4}(t-t')\left(\boldsymbol{\phi}(\mathbf{y}_{it}-\mathbf{y}_{i't'})\cdot\mathbf{z}(\frac{t}{N})^T\right)\uparrow\otimes\mathbf{z}(\frac{t'}{N})\right\rangle \qquad (29)
\end{aligned}
$$

$$
K^*_D(\Pi_A,\Pi_B) = \sum_{\substack{i,i'\in\mathcal{I}_J:\\ i'\neq i}}\left\langle \boldsymbol{\mathcal{G}}\left(\frac{1}{\sqrt{\Lambda}}\sum_{\substack{s,s'\in\mathcal{I}_N:\\ s'\neq s}} G_{\sigma'_4}(s-s')\left(\boldsymbol{\phi}(\mathbf{x}_{is}-\mathbf{x}_{i's'})\cdot\mathbf{z}(\frac{s}{N})^T\right)\uparrow\otimes\mathbf{z}(\frac{s'}{N})\right), \boldsymbol{\mathcal{G}}\left(\frac{1}{\sqrt{\Lambda}}\sum_{\substack{t,t'\in\mathcal{I}_N:\\ t'\neq t}} G_{\sigma'_4}(t-t')\left(\boldsymbol{\phi}(\mathbf{y}_{it}-\mathbf{y}_{i't'})\cdot\mathbf{z}(\frac{t}{N})^T\right)\uparrow\otimes\mathbf{z}(\frac{t'}{N})\right)\right\rangle \qquad (30)
$$

and the summation over the pairs of body-joints in Eq. (30) is replaced by the concatenation along the fourth mode to obtain representations $\left[\boldsymbol{\mathcal{V}}_{ii'}\right]^{\oplus_4}_{i>i':\, i,i'\in\mathcal{I}_J}$ and $\left[\bar{\boldsymbol{\mathcal{V}}}_{ii'}\right]^{\oplus_4}_{i>i':\, i,i'\in\mathcal{I}_J}$ for Π_A and Π_B. Thus, K^*_D becomes:

$$
K^*_D(\Pi_A,\Pi_B) = \left\langle \sqrt{2}\left[\boldsymbol{\mathcal{V}}_{ii'}\right]^{\oplus_4}_{i>i':\, i,i'\in\mathcal{I}_J}, \sqrt{2}\left[\bar{\boldsymbol{\mathcal{V}}}_{ii'}\right]^{\oplus_4}_{i>i':\, i,i'\in\mathcal{I}_J}\right\rangle \qquad (32)
$$

As Eq. (32) suggests, we avoid repeating the same evaluations in our representations: we stack only unique pairs of body-joints $i>i'$. Moreover, we ensure we run computations temporally only for $s>s'$. In practice, we have to evaluate only $\binom{JN}{2}$ unique spatio-temporal pairs in Eq. (32) rather than naive J^2N^2 per sequence. The final representation is of $Z'_2\cdot\binom{JZ'_3}{2}$ size, where Z'_2 and Z'_3 are the numbers of pivots for approximation of $G_{\sigma'_2}$ and $G_{\sigma'_3}$.

We assume that all sequences have N frames for simplification of presentation. Our formulations are equally applicable to sequences of arbitrary lengths *e.g.*, M and N. Thus, we apply in practice $G'_{\sigma'_3}(\frac{s}{M}-\frac{t}{N},\frac{s'}{M}-\frac{t'}{N})$ and $\Lambda=J^2MN$ in Eq. (29).

Moreover, a displacement between any pair of joints $\mathbf{x},\mathbf{y}\in\mathbb{R}^3$ lies within the Cartesian coordinate system, thus $\mathbf{x}-\mathbf{y}\in\mathbb{R}^3$. In practice, in place of generic $G_{\sigma'_2}$, we use the sum kernel $G'_{\sigma'_2}(\mathbf{x}-\mathbf{y})=G_{\sigma'_2}(x_1-y_1)+G_{\sigma'_2}(x_2-y_2)+G_{\sigma'_2}(x_3-y_3)$ so the kernel $G'_{\sigma'_2}(\mathbf{x}-\mathbf{y})\approx[\phi(x_1);\phi(x_2);\phi(x_3)]^T[\phi(y_1);\phi(y_2);\phi(y_3)]$. However, for the simplicity of notation, we refer to it in our formulations by its generic form $G_{\sigma'_2}(\mathbf{x}-\mathbf{y})\approx\phi(\mathbf{x})^T\phi(\mathbf{y})$, as we can simply define $\phi(\mathbf{x})=[\phi(x_1);\phi(x_2);\phi(x_3)]$.

B. Positive Definiteness of SCK and DCK

SCK/DCK are sums over products of RBF subkernels. According to [95], sums, products and linear combinations (for non-negative weights) of positive definite kernels yield positive definite kernels. Moreover, subkernel $G_{\sigma'_2}\left((\mathbf{x}_{is}-\mathbf{x}_{i's'})-(\mathbf{y}_{jt}-\mathbf{y}_{j't'})\right)$ employed by DCK in Eq. (29) (top) can be rewritten as:

$$
\begin{aligned}
&G_{\sigma'_2}\left(\mathbf{z}_{isi's'}-\mathbf{z}'_{jtj't'}\right), \qquad (33)\\
&\text{where } \mathbf{z}_{isi's'}=\mathbf{x}_{is}-\mathbf{x}_{i's'} \text{ and } \mathbf{z}'_{jtj't'}=\mathbf{y}_{jt}-\mathbf{y}_{j't'}.
\end{aligned}
$$

The RBF kernel $G_{\sigma'_2}$ is positive definite (PD) by definition and the mappings from $\mathbf{x}_{is}$ and $\mathbf{x}_{i's'}$ to $\mathbf{z}_{isi's'}$ and from $\mathbf{y}_{jt}$ and $\mathbf{y}_{j't'}$ to $\mathbf{z}'_{jtj't'}$, respectively, are unique. Thus, the entire kernel is PD.

Whitening on SCK results in a positive (semi)definite (PSD) kernel as we employ the Power-Euclidean kernel *e.g.*, if $\mathbf{X}$ is PD then $\mathbf{X}^\gamma$ stays also PD for $0<\gamma\leq 1$ because $\mathbf{X}^\gamma=\boldsymbol{U}\boldsymbol{\lambda}^\gamma\boldsymbol{V}$ and element-wise rising of eigenvalues to the power of γ gives us $\mathrm{daig}(\boldsymbol{\lambda})^\gamma\geq 0$. Thus, the sum over dot-products of positive (semi)definite matrices raised to the power of γ stays PSD/PD.

C. Computational Complexity

Non-linearized SCK with ker. SVM have complexity $\mathcal{O}(JN^2T^\rho)$ given J body joints, N frames per sequence, T sequences, and $2<\rho<3$ which concerns complexity of kernel SVM. Linearized SCK with linear SVM takes $\mathcal{O}(JNTZ^r_*)$ for total of Z_* pivots and tensor of order $r=3$. Note that $N^2T^\rho\gg NTZ^r_*$. For $N=50$ and $Z_*=20$, this is $3.5\times$ (or $32\times$) faster than the exact kernel for $T=557$ (or $T=5000$) used in our experiments.

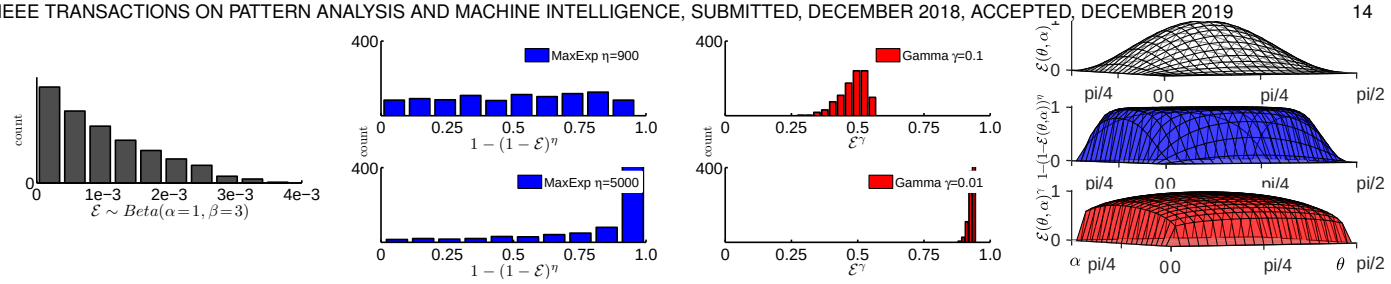
Non-linearized DCK+kernel SVM enjoys $\mathcal{O}(J^2N^4T^\rho)$ complexity. Linearized DCK+SVM enjoys $\mathcal{O}(J^2N^2TZ^3)$ for Z pivots per kernel, *e.g.* $Z=Z_2=Z_3$ given $G_{\sigma'_2}$ and $G_{\sigma'_3}$. As $N^4T^\rho\gg N^2TZ^3$, the linearization is $11000\times$ faster than the exact kernel, for say $Z=5$. Slice-wise EPN applied to SCK has negligible cost $\mathcal{O}(JTZ^{\omega+1}_*)$, where $2<\omega<2.376$ concerns complexity of eigenvalue decomposition applied per tensor slice.

Note that EPN incurs negligible cost (see [96] for details). EPN applied to DCK utilizes HOSVD and results in complexity $\mathcal{O}(J^2TZ^4)$. As HOSVD is performed by truncated SVD on matrices obtained from unfolding $\boldsymbol{\mathcal{V}}_{ii'}\in\mathbb{R}^{Z\times Z\times Z}$ along a chosen mode, $\mathcal{O}(Z^4)$ represents the complexity of truncated SVD on matrices $\boldsymbol{V}_{ii'}\in\mathbb{R}^{Z\times Z^2}$ which have rank less than or equal Z.

Linearized SCK$\oplus$ with linear SVM also takes $\mathcal{O}(JNTZ^r_*)$ for a total of Z_*. However, $Z_*=3Z_2+Z_3+Z_4+Z_5$ thus $Z_*=28$. The linearized DCK$\oplus$ takes $\mathcal{O}(J^2N^2TZ^3Z_6)$ where $Z_6=4$ in our experiments. EPN applied to SCK$\oplus$ and DCK$\oplus$ results in complexity $\mathcal{O}(JTZ^{2(r-1)}_*)$ and $\mathcal{O}(J^2TZ^4Z_6)$.

D. What is (Tensor) Eigenvalue Power Normalization?

Below, we show that EPN in fact retrieves factors which quantify whether there is at least one datapoint $\boldsymbol{\phi}(\mathbf{x}_n)$ from $n\in\mathcal{I}_N$ projected into each subspace spanned by r-tuples of eigenvectors from matrices $\boldsymbol{A}_1=\boldsymbol{A}_2=...=\boldsymbol{A}_r$. For brevity, assume order



(a) Initial spectral dist. **(b)** Pushforward (MaxExp) **(c)** Pushforward (Gamma) **(d)** Spectral detectors

Fig. 9: The intuitive principle of the EPN. Given a discrete spectrum following a Beta distribution in Fig. 9a, the pushforward measures by MaxExp and Gamma in Fig. 9b and 9c are very similar for large η (and small γ). Note that both EPN functions in bottom plots whiten the spectrum (map most values to be close to 1) thus removing burstiness. Fig. 9d illustrates the principle of detecting higher-order occurrence(s) in one of $\binom{Z_*}{r}$ subspaces represented by $\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}$ (we write $\mathcal{E}$ for simplicity). Fig. 9d (*top*) No EPN: $\mathcal{E}(\theta,\alpha)$, (*middle*) MaxExp: $1-(1-\mathcal{E}(\theta,\alpha))^{\eta}$ and (*bottom*) Gamma: $\mathcal{E}(\theta,\alpha)^{\gamma}$. Note how MaxExp/Gamma reach high detection values close to borders. Refer Section A for def. of $\mathcal{E}(\theta,\alpha)$.

$r=3$, a super-symmetric case, and a 3-tuple of eigenvectors $\mathbf{u}$, $\mathbf{v}$, and $\boldsymbol{w}$ from $\boldsymbol{A}$. Note that $\mathbf{u}\perp\mathbf{v}$, $\mathbf{v}\perp\boldsymbol{w}$ and $\mathbf{u}\perp\boldsymbol{w}$. Moreover, note that if we have Z_* unique eigenvectors, we can enumerate $\binom{Z_*}{r}$ r-tuples and thus $\binom{Z_*}{r}$ subspaces $\mathbb{R}^r\subset\mathbb{R}^{Z_*}$. For brevity, let $||\boldsymbol{\phi}(\mathbf{x})||_2=1$ and $\boldsymbol{\phi}(\mathbf{x})\geq 0$. Also, we write $\boldsymbol{\phi}_n$ instead of $\boldsymbol{\phi}(\mathbf{x})$ for $n\in\mathcal{I}_N$. Next, let us write our super-symmetric tensor as:

$$\boldsymbol{\mathcal{X}}=\frac{1}{N}\sum_{n\in\mathcal{I}_N}\uparrow\otimes_r\boldsymbol{\phi}_n, \tag{34}$$

and the 'diagonalization' of $\boldsymbol{\mathcal{X}}$ w.r.t. by eigenvec. $\mathbf{u}$, $\mathbf{v}$, and $\boldsymbol{w}$ as:

$$\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}=((\boldsymbol{\mathcal{X}}\otimes_1\mathbf{u})\otimes_1\mathbf{v})\otimes_3\boldsymbol{w}, \tag{35}$$

where $\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}$ is a coefficient from the core tensor $\boldsymbol{\mathcal{E}}$ for eigenvectors $\mathbf{u}$, $\mathbf{v}$, and $\boldsymbol{w}$. Now, we combine Eq. 34 and 35 and obtain:

$$\begin{aligned}\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}&=\left(\left(\left(\frac{1}{N}\sum_{n\in\mathcal{I}_N}\uparrow\otimes_3\boldsymbol{\phi}_n\right)\otimes_1\mathbf{u}\right)\otimes_2\mathbf{v}\right)\otimes_3\boldsymbol{w}\\&=\frac{1}{N}\sum_{n\in\mathcal{I}_N}\langle\boldsymbol{\phi}_n,\mathbf{u}\rangle\langle\boldsymbol{\phi}_n,\mathbf{v}\rangle\langle\boldsymbol{\phi}_n,\boldsymbol{w}\rangle\end{aligned} \tag{36}$$

We assume $\boldsymbol{\phi}_n$ is projected into subspace spanned by $\mathbf{u}$, $\mathbf{v}$ and $\boldsymbol{w}$ when $\psi'_n=\langle\boldsymbol{\phi}_n,\mathbf{u}\rangle\langle\boldsymbol{\phi}_n,\mathbf{v}\rangle\langle\boldsymbol{\phi}_n,\boldsymbol{w}\rangle$ is maximized. As our $\mathbf{u}$, $\mathbf{v}$, and $\boldsymbol{w}$ are orthogonal w.r.t. each other and $||\boldsymbol{\phi}_n||_2=1$, simple Lagrange eq. $\mathcal{L}=\Pi_{i=1}^{r}\boldsymbol{e}_i^T\boldsymbol{\phi}_n+\lambda(||\boldsymbol{\phi}_n||_2^2-1)$ yields maximum of $\kappa=(1/\sqrt{r})^r$ at $\boldsymbol{\phi}_n=[(1/\sqrt{r}),...,(1/\sqrt{r})]^T$. For each $n\in\mathcal{I}_N$, we store $\psi_n=\psi'_n/\kappa$ in vector $\boldsymbol{\psi}$.

Assume that $\boldsymbol{\psi}\in\{0,1\}^N$ stores N outcomes of drawing from Bernoulli distribution under the i.i.d. assumption for which the probability p of an event $(\psi_n=1)$ and $1-p$ for $(\psi_n=0)$ is estimated as an expected value, $p=\text{Avg}_n\,\psi_n$ (even if $0\leq\boldsymbol{\psi}\leq 1$ in reality). Then the probability of at least one projection event $(\psi_n=1)$ into the subspace spanned by r-tuples in N trials becomes:

$$\hat{\mathcal{E}}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}=1-(1-p)^N=1-\left(1-\frac{\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}}{\kappa}\right)^N\approx\left(\frac{\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}}{\kappa}\right)^{\gamma}. \tag{37}$$

Thus, each of $\binom{Z_*}{r}$ subspaces spanned by r-tuples acts as a detector of projections into this subspace. The middle part of Eq. (37) (so-called MaxExp pooling) and its connection to the right-hand part of Eq. (37) (so-called Gamma) are detailed in [2]. In fact, our $\boldsymbol{\psi}$ can be negative so extending Eq. (37) to $\text{Sgn}(\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}})\left(1-(1-\frac{|\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}|}{\kappa})^{N+\eta}\right)$ makes our model a detector of asymmetry between projections into 'positive' and 'negative' parts of each subspace, and η compensates for non-binary $\boldsymbol{\psi}$.

Figure 9 illustrates that MaxExp and Gamma are in fact very similar. Figure 9a shows an initial Beta distribution of spectrum. Figures 9b and 9c (bottom) show that for sufficiently large parameters η and γ, both MaxExp and Gamma shift most of the distribution values to be approximately equal 1. Figure 9c illustrates the effect of EPN on eigenvalue $\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}$ (denoted as $\mathcal{E}$ for simplicity) representing a single subspace spanned by eigenvectors $\mathbf{u}$, $\mathbf{v}$, $\boldsymbol{w}$ such that $\mathbf{u}\perp\mathbf{v}$, $\mathbf{v}\perp\boldsymbol{w}$ and $\mathbf{u}\perp\boldsymbol{w}$. As a single projection into the subspace is defined as $\psi_n=\langle\boldsymbol{\phi}_n,\mathbf{u}\rangle\langle\boldsymbol{\phi}_n,\mathbf{v}\rangle\langle\boldsymbol{\phi}_n,\boldsymbol{w}\rangle/\kappa$, we note this is the product of three projections of $\boldsymbol{\phi}_n$ onto $\mathbf{u}$, $\mathbf{v}$, $\boldsymbol{w}$, respectively, measured by the cosine (dot-product). Thus, we parametrize such a projection by the spherical coordinates, that is:

$$\begin{aligned}&\pi_{\mathbf{u}}(\theta,\alpha)=\cos(\theta)\cdot\sin(\alpha),\ \pi_{\mathbf{v}}(\theta,\alpha)=\sin(\theta)\cdot\sin(\alpha),\\&\pi_{\mathbf{u}}(\alpha)=\cos(\alpha),\end{aligned} \tag{38}$$

where the azimuthal coordinate θ runs from 0 to 2π and the polar coordinate α runs from 0 to π. We rewrite the projection as:

$$\pi_{\mathbf{u},\mathbf{v},\boldsymbol{w}}(\theta,\alpha)=\pi_{\mathbf{u}}(\theta,\alpha)\cdot\pi_{\mathbf{v}}(\theta,\alpha)\cdot\pi_{\boldsymbol{w}}(\alpha)/\kappa. \tag{39}$$

We note that $\pi_{\mathbf{u},\mathbf{v},\boldsymbol{w}}(\theta,\alpha)$ and ψ_n are isomorphic as $||\boldsymbol{\phi}_n||_2=1$, thus it suffices to note $\mathcal{E}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}\sim\pi_{\mathbf{u},\mathbf{v},\boldsymbol{w}}(\theta,\alpha)$ and show the EPN pushforward output of $\mathcal{E}$ to understand how EPN behaves around the boundaries of the spanning vectors $\mathbf{u}$, $\mathbf{v}$, $\boldsymbol{w}$. Figure 9d (top) shows that $\mathcal{E}$ by itself has a weak response in the proximity of the spanning vectors $\mathbf{u}$, $\mathbf{v}$, $\boldsymbol{w}$. However, MaxExp and Gamma in Figures 9d middle and bottom manage to boost projections in the proximity of the spanning vectors in the similar manner to each other, both behaving like spectral detectors.

To conclude, let us consider the dot-product between Power Normalized tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ computed according to Eq. (15-17). Then:

$$\begin{aligned}\left\langle\hat{\boldsymbol{\mathcal{V}}}(\boldsymbol{\mathcal{X}}),\hat{\boldsymbol{\mathcal{V}}}(\boldsymbol{\mathcal{Y}})\right\rangle&=\left\langle\sum_{\substack{\mathbf{u}\in\boldsymbol{U}(\boldsymbol{\mathcal{X}})\\\mathbf{v}\in\boldsymbol{V}(\boldsymbol{\mathcal{X}})\\\boldsymbol{w}\in\mathbf{W}(\boldsymbol{\mathcal{X}})}}\hat{\mathcal{E}}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}\mathbf{u}\mathbf{v}^T\uparrow\otimes\boldsymbol{w},\sum_{\substack{\mathbf{u}'\in\boldsymbol{U}(\boldsymbol{\mathcal{Y}})\\\mathbf{v}'\in\boldsymbol{V}(\boldsymbol{\mathcal{Y}})\\\boldsymbol{w}'\in\mathbf{W}(\boldsymbol{\mathcal{Y}})}}\hat{\mathcal{E}}'_{\mathbf{u}',\mathbf{v}',\boldsymbol{w}'}\mathbf{u}'\mathbf{v}'^T\uparrow\otimes\boldsymbol{w}'\right\rangle\\&=\sum_{\substack{\mathbf{u}\in\boldsymbol{U}(\boldsymbol{\mathcal{X}})\\\mathbf{v}\in\boldsymbol{V}(\boldsymbol{\mathcal{X}})\\\boldsymbol{w}\in\mathbf{W}(\boldsymbol{\mathcal{X}})}}\sum_{\substack{\mathbf{u}'\in\boldsymbol{U}(\boldsymbol{\mathcal{Y}})\\\mathbf{v}'\in\boldsymbol{V}(\boldsymbol{\mathcal{Y}})\\\boldsymbol{w}'\in\mathbf{W}(\boldsymbol{\mathcal{Y}})}}\hat{\mathcal{E}}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}\hat{\mathcal{E}}'_{\mathbf{u}',\mathbf{v}',\boldsymbol{w}'}\langle\mathbf{u},\mathbf{u}'\rangle\langle\mathbf{v},\mathbf{v}'\rangle\langle\boldsymbol{w},\boldsymbol{w}'\rangle.\end{aligned} \tag{40}$$

Eq. (40) shows that all subspaces of $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ spanned by r-tuples (3-tuples in this example) are compared against each other for alignment by the cosine distance. When two subspaces $[\mathbf{u}\,\mathbf{v}\,\boldsymbol{w}]^T$ and $[\mathbf{u}'\,\mathbf{v}'\,\boldsymbol{w}']^T$ are aligned, for a strong similarity between these subspaces, a detection of at least one $\boldsymbol{\phi}_n$ and $\boldsymbol{\phi}'_n$ evidenced by $\hat{\mathcal{E}}_{\mathbf{u},\mathbf{v},\boldsymbol{w}}$ and $\hat{\mathcal{E}}'_{\mathbf{u}',\mathbf{v}',\boldsymbol{w}'}$ is also needed. We term Eq. (40) together with Eq. (15-17) as Tensor Power Euclidean dot-product which has the associated Tensor Power Euclidean metric $||\boldsymbol{\mathcal{X}}-\boldsymbol{\mathcal{Y}}||_{\mathcal{T}}=||\hat{\boldsymbol{\mathcal{V}}}(\boldsymbol{\mathcal{X}})-\hat{\boldsymbol{\mathcal{V}}}(\boldsymbol{\mathcal{Y}})||_F$.

REFERENCES

[1] T.-Y. Lin and S. Maji, "Improved Bilinear Pooling with CNNs," *BMVC*, 2017. 1, 2, 5

[2] P. Koniusz, H. Zhang, and F. Porikli, "A deeper look at power normalizations," *CVPR*, pp. 5774–5783, 2018. 1, 2, 5, 11, 14

[3] P. Koniusz, F. Yan, P. Gosselin, and K. Mikolajczyk, "Higher-order occurrence pooling on mid- and low-level features: Visual concept detection," *Technical Report*, 2013. 1, 2

[4] ——, "Higher-order occurrence pooling for bags-of-words: Visual concept detection," *TPAMI*, 2016. 1, 2, 3, 5, 9

[5] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields," *CoRR abs/1812.08008*, 2018. 1, 8, 11, 12

[6] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, 2013. 1

[7] T. Mahmud, M. Hasan, and A. K. Roy-Chowdhury, "Joint prediction of activity labels and starting times in untrimmed videos," *ICCV*, 2017. 1

[8] A. Cherian, B. Fernando, M. Harandi, and S. Gould, "Generalized rank pooling for action recognition," *CVPR*, 2017. 1, 11

[9] A. Cherian and S. Gould, "Second-order temporal pooling for action recognition," *IJCV*, 2018. 1

[10] P. Turaga and R. Chellappa, "Locally time-invariant models of human activities using trajectories on the grassmannian," *CVPR*, 2009. 1

[11] L. L. Presti and M. La Cascia, "3D skeleton-based human action classification: A survey," *Pattern Recognition*, 2015. 1, 2

[12] R. Vemulapalli, F. Arrate, and R. Chellappa, "Human action recognition by representing 3D skeletons as points in a Lie Group," *CVPR*, pp. 588–595, 2014. 1, 2, 8, 10, 11, 12

[13] M. Harandi, M. Salzmann, and F. Porikli, "Bregman divergences for infinite dimensional covariance matrices," *CVPR*, 2014. 1

[14] M. E. Hussein, M. Torki, M. Gowayyed, and M. El-Saban, "Human action recognition using a temporal hierarchy of covariance descriptors on 3D joint locations," *IJCAI*, 2013. 1, 2

[15] A. Elgammal and C.-S. Lee, "Tracking people on a torus," *TPAMI*, 2009. 1

[16] B. Li, O. I. Camps, and M. Sznaier, "Cross-view activity recognition using hankelets," *CVPR*, 2012. 1

[17] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *NIPS*, 2014. 1, 8

[18] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," *ICCV*, 2015. 1

[19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," *CVPR*, 2014. 1

[20] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *CVPR*, 2015. 1

[21] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, "Temporal segment networks: Towards good practices for deep action recognition," *ECCV*, 2016. 1

[22] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," *CVPR*, 2018. 1

[23] G. Chéron, I. Laptev, and C. Schmid, "P-CNN: Pose-based CNN Features for Action Recognition," *ICCV*, 2015. 1, 2

[24] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *TPAMI*, 2013. 1

[25] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele, "A database for fine grained activity detection of cooking activities," *CVPR*, 2012. 1, 2, 7, 8

[26] C. Wang, Y. Wang, and A. L. Yuille, "An approach to pose-based action recognition," *CVPR*, 2013. 1, 2

[27] S. Zuffi and M. J. Black, "Puppet flow," *IJCV*, vol. 101, no. 3, pp. 437–458, 2013. 1, 2

[28] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," *CVPR*, 2018. 1, 8, 11

[29] P. Koniusz, A. Cherian, and F. Porikli, "Tensor representations via kernel linearization for action recognition from 3D skeletons," *ECCV*, 2016. 1, 2

[30] A. Cherian, P. Koniusz, and S. Gould, "Higher-order pooling of cnn features via kernel linearization for action recognition," *WACV*, 2017. 1, 11

[31] L. Xia, C.-C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3D joints," *CVPR Workshops*, pp. 20–27, 2012. 2, 7, 8, 10

[32] L. Seidenari, V. Varano, S. Berretti, A. D. Bimbo, and P. Pala, "Recognizing actions from depth cameras as weakly aligned multi-part bag-of-poses," *CVPR Workshop*, 2013. 2, 7, 8, 10

[33] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3D points," *CVPR Workshop*, pp. 9–14, 2010. 2, 7, 8, 10, 11

[34] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," *ICCV*, pp. 2556–2563, 2011. 2, 7, 8

[35] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+ d: A large scale dataset for 3d human activity analysis," *CVPR*, pp. 1010–1019, 2016. 2, 7, 8

[36] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," *CoRR abs/1705.06950*, 2017. 2, 7, 8, 11

[37] V. M. Zatsiorsky, "Kinematic of human motion," *Human Kinetics Publishers*, 1997. 2

[38] G. Johansson, "Visual perception of biological motion and a model for its analysis," *Perception and Psychophysics*, vol. 14, no. 2, pp. 201–211, 1973. 2

[39] F. Lv and R. Nevatia, "Recognition and segmentation of 3-D human action using hmm and multi-class adaboost," *ECCV*, pp. 359–372, 2006. 2

[40] V. Parameswaran and R. Chellappa, "View invariance for human action recognition," *IJCV*, vol. 66, no. 1, pp. 83–101, 2006. 2

[41] Y. Wu, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," *CVPR*, 2012. 2, 10, 11

[42] X. Yang and Y. Tian, "Effective 3D action recognition using eigenjoints," *J. Vis. Comun. Image Represent.*, vol. 25, no. 1, pp. 2–11, 2014. 2

[43] Y. Yacoob and M. J. Black, "Parameterized modeling and recognition of activities," *ICCV*, pp. 120–128, 1998. 2

[44] E. Ohn-Bar and M. M. Trivedi, "Joint angles similarities and HOG^2 for action recognition," *CVPR Workshop*, 2013. 2

[45] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy, "Sequence of the most informative joints (SMIJ)," *J. Vis. Comun. Image Represent.*, vol. 25, no. 1, pp. 24–38, 2014. 2

[46] L. Bo, K. Lai, X. Ren, and D. Fox, "Object recognition with hierarchical kernel descriptors," *CVPR*, 2011. 2

[47] L. Wang, J. Zhang, L. Zhou, C. Tang, and W. Li, "Beyond covariance: Feature representation with nonlinear kernel matrices," *ICCV*, 2015. 2, 10

[48] J. Cavazza, A. Zunino, M. S. Biagio, and M. Vittorio, "Kernelized covariance for action recognition," *CoRR abs/1604.06582*, 2016. 2

[49] J. Zhang, L. Wang, and L. Zhou, "Beyond covariance: Sice and kernel based visual feature representation," *IJCV*, 2020. 2

[50] A. Gaidon, Z. Harchoui, and C. Schmid, "A time series kernel for action recognition," *BMVC*, pp. 63.1–63.11, 2011. 2

[51] T.-K. Kim, K.-Y. K. Wong, and R. Cipolla, "Tensor canonical correlation analysis for action classification," *CVPR*, 2007. 2

[52] A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," *ICML*, 2005. 2

[53] M. A. Vasilescu and D. Terzopoulos, "Tensortextures: multilinear image-based rendering," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 336–342, 2004. 2

[54] ——, "Multilinear analysis of image ensembles: Tensorfaces," *ECCV*, 2002. 2
[55] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "A survey of multilinear subspace learning for tensor data," *Pattern Recognition*, vol. 44, no. 7, pp. 1540–1551, 2011. 2
[56] P. Koniusz and A. Cherian, "Sparse coding for third-order super-symmetric tensor descriptors with application to texture recognition," *CVPR*, 2016. 2
[57] X. Zhao, S. Wang, S. Li, and J. Li, "A comprehensive study on third order statistical features for image splicing detection," *Digital Forensics and Watermarking*, pp. 243–256, 2012. 2
[58] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," *CVPR*, 2016. 2
[59] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," *ECCV*, 2016. 2
[60] Y. Zhou, B. Ni, R. Hong, M. Wang, and Q. Tian, "Interaction part mining: A mid-level approach for fine-grained action recognition," *CVPR*, 2015. 2
[61] E. Shechtman and M. Irani, "Space-time behavior based correlation," *CVPR*, 2005. 2
[62] P. Koniusz and H. Zhang, "Power normalizations in fine-grained image, few-shot image and graph classification," *TPAMI*, 2020. 2
[63] L. Wang, P. Koniusz, and D. Q. Huynh, "Hallucinating idt descriptors and i3d optical flow features for action recognition with cnns," *ICCV*, 2019. 2
[64] T.-Y. Lin, S. Maji, and P. Koniusz, "Second-order democratic aggregation," *ECCV*, 2018. 2
[65] H. Zhang, L. Zhang, X. Qi, H. Li, P. H. S. Torr, and P. Koniusz, "Few-shot action recognition with permutation-invariant attention," *ECCV*, pp. 525–542, 2020. 2
[66] H. Zhang and P. Koniusz, "Power normalizing second-order similarity network for few-shot learning," *WACV*, 2019. 2
[67] C. Simon, P. Koniusz, R. Nock, and M. Harandi, "Adaptive subspaces for few-shot learning," *CVPR*, 2020. 2
[68] S. Zhang, D. Luo, L. Wang, and P. Koniusz, "Few-shot object detection by second-order pooling," *ACCV*, 2020. 2
[69] C. Simon, P. Koniusz, R. Nock, and M. Harandi, "On modulating the gradient for meta-learning," *ECCV*, 2020. 2
[70] F. Shiri, X. Yu, F. Porikli, and P. Koniusz, "Face destylization," *DICTA*, 2017. 2
[71] F. Shiri, F. Porikli, R. Hartley, and P. Koniusz, "Identity-preserving face recovery from portraits," *WACV*, 2018. 2
[72] F. Shiri, X. Yu, F. Porikli, R. Hartley, and P. Koniusz, "Recovering faces from portraits with auxiliary facial attributes," *WACV*, 2019. 2
[73] ——, "Identity-preserving face recovery from stylized portraits," *IJCV*, vol. 127, no. 6-7, pp. 863–883, 2019. 2
[74] L. Wang and P. Koniusz, "Self-supervising action recognition by statistical moment and subspace descriptors," *ACM Multimedia*, 2021. 2
[75] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," *AAAI*, 2018. 2, 8, 10, 11, 12
[76] K. Sun, P. Koniusz, and Z. Wang, "Fisher-bures adversary graph convolutional networks," *UAI*, vol. 115, pp. 465–475, 2019. 2
[77] H. Zhu and P. Koniusz, "Simple spectral graph convolution," *ICLR*, 2021. 2
[78] ——, "REFINE: Random RangE FInder for network embedding," *CIKM*, 2021. 2
[79] T. Huckle, www5.in.tum.de/persons/huckle/tensor-kurs_1.pdf, 2019. 3
[80] T. Jebara, R. Kondor, and A. Howard, "Probability product kernels," *JMLR*, vol. 5, pp. 819–844, 2004. 3
[81] H. Jégou, M. Douze, and C. Schmid, "On the burstiness of visual elements," *CVPR*, pp. 1169–1176, 2009. 5
[82] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *TPAMI*, vol. 33, no. 3, pp. 500–513, Mar. 2011. 8
[83] H. Wang, A. Kläser, C. Schmid, and C. L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *IJCV*, vol. 103, no. 1, pp. 60–79, 2013. 9
[84] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the Fisher Kernel for Large-Scale Image Classification," *ECCV*, 2010. 9
[85] A. B. Tanfous, H. Drira, L. Zhou, and B. B. Amor, "Coding kendall's shape trajectories for 3d action recognition," *CVPR*, 2018. 10
[86] Y. Tas and P. Koniusz, "Cnn-based action recognition and supervised domain adaptation on 3d body skeletons via kernel feature maps," *BMVC*, 2018. 10
[87] M. Zanfir, M. Leordeanu, and C. Sminchisescu, "The moving pose: An efficient 3D kinematics descriptor for low-latency action recognition and detection," *ICCV*, 2013. 10
[88] Q. Ke, S. Bennamoun, M. ana An, F. Sohel, and F. Boussaid, "A new representation of skeleton sequences for 3d action recognition," *CVPR*, 2017. 10
[89] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View adaptive recurrent neural networks for high performance human action recognition from skeleton data," *ICCV*, 2017. 10
[90] J. Wang and A. Cherian, "Learning discriminative video representations using adversarial perturbations," *ECCV*, 2018. 10, 11
[91] B. Li, M. He, X. Cheng, Y. Chen, and Y. Dai, "Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep cnn," *CoRR abs/1704.05645v2*, 2017. 10
[92] J.-F. Hu, W.-S. Zheng, J. Pan, J. Lai, and J. Zhang, "Deep bilinear learning for rgb-d action recognition," *ECCV*, 2018. 10
[93] A. Cherian, S. Sra, S. Gould, and R. Hartley, "Non-linear temporal subspace representations for activity recognition," *CVPR*, 2018. 11
[94] Y. Shi, Y. Tian, Y. Wang, W. Zeng, and T. Huang, "Learning long-term dependencies for action recognition with a biologically-inspired deep network," *ICCV*, 2017. 11
[95] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge University Press, 2004. 13
[96] P. Koniusz, A. Cherian, and F. Porikli, "Tensor representations via kernel linearization for action recognition from 3D skeletons (extended version)," *CoRR abs/1604.00239*, 2016. 13

Piotr Koniusz. A Senior Researcher in Machine Learning Research Group at Data61/CSIRO (NICTA), and a Senior Honorary Lecturer at the Australian National University (ANU). He was a postdoctoral researcher in the team LEAR, INRIA, France. He received his BSc in Telecommunications and Software Engineering in 2004 from the Warsaw University of Technology, Poland, and completed his PhD in Computer Vision in 2013 at CVSSP, University of Surrey, UK.

Lei Wang. He received the M.E. degree in software engineering from The University of Western Australia (UWA), Australia, in 2018. Since then, he has worked as a Computer Vision Researcher at iCetana Pty Ltd. He is currently a PhD student at the Australian National University and Data61/CSIRO under the supervision of Dr. Piotr Koniusz. His research interests include human action recognition in videos, machine learning and computer vision.

Anoop Cherian. A Research Scientist at Mitsubishi Electric Research Labs (MERL) Cambridge, MA and an Adjunct Researcher at the Australian Centre for Robotic Vision (ACRV) at the Australian National University, Canberra. Before joining ANU, he was a postdoctoral researcher in the LEAR team at INRIA, Grenoble. He received my MS and PhD in 2010 and 2013 from the University of Minnesota, Minneapolis, USA. He received his undergraduate (honors) degree in computer science and engineering at the National Institute of Technology (NIT), Calicut, India in 2002.