Scalable and Compact 3D Action Recognition with Approximated RBF Kernel Machines

Jacopo Cavazza[a], Pietro Morero[a], Vittorio Murino[a,b]

[a] *Pattern Analysis and Computer Vision, Istituto Italiano di Tecnologia via Morego 30, 16163 Genova Bolzaento, Italy*
[b] *Computer Science Department, Università degli studi di Verona, Italy*

Abstract

Despite the recent deep learning (DL) revolution, kernel machines still remain powerful methods for action recognition. DL has brought the use of large datasets and this is typically a problem for kernel approaches, which are not scaling up efficiently due to kernel Gram matrices. Nevertheless, kernel methods are still attractive and more generally applicable since they can equally manage different sizes of the datasets, also in cases where DL techniques show some limitations. This work investigates these issues by proposing an explicit approximated representation that, together with a linear model, is an equivalent, yet scalable, implementation of a kernel machine. Our approximation is directly inspired by the exact feature map that is induced by an RBF Gaussian kernel but, unlike the latter, it is finite dimensional and very compact. We justify the soundness of our idea with a theoretical analysis which proves the unbiasedness of the approximation, and provides a vanishing bound for its variance, which is shown to decrease much rapidly than in alternative methods in the literature. In a broad experimental validation, we assess the superiority of our approximation in terms of 1) ease and speed of training, 2) compactness of the model, and 3) improvements with respect to the state-of-the-art performance.

Keywords: Kernel Machines, Kernel Approximation, Action Recognition, Skeletal Joints, Covariance Representation

Email addresses: jacopo.cavazza@iit.it (Jacopo Cavazza), pietro.morerio@iit.it (Pietro Morero), vittorio.murino@iit.it (Vittorio Murino)

1. Introduction

Action recognition is a paramount research domain in machine intelligence and computer vision, being nowadays ubiquitous in many application domains such as human-robot interaction, autonomous driving, elderly care and video-surveillance, just to name a few [1]. Yet, major difficulties arise when dealing with videos due to general visual ambiguities such as illumination variations, the presence of clutter/noise in the scene, occlusions or unfavorable recording viewpoint. Moreover, the variability of action evolution, as either executed by different human subjects or implicit in the structure of the action execution, further contributes to complicate the classification process. Fortunately, the adoption of novel range sensors constitutes an effective countermeasure as they provide alternative data to process, more robust to the above mentioned issues. Actually, this type of sensors (e.g. Kinect) also allows to represent a given action – other than by dense range data – as a collection of skeletal joint positions progressing in time, through real-time algorithms [2]. Action recognition can thus be reformulated as the problem of classifying the multivariate time-series $\mathbf{P} \in \mathbb{R}^{3J \times T}$, which collect the three-dimensional coordinates of the J skeletal joints positions over T temporal acquisitions.

Within the data structure $\mathbf{P}$, J is fixed by the selection of the device which acquires the joints (*e.g.*, Kinect or VICON), while T typically changes across instances. Therefore, a minimal requirement for encoding this data is to be invariant to the variability of T. Among the possible feature encoding methods (see [1] for a literature review), the symmetric and positive definite (SPD) covariance (COV) operator guarantees this property, while also demonstrated to score a solid performance in 3D action recognition [3, 4, 5, 6, 7]. Actually, in addition to properly modeling the skeletal dynamics with a second order statistics, the COV operator is also naturally able to handle different temporal durations of the action instances. This avoids slow pre-processing stages such as time warping or interpolation [8], needed to "re-align" the different sequences

before the actual classification. Moreover, performance achieved by COV-based methods are always comparable and sometimes superior to the one achieved by deep learning methods [9, 10, 11, 12, 13, 14, 15, 16], which, instead, typically require a massive amount of data and large computational power (on GPUs) for training.

All covariance-based paradigms for action recognition can be framed as the problem of classifying $d \times d$ data instances $\mathbf{X}$. In the case of skeleton data, $d = 3J$ and $\mathbf{X} = \frac{1}{T-1}\mathbf{P}\mathbf{J}\mathbf{P}^\top$, where $\mathbf{J} = \frac{1}{T}\mathbf{I} - \mathbf{1}_{T\times T}$ (being $\mathbf{I}$ the identity matrix) is the centering matrix as defined in [17, 18]. To accomplish such task, kernel theory [19] naturally promotes max-margin approaches in order to learn decision boundaries maximally separating (action) classes. Interestingly, this can be done by *only* evaluating a kernel function K that, in our work, is fixed as the Radial Basis Function (RBF) Gaussian kernel:

$$K(\mathbf{X}, \mathbf{Y}) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{X} - \mathbf{Y}\|_F^2\right). \tag{1}$$

The choice of this kernel is motivated by a set of beneficial properties, *i.e.*, 1) invariance to translations, 2) isotropy and 3) infinite-smoothness. Moreover, due to its robustness with respect to the parameter σ, it has been broadly and effectively used in the literature for many tasks [19, 20, 21, 22, 23, 17, 24, 18]. More specifically, when applying the change of variables $\mathbf{X} = \log(\frac{1}{T-1}\mathbf{P}\mathbf{J}\mathbf{P}^\top)$, equation (1) becomes the log-Euclidean kernel, which, thanks to its strong theoretical properties, is well suited to compare SPD matrices [25]. To this end, it has been widely exploited in computer vision and related fields, such as action recognition [5] or pedestrian re-identification [26], to name a few.

Unfortunately, this approach has a limited scalability, since (1) has to be computed for each pair of examples within the training set $\{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$ and for each ordered pair across training and test sets $\{\mathbf{Y}_1, \ldots, \mathbf{Y}_M\}$. This yields to the training and test Gram matrices $K(\mathbf{X}_i, \mathbf{X}_j)$ and $K(\mathbf{Y}_k, \mathbf{X}_i)$, $i, j = 1, \ldots, N$ and $k = 1, \ldots, M$, respectively. In the case of large number of samples M and/or N, Gram matrices are quite hard to both store and manipulate when performing the optimization to determine the decision boundaries. For instance,

if $M, N \sim 10^4$, about 10^{12} products are required to perform a matrix inversion, which will likely result in an out-of-memory error.

Such problem can be circumvented if we are able to obtain an explicitly computable feature representation $\boldsymbol{\phi}$ such that $\langle \boldsymbol{\phi}(\mathbf{X}), \boldsymbol{\phi}(\mathbf{Y}) \rangle$ equals (1), even approximately. In fact, while a linear machine fed with $\boldsymbol{\phi}$ is theoretically equivalent to a kernel machine (thanks to the kernel trick [19]), training a linear SVM is scalable even in the big data regime, differently from an exact kernel SVM [27, 28, 29]. However, despite a few approximation schemes have been proposed [20, 21, 22, 23, 30], there is not yet a definitive answer about which performs the best in applicative settings.

With respect to all the problems presented above, our paper provides the following contributions.

1. We propose a novel, explicit *random* feature map, which can rigorously be interpreted as a compact approximation inspired by the exact (and infinite-dimensional) feature encoding induced by (1).

2. We theoretically show that, marginalizing the sources of randomness, the proposed estimator of (1) is unbiased, and its variance has an explicit upper bound that is i) more clearly interpretable and ii) more rapidly decreasing as a function of the size of the approximation. These properties make our approach more favorable with respect to competing methods in the literature [20, 21, 22, 23, 30].

3. Our formalism is general enough to recover a previously proposed approximation scheme [6] as a particular case and, at the same time, we endow the action recognition pipeline [7] with a theoretical background that justifies its empirical performance.

4. Differently to previous works [20, 21, 22, 23, 30] where feature approximation schemes are tested on controlled benchmarks against the exact kernel machine only, we perform an extensive validation against state-of-the-art approach in human action recognition from skeletal data. As the results certify, our method guarantees a compact representation, a solid classification performance and a remarkable speed of training.

The rest of the paper is structured as follows. Section 2 recaps the relevant works in action recognition and kernel approximation. In Section 3, we dissect the proposed approximation in formal terms. The experimental validation is presented in Sections 4 and 5. Finally, Section 6 draws conclusions, profiles limitations and sketches the future work.

2. Related Work

In this Section, we discuss some of the most relevant related works in the field of 3D human action recognition, focusing on state-of-the art approaches in (approximated) kernel methods and feature learning.

Kernel methods. Within 3D action recognition methods on manifolds, a major role is played by symmetric and positive definite (SPD) matrices and, among them, covariance operators. The latter are either extended to the infinite dimensional case [4] or hierarchically combined in a temporal pyramid [31]. The conceptual analogy with trial-specific kernel matrices is investigated [3, 32], whereas kernelized covariance can capture arbitrary non-linear relationships [5].

Alternatively, Hankel matrices proficiently model action dynamics when used in tandem with a Hidden Markov Model [33] or a Riemannian nearest neighbors with class-prototypes [34]. As a slightly different paradigm, the Lie group [8] and associated Lie algebra [35] of the special Euclidean group of roto-translations are very effective in classifying skeletal joints temporal sequences.

However, as already mentioned, kernel methods usually do not scale up easily to big datasets due to demanding storage and computational costs. There are various possible solutions available in the literature: smooth differentiable approximations of kernel machine in the primal form [36], low-dimensional subspaces guided by information theoretical tools [37, 38], random projections [39, 40, 41] or hashing [42]. Among them, instead of the exact kernel function k, an explicit feature map ϕ is computed, so that the induced linear kernel $\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$ approximates $k(\mathbf{x}, \mathbf{y})$. Our work belongs to this class of approaches. Within the latter proposed methods, a few works [20, 22, 23, 30], exploit the

formalism of the Fourier Transform. Recently, Kar & Karnick [21] have proposed an approximated feature maps for dot product kernels $k(\mathbf{x}, \mathbf{y}) = k(\langle \mathbf{x}, \mathbf{y} \rangle)$ by leveraging on the Taylor expansion of k.

Feature learning. The representation for skeletal joints can be learned from the data itself. Du et al. [43] propose a hierarchy of bidirectional recurrent neural networks to represent in a bottom-up fashion all the structural relationships between joints in the human skeleton. Starting from legs, arms and torso, modeled with separated networks, higher levels of the hierarchy aggregate all parts while a final softmax layer is responsible for the final action classification. Long-Short Term Memory (LSTM) models can be proficiently applied to 3D action recognition. Indeed, after the introduction of the first modern big-size dataset for 3D action recognition from joints [9], the performance of LSTM networks achieves the state-of-the-art level by either performing a direct training on the raw joint coordinates of the human body [9] or implementing the true human skeleton structure with a direct acyclic graph [10] and, eventually, recurring to attention mechanisms [11]. Recently, multiple deep RNN [44] and LSTM [45] have been combined in an adaptive tree-structure for hierarchical classification. Alternatively, joint trajectories are used to produce distance maps, then converted into images to fine-tune convolutional neural networks (CNN), which can be therefore applied for 3D action recognition [15, 16, 12].

2.1. Originality aspects

In this work, we propose two novel approximating feature maps that are explicitly designed for kernel machines fed with covariance operators. With respect to previously proposed approximating schemes [20, 22, 23, 30, 21], our method is endowed with stronger theoretical guarantees and achieves better performance.

When compared with state-of-the-art methods for 3D action recognition, our method is more compact and scalable with respect to kernel methods [3, 32, 8, 5, 35] and, when compared with deep learning methods [9, 10, 15, 16, 12, 11], our pipeline is easier and faster to train, yet reaching comparable performance.

This approach extends two our previous works [6, 7]. With respect to [6], we propose an alternative strategy to deploy an approximation which improves upon the previous method while having the same computational complexity. Further, we extend the experimental validation to a new dataset (the large-scale NTU RGB+D [9] benchmark) while also completing the analysis by discussing the computational cost. Finally, we propose a variation of the approximating scheme of [6] such that performance can be improved while maintaining the same computational burden. With respect to [7], we show that the architecture thereby proposed can be framed into our theoretical analysis as a particular case. In fact, we can interpret it as a linearization of our proposed encoding where, as to recover from such compression, we use a data-driven learning scheme to replace the random sampling of weights and boost its descriptiveness. Finally, we extend its experimental validation on new datasets with state-of-the-art comparisons.

3. Approximating the RBF kernel with Kronecker products

In this Section, we present in formal terms our original technique to approximate the RBF kernel (1) by means of a low-dimensional and explicit feature map, characterized by a random component which is ultimately responsible of the quality of the approximation itself. Indeed, when averaging upon all the possible realization of such component, our representation approximates (1) with zero bias. Additionally, the variance of such estimation can be controlled by an explicit upper bound that easily writes as a function which rapidly decreases as the feature dimensionality increases.

3.1. Construction of the approximated feature map

Given $\mathbf{X} \in \mathbb{R}^{d \times d}$ and fixed a strictly positive integer ν, that corresponds to the feature dimensionality, our approximation is defined as follows.

Definition 1. *We define a ν dimensional vector $\boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X})$ whose components $\phi_{\mathsf{kron}-\pi,1}(\mathbf{X}), \ldots, \phi_{\mathsf{kron}-\pi,\nu}(\mathbf{X})$ are ($1/\sqrt{\nu}$-multiplied) independent realizations*

of the following scalar function

$$\varphi_{\mathsf{kron}-\pi}(\mathbf{X}) = \frac{1}{\sigma^{2n}}\sqrt{\frac{\exp(-\frac{1}{\sigma^2})}{\rho(n)n!}}\mathrm{tr}\left(\otimes_{\kappa=1}^{n}{\mathbf{W}^{(\kappa)}}^{\top}\mathbf{X}\right). \tag{2}$$

In (2)*,* $\sigma > 0$ *defines the bandwidth of the kernel function* (1)*,* n *is sampled from* any *distribution* ρ *supported over the integers. Furthermore, the following assumptions are made:*

A.1 $\mathbf{W}^{(\kappa)}$ *are (elementwise) drawn from the distribution* $\mathcal{P}$ *with null expected value and standard deviation equals to the kernel's bandwidth* σ*.*

A.2 The $d \times d$ *matrix which is inputted to* $\varphi_{\mathsf{kron}-\pi}$ *lies on the Frobenius norm-unitary sphere, that is* $\|\mathbf{X}\|_F = 1$*.*

Note that the $\varphi_{\mathsf{kron}-\pi}(\mathbf{X})$ has two sources of randomness. First, the integer n, which is sampled from ρ. Second, precisely n matrices $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(\kappa)}, \ldots, \mathbf{W}^{(n)}$ are sampled, so that each of their element is independently drawn from $\mathcal{P}$. More in detail, for each $\kappa = 1, \ldots, n$, the transpose of $\mathbf{W}^{(\kappa)}$ is (row-by-column) multiplied by $\mathbf{X}$. Afterwards, the results of the previous operation are combined together with a Kronecker product and, finally, the trace operator is evaluated. For the sake of clarity, let us notice that, since the trace operator applied on matrix returns a scalar, $\varphi_{\mathsf{kron}-\pi}(\mathbf{X}) \in \mathbb{R}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X}) \in \mathbb{R}^{\nu}$, since it stacks ν independent realizations of $\varphi_{\mathsf{kron}-\pi}(\mathbf{X})$ (divided by $\sqrt{\nu}$, which is factorized out of the definition of φ only for convenience in the demonstrations). Algorithm 1 provides the pseudo-code for the construction process.

With respect to the assumptions A.1 and A.2, the first one constrains the distribution $\mathcal{P}$. Indeed, let us notice that, in all our theoretical exposition, the distributions ρ and $\mathbf{P}$ are allowed to be highly general, and we will specify them only in the experiments when we need to numerically sample from them. For instance, A.1 is satisfied if $\mathcal{P} = \mathcal{N}(0, \sigma^2)$, being fixed as a zero-mean Gaussian with σ^2 variance.

Instead, A.2 is only technical and does not really represent a constraint under an applicative point of view. Indeed, given an arbitrary input data $\mathbf{X}$, we can

Algorithm 1: Approx, by Kronecker product.

Input: A normalized $d \times d$ input matrix $\mathbf{X}$, the desired feature size ν, the probability distributions ρ over integers and $\mathcal{P}$ over real numbers, the kernel bandwith $\sigma > 0$.

Output: $[\phi_{\mathsf{kron}-\pi,1}(\mathbf{X}), \ldots, \phi_{\mathsf{kron}-\pi,\nu}(\mathbf{X})]$

foreach $j = 1, \ldots, \nu$ **do**

1 Sample n according to ρ

foreach $\kappa = 1, \ldots, n$ **do**

2 Sample $\mathbf{W}^{(\kappa)} \in \mathbb{R}^{d \times d}$ from $\mathcal{P}$ elementwise.

end

3 Compute the scalar $\pi(\mathbf{X}) = \mathrm{tr}\left(\otimes_{\kappa=1}^{n} {\mathbf{W}^{(\kappa)}}^{\top} \mathbf{X}\right)$

4 **Return** $\phi_{\mathsf{kron}-\pi,\mathsf{j}}(\mathbf{X}) = \sigma^{-2n} \left(\frac{\exp(-\sigma^{-2})}{\nu \rho(n) n!}\right)^{1/2} \pi(\mathbf{X})$

end

achieve A.2 by dividing $\mathbf{X}$ entrywise by $\|\mathbf{X}\|_F$. Such operation is easy to perform and it is along the line of the classical pre-processing which is applied on the data before passing them to a kernel method - as for instance, the component-wise division by the standard deviation is a common preprocessing step before SVM training [19]. If compared with similar results in [20, 22, 23, 21, 30], the assumption of unitary norm for $\mathbf{X}$ and $\mathbf{Y}$ is in line with the analogous assumptions of sampling the data from a given submanifold - with the remarkable difference that our assumption is easy to satisfy also in an applicative domain.

Before digging into the details of the theoretical foundation, lets us provide the intuition behind equation (2).

3.2. Intuition behind the genesis of $\varphi_{\mathsf{kron}-\pi}$

According to the well established kernel theory [19], the exact feature map $\mathbf{f}$ associated to the RBF kernel (1) is infinite-dimensional. Still, it can be expressed in closed form. In fact, without loss of generality, let us assume $d = 1$ and, for the sake of simplicity, let $\sigma = 1$. Consequently, we replace the matrices $\mathbf{X}, \mathbf{Y}$

with the scalars x, y and, in such a case, the kernel function (1) rewrites as $K(x, y) = \exp(-\frac{1}{2}(x - y)^2)$.

We would like to write the exact infinite dimensional feature map $x \mapsto \mathbf{f}(x)$ for such RBF kernel, i.e. the exact infinite-dimensional vector $\mathbf{f}(\cdot)$ such that

$$\langle \mathbf{f}(x), \mathbf{f}(y) \rangle = K(x, y) = \exp(-(x - y)^2/2) \tag{3}$$

where the inner product $\langle \cdot, \cdot \rangle$ is computed over the square-integrable series of $\mathbf{f}(\cdot)$. Since

$$\exp(-(x - y)^2/2) = \exp(-x^2/2) \cdot \exp(xy) \cdot \exp(-y^2/2), \tag{4}$$

we can take advantage of the Taylor expansion to obtain

$$\mathbf{f}(x) = \sqrt{e^{-x^2}} \left[1, x, \frac{x^2}{\sqrt{2!}}, \frac{x^3}{\sqrt{3!}}, \dots, \frac{x^n}{\sqrt{n!}}, \dots \right]. \tag{5}$$

As certified by Lagrange's remainder formula for Taylor expansions [46], a good approximation of (5) is obtained by considering all the terms which are less or equal to an arbitrary degree n. In the scalar case, these terms are exactly n. Differently, in order to compute the products for $d > 1$, the terms of a given degree n must include all the possible combinations $X_{11}^{\alpha_{11}} X_{12}^{\alpha_{12}} \cdots X_{ij}^{\alpha_{ij}} \cdots X_{dd}^{\alpha_{dd}}$, where X_{ij} are the components of $\mathbf{X}$ and α_{ij} are d^2 non-negative integers such that $\sum_{ij} \alpha_{ij} = n$. That is, we have to consider all the $n/\prod_{ij} \alpha_{ij}!$ combinations, and this has an exponential complexity with respect to d (check [47, page 39.]). This clearly produces an exponentially-sized feature map that, as shown in [24], is formally fine but obviously not applicable in real-world datasets. In fact, as operative condition assumed in [24], d needs to be less than 4.

Since the analytical pipeline inspired by Taylor's remainder theorem is not viable in practical pattern analysis, in this work we propose a manageable alternative solution. When asked to build a ν-dimensional representation, we repeat ν times the following pipeline. We sample n from ρ and we use n as a pointer to index which component of (5) to sample. Then, as a surrogate technique for computing all the possible combinations of products of degree n, we introduce $\otimes_{\kappa=1}^{n} {\mathbf{W}^{(\kappa)}}^{\top} \mathbf{X} = \mathbf{W}^{(1)\top} \otimes \mathbf{X} \otimes \dots \mathbf{W}^{(n)\top} \otimes \mathbf{X}$. The latter is directly

inspired from the technique of random rescaling, which is common practice in random approximated feature map approaches [20, 22, 23, 30, 21], where introducing random projections can be interpreted as a trick to "recover" from the sparse sampling of n. In the limit case where $\mathbf{W}^{(\kappa)}$ are identity matrices, $\otimes_{\kappa=1}^{n} {\mathbf{W}^{(\kappa)}}^{\top} \mathbf{X} = \mathbf{X}^{\otimes n}$ and we can find a clear analogy between scalar exponentiation in (5) and Kronecker exponentiation in (2), being the latter a $d \times d$ generalization of the former.

3.3. Unbiasedness and variance bound

In this Section, we demonstrate that, thanks to assumptions A.1 and A.2, once averaging upon all possible realizations of n from ρ and $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(n)}$ from $\mathcal{P}$, we have no bias in approximating the kernel - that is, the expected value of our $\langle \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{Y}) \rangle$ coincides with (1) and, at the same time, we are able to control the variance of the estimation.

Unbiasedness of $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$. As previously explained, an exact feature map $\mathbf{f}$ is able to satisfy the equality $\langle \mathbf{f}(\mathbf{X}), \mathbf{f}(\mathbf{Y}) \rangle = K(\mathbf{X}, \mathbf{Y})$. Thanks to the well established kernel trick [19], one does not need to compute $\mathbf{f}$ explicitly but, instead, a kernel machine can be trained by evaluating the kernel function only. In many cases (like the one of RBF kernel (1)), computing $\mathbf{f}$ explicitly is impossible due to its infinite dimension. Moreover, on the opposite, computing the kernel function does not scale to big datasets, since evaluating $K(\mathbf{X}, \mathbf{Y})$ for *every* $\mathbf{X}$ and $\mathbf{Y}$ has a quadratic complexity. Due to the prohibitive size of the Gram matrices, either the training or inference stages may be simply not computationally affordable (typically because of out-of-memory issues).

In order to accommodate for that, we propose to replace $\mathbf{f}$ with a map $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$, such that

$$\langle \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{Y}) \rangle \approx K(\mathbf{X}, \mathbf{Y}) \tag{6}$$

with the crucial difference that $\boldsymbol{\phi}$ is explicitly computable. In other words, while the kernel trick allows to replace the feature map $\mathbf{f}$ with the kernel function K, we revert the perspective, and evaluate the kernel function with $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$, which,

differently from $\mathbf{f}(\mathbf{X})$, is finite-dimensional and explicitly computable. In fact, a linear model fed with $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ is a theoretically valid estimate for the exact kernel machine fed with (1).

As well established in the literature that similarly proposed random approximated feature maps [20, 21, 30, 22, 23], we want to demonstrate the validity of the approximation by showing that, once averaging upon all the sources of randomness which affect our feature map $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$, an equality holds in eq. 6. In other words, we want to prove the absence of biases in the approximation.

Theorem 1 (Unbiased approximation for $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$)**.** *With the previous notations, the linear kernel* $\langle \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{Y}) \rangle$ *induced by* $\varphi_{\mathsf{kron}-\pi}$ *is an unbiased estimator for* $K(\mathbf{X}, \mathbf{Y})$ *as in* (1)*. Indeed,*

$$\mathbb{E}_{n,\mathcal{P}} \left[\langle \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{Y}) \rangle \right] = K(\mathbf{X}, \mathbf{Y}), \tag{7}$$

being the expected value jointly computed over all possible realizations of n *from* ρ *and of* $\mathbf{W}^{(\kappa)}$ *from* $\mathcal{P}$*,* $\kappa = 1, \ldots, n$*.*

Proof. See the Supplementary Material, Section 1. □

Bound on the variance for $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$. Theorem 1 guarantees that, on average, $\langle \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{Y}) \rangle$ is a good approximation for $K(\mathbf{X}, \mathbf{Y})$, since there is no bias. This is a strong and necessary assumption to ensure that our statistical estimator is reliable, but it does not take into account the variance, i.e. the *quality* of the approximation. Namely, even an unbiased estimator can heavily deviate from its expected value if there are no theoretical guarantees for its variance. We can prove that our estimator well behaves also in this respect, since $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ induces a linear kernel whose variance can be upper bounded as follows.

Theorem 2 (Bound on the variance of $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$)**.** *With the previous notation, the linear kernel* $\langle \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{Y}) \rangle$ *induced by* $\varphi_{\mathsf{kron}-\pi}$ *has a controlled variance which is bounded by a linear function of the feature dimensionality* ν*.*

Precisely,

$$\mathbb{var}\left[\langle \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{Y})\rangle\right] \leq \frac{C_\rho}{\nu^3} \exp\left(\frac{9\, m_4(\mathcal{P}) - 2\sigma^4}{\sigma^8}\right)$$

where the variance is computed over all possible realizations of n from ρ and all possible manners of sampling $\mathbf{W}^{(\kappa)}$ from $\mathcal{P}$ for each κ. C_ρ is defined as $C_\rho = \sum_{n=0}^{\infty} \frac{1}{\rho(n) \cdot n!}$ and $m_4(\mathcal{P})$ denotes the fourth order moment of $\mathcal{P}$.

Proof. See the Supplementary Material, Section 2. □

If we neglect the function $\exp\left(\frac{9\, m_4(\mathcal{P}) - 2\sigma^4}{\sigma^8}\right)$, which is fixed after we select $\mathcal{P}$ and the bandwidth σ in (1), the boundary on the variance rewrites as C_ρ/ν^3. This means that, as the feature dimension ν increases, the variance very sharply converges to zero as $1/\nu^3$, i.e. our approximation converges to its expected value.

The constant C_ρ may however affect the quality of this limit. For instance, if we choose ρ to be a Geometric distribution of parameter $0 < \theta \leq 1$, we have $\rho(n) = (1-\theta)^n\theta$ and one can analytically obtain

$$C_\rho = \frac{1-\theta}{\theta} \exp\left(\frac{1-\theta}{\theta}\right). \tag{8}$$

The previous function increases and diverges for $\theta \to 1^-$ and $\theta \to 0^+$ making the bound potentially loose. The limit case $\theta \approx 0$ is very unfavorable also in in practice: in such a case a value sampled from ρ is high with high probability and, therefore, many Kronecker products need to be evaluated in (2). On the opposite side, the case $\theta \approx 1$ is very favorable in practical terms since n is small with high probability and therefore the cost of computing (2) approaches the minimal one. Further considerations on the practical choice of θ are also reported in Section 4.4.

To conclude our discussion on the variance, we provide the following result, which is derived from Theorem 1 and 2 as a straightforward consequence of Chebyshev inequality.

Corollary 1. *Under the previous hypothesis, for any $\epsilon > 0$ and $\mathbf{X}, \mathbf{Y}$ $d \times d$ matrices, the probability $\mathbb{P}\left[|\langle \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{Y})\rangle - K(\mathbf{X}, \mathbf{Y})| > \epsilon\right]$ does not exceed the quantity $\frac{C_\rho}{\nu^3 \epsilon^2} \exp\left(\frac{9m_4(\mathcal{P}) - 2\sigma^4}{\sigma^8}\right)$.*

This result ensures that the probability of the undesired event

$$|\langle \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{Y})\rangle - K(\mathbf{X},\mathbf{Y})| > \epsilon$$

is small, since upper bounded by a quantity which is inversely quadratic in ϵ and inversely cubic in ν: this means that even a small value of ν ensures the latter probability to be small and guarantees the soundness of the approximation.

3.4. An alternative formulation

If inspecting equation (5), it would be natural to replace classical exponentiation - which works with scalars - with Kronecker exponentiation $\mathbf{X}^{\otimes n}$. However, with respect to the feature map $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ presented in the previous Section, one may observe that $\otimes_{\kappa=1}^{n} {\mathbf{W}^{(\kappa)}}^{\top}\mathbf{X} \neq \mathbf{X}^{\otimes n}$ for a general distribution of the weights (the equality would be true only if $\mathbf{W}^{(\kappa)}$ equals to the identity matrix $\mathbf{I}$ for every κ). We could thus argue that the following expression would be more appropriate for φ.

Definition 2. *Using the previous notations, for any $d \times d$ matrix $\mathbf{X}$ we define the scalar quantity*

$$\varphi_{\mathsf{kron}-\mathsf{e}}(\mathbf{X}) = \frac{1}{\sigma^{2n}}\sqrt{\frac{\exp(-\frac{1}{\sigma^2})}{\rho(n)n!}}\mathrm{tr}(\mathbf{V}^{\top}\mathbf{X}^{\otimes n}) \tag{9}$$

where $n \sim \rho$, we still require $\varphi_{\mathsf{kron}-\mathsf{e}}$ to satisfy Assumption A.2 (see Theorem 1), while also assuming

A′.1 The matrix $\mathbf{V}$ is the Kronecker product of n matrices of size $d \times d$, whose entries are drawn independently from $\mathcal{N}(0,\sigma^2)$ (so are consequently the entries of $\mathbf{V}$ – see suppl. material).

A.2 The $d \times d$ matrix which is inputted to $\varphi_{\mathsf{kron}-\pi}$ lies on the Frobenius norm-unitary sphere, that is $\|\mathbf{X}\|_F = 1$.

Then, define the ν dimensional vector $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}(\mathbf{X})$ where each component is an independent realization of $\varphi_{\mathsf{kron}-\mathsf{e}}(\mathbf{X})/\sqrt{\nu}$. The explicit steps to compute $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}(\mathbf{X})$ given $\mathbf{X}$ are enumerated in Algorithm 2.

At a first glance, equation (9) seems closer to an arbitrary component of the exact feature map (5). This is because, as opposed to (2), the exponentiation operator for scalars is here directly replaced with the Kronecker exponentiation for matrices. Again, as for $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$, we introduce some random weights – here, denoted by $\mathbf{V}$ in order to accommodate for the compression generated by approximating an infinite dimensional vector.

For what concerns the assumptions, A.2 was also hypothesized in Section 3.3 and can be considered as a simple pre-processing step where each entry of the data $\mathbf{X}$ is divided by $\|\mathbf{X}\|_F$. On the contrary, if we compare A.1 with A'.1, we find a remarkable difference. In fact, A.1 was only constraining the mean and variance of the distribution $\mathcal{P}$. Differently, A'.1 not only constrains the probability distribution to be Gaussian but, additionally, we have to explicitly assume that $\mathbf{V}$ factorizes as the Kronecker product of n variables. Indeed, despite $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ seems more naturally close to the exact feature map than $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$, it needs the more restrictive assumption A$'$.2. Without the latter, it is impossible to prove any theoretical result about the approximation $\langle \boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}(\mathbf{Y}) \rangle$ for (1). The reason for that is extremely technical and we illustrate it in the Supplementary Material, Section 4.

It is straightforward to see that Definition 2 actually corresponds to the generalization to the kernel (1) of the approach in [6], which is instead explicitly devised for the log-Euclidean kernel of covariance operators. Here, in fact, $\mathbf{X}$ and $\mathbf{Y}$ can be generic $d \times d$ data structures. Ultimately, we can state that the approximation devised in [6] is a particular case of $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$, which, in turn, is a reformulation of $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$. We can also prove what follows.

Theorem 3 (Unbiased approximation and bound on variance for $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$)**.** *Under the assumptions A$'$.1 and A.2, the linear kernel $\langle \boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}(\mathbf{Y}) \rangle$ induced by $\varphi_{\mathsf{kron}-\pi}$ is an unbiased estimator for $K(\mathbf{X}, \mathbf{Y}) = \exp(-\frac{1}{\sigma^2}\|\mathbf{X} - \mathbf{Y}\|_F^2)$. Actually it results*

$$\mathbb{E}_{n,\mathbf{V}} \left[\langle \boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}(\mathbf{Y}) \rangle \right] = K(\mathbf{X}, \mathbf{Y}), \tag{10}$$

being the expected value jointly computed over all possible realizations of n from

Algorithm 2: Approx, by Kronecker exponentiation

Input: A $d \times d$ input matrix $\mathbf{X}$, the desired feature size ν, the probability distributions ρ over integers and $\mathcal{P}$ over real numbers, the kernel bandwith $\sigma > 0$.

Output: $[\phi_{\mathsf{kron-e},1}(\mathbf{X}), \ldots, \phi_{\mathsf{kron-e},\nu}(\mathbf{X})]$

foreach $j = 1, \ldots, \nu$ **do**

Sample n according to ρ

Sample $\mathbf{V}$ as the Kronecker product of n random $d \times d$ matrices, each of the independently sampled from $\mathcal{P}$;

Compute the scalar $e(\mathbf{X}) = \mathrm{tr}\left(\mathbf{V}^\top \mathbf{X}^{\otimes n}\right)$

Return $\phi_{\mathsf{kron-e},\mathsf{j}}(\mathbf{X}) = \sigma^{-2n}\left(\frac{\exp(-\sigma^{-2})}{\nu\rho(n)n!}\right)^{1/2} e(\mathbf{X})$

end

ρ and of the weight matrix $\mathbf{V}$.

In addition, the variance of the proposed estimator is explicitly bounded according to the following inversely-cubic function of ν,

$$\mathbb{Var}_{n,\mathbf{V}}\left[\langle \boldsymbol{\phi}_{\mathsf{kron-e}}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron-e}}(\mathbf{Y})\rangle\right] \leq \frac{C_\rho}{\nu^3} \exp\left(\frac{3-2\sigma^2}{\sigma^4}\right).$$

Proof. See the Supplementary Material, Section 4. □

As a corollary, for any $\mathbf{X}, \mathbf{Y}$ and $\epsilon > 0$,

$$\mathbb{P}\left[|\langle \boldsymbol{\phi}_{\mathsf{kron-e}}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron-e}}(\mathbf{Y})\rangle - K(\mathbf{X}, \mathbf{Y})| > \epsilon\right] \leq \frac{C_\rho}{\nu^3\epsilon^2} \exp\left(\frac{3-2\sigma^2}{\sigma^4}\right),$$

ensuring that the proposed feature map is almost always approximating the desired kernel function in a reliable manner.

3.5. The perceptron heuristics

So far, n was randomly sampled from the distribution ρ. However, for both $\boldsymbol{\phi}_{\mathsf{kron-e}}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$, sampling big values of n increases the number of Kronecker products to be computed and this impact on the computational cost of the method which will be discussed in Section 4.4.

In this Section, we want to investigate the case where, in order to circumvent the previous issue, we fix $n = 1$ in a deterministic manner. This makes (2) and (9) formally identical and corresponds selecting only the component of degree 1 in (5). In these terms, we can interpret it as a *linearization* of the exact feature map associated to the RBF kernel function (1).

Intuitively, the randomness in n can lead to "explore" all the infinite components in the exact feature map $\mathbf{f}(\mathbf{X})$ in order to accumulate enough patterns in $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ to properly approximate the RBF Gaussian kernel. Imposing $n = 1$ can be instead thought of as a sort of linearization as to approximate $\mathbf{f}(x)$ in (5). In such a case, there is clearly a little room for the weights $\mathbf{W}^{(\kappa)}$ to help recovering from the compression. Therefore, as an opposed paradigm to randomly sample the weights, we can try to learn them in a data-driven fashion, in order to promote class-disambiguation. In fact, since our ultimate goal is accomplishing the action recognition task, the perspective of learning from the data itself seems appealing, especially due to the recent outstanding performance of (deep) feature learning methods [9, 10, 15, 16, 12, 11, 7].

Motivated by the previous considerations, we are now interested in *learning* the weights of $\varphi_{\mathsf{kron}-\pi}$ from data. We propose to do so by taking advantage of the formal analogy between $\varphi_{\mathsf{kron}-\pi}$ and the hidden layer of a perceptron. Since $n = 1$, we only have $\mathbf{W}^{(1)} = \mathbf{W}$ in (2) and we can also write

$$\varphi_{\mathsf{kron}-\pi}(\mathbf{X}) \propto tr(\mathbf{W}^\top \mathbf{X}) = \langle \mathbf{W}, \mathbf{X} \rangle_F = \mathrm{vec}(\mathbf{W})^\top \mathrm{vec}(\mathbf{X}). \tag{11}$$

As a result, if we denote as $\mathbb{W}$ the $\nu \times d^2$ matrix which stacks by rows all the parameters $\mathbf{W} = \mathbf{W}^{(1)}$ of each independent realization of $\varphi_{\mathsf{kron}-\pi}$, we get that

$$\boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X}) = \mathbb{W}\mathrm{vec}(\mathbf{X})^\top \tag{12}$$

meaning that $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ actually computes the hidden representation of a (1-layer) perceptron fed with (the vectorization of) $\mathbf{X}$ as data. Furthermore, a squeezing non-linearity (such as tanh or sigmoid) function on top of (12) can be actually interpreted as a sort of data normalization which is a good practice before SVM training. Since the latter can be implemented in a neural network by means of

Algorithm 3: The perceptron heuristics.

Input: A $d \times d$ input matrix $\mathbf{X}$, a training set $\boldsymbol{\mathcal{D}}$ of $d \times d$ matrices, the desired feature size ν, the probability distributions ρ over integers and $\mathcal{P}$ over real numbers, the kernel bandwith $\sigma > 0$.

Output: The ν-dim feature map $\boldsymbol{\phi}_{\mathrm{P}}(\mathbf{X})$

1 Learn $\nu \times d^2$ weight matrix $\mathbb{W}$ from the hidden layer parameters of the architecture of [7] trained on $\boldsymbol{\mathcal{D}}$.

2 **Return** $\boldsymbol{\phi}_{\mathrm{P}}(\mathbf{X})$ as the multiplication of $\mathbb{W}$ by the vectorization of $\mathbf{X}$.

a hinge loss with weight decay, we can therefore establish a connection between our paradigm $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ + linear SVM and a feed-forward perceptron, having one hidden layer of size ν, with sigmoid non-linearities and hinge loss with weight decay for final classification.

Let us summarize the previous findings. Consider $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$, set $n = 1$ and, instead of a random sampling, learn the weights $\mathbf{W} = \mathbf{W}^{(1)}$ for each $\varphi_{\mathsf{kron}-\pi}$-component from the hidden layer of the architecture composed by a supervised feed-forward perceptron with sigmoid as non-linearities and cross entropy loss. Then, use the network to extract the feature map, that we term $\boldsymbol{\phi}_{\mathrm{P}}$, and use it in combination of a linear SVM. This can be interpreted as a deterministic implementation of $\varphi_{\mathsf{kron}-\mathsf{e}}$ and $\varphi_{\mathsf{kron}-\pi}$ where random weights' sampling is replaced with their data-driven optimization. (see the Supplementary Material, Section 5, for further details).

In Section 5, we will validate the previous heuristics of replacing $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ as given by Algorithm 1 with the map $\boldsymbol{\phi}_{\mathrm{P}}$ which is computed according to pseudo-code presented in Algorithm 3.

4. Experimental results: evaluation vs. other approximations

In this Section we will present our experimental validation of $\boldsymbol{\phi}_{\mathsf{kron}-\pi}, \boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ as well as the perceptron heuristics $\boldsymbol{\phi}_{\mathrm{P}}$. To begin with, we will describe the

benchmark datasets adopted and explain the data preprocessing that we carried out.

4.1. 3D action recognition datasets and preprocessing

We present here all the datasets considered for the experiments, namely UTKinect [48], Florence3D [49], MSR-Action-Pairs (MSR-*pairs*) [50], MSR-Action3D [51], Gaming-3D (G3D) [52], HDM-05 [53], MSRC-Kinect12 [54] and NTU RGB+D [9].

We follow usual training and testing splits proposed in the literature. For Florence3D, G3D, and UTKinect, we use the protocols of [8, 35, 34]. For MSR-Action3D, we adopt the splits originally proposed by [51]. On MSRC-Kinect12, once highly corrupted action instances are removed as in [31], training is performed on odd-index subject, while testing on the even-index ones. On HDM-05, the training split exploits all the data from the "`bd`" and "`mm`" subjects, being "`bk`", "`dg`" and "`tr`" left out for testing [3]. To be consistent with the literature, we replicated the 14 classes experiments (HDM-05_{14}) as in [3, 5]. When dealing with the whole dataset (HDM-05_{all}), since some of the total classes are missing from the training/testing splits, we adopted the protocol of [55] to partition the dataset into 65 action classes. For NTU RGB+D, we followed the authors' instructions [1] in removing the most corrupted instances, also purging the trials with missing joints recordings. Finally, we replicated both the cross-subject and cross-view testing protocols proposed in [9], denoting them as NTU-×-*subject* and NTU-×-*view*.

In all experiments, as a common data pre-processing step [8, 33, 35, 34, 9, 5, 32, 10], we fix one root joint (the one located at the hip center), and we compute the relative differences of all the other $J-1$ 3D joint positions. By doing this at any timestamps $t = 1, \ldots, T$ we obtain a $3(J-1)$-dimensional (column) vector $\mathbf{p}(t)$ of relative displacements. As the representation for data

[1] `https://github.com/shahroudy/NTURGB-D#samples-with-missing-skeletons`

instance $[\mathbf{p}(1), \ldots, \mathbf{p}(T)]$, we compute a covariance matrix

$$\mathbf{C} = \frac{1}{T-1} \sum_{t=1}^{T} (\mathbf{p}(t) - \boldsymbol{\mu})(\mathbf{p}(t) - \boldsymbol{\mu})^{\top}, \tag{13}$$

being $\boldsymbol{\mu} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{p}(t)$ the temporal average of $\mathbf{p}(t)$. Finally, the input representation for our approximated feature map is obtained as

$$\mathbf{X} = \log \mathbf{C} = \mathbf{U} \mathrm{diag}(\log(\boldsymbol{\varsigma})) \mathbf{U}^{\top}, \tag{14}$$

being $\boldsymbol{\varsigma}$ the vector of eigenvalues (eventually regularized by an additive factor as in [17]) and $\mathbf{U}$ the matrix of eigenvectors of $\mathbf{C}$. Finally, since the log of a symmetric matrix is symmetric, in order to avoid to process identical entries twice, we zero out all the lower triangular entries in $\mathbf{X}$ before dividing by $\|\mathbf{X}\|_F$.

4.2. Implementation details

The implementation for $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$, $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ and $\boldsymbol{\phi}_{\mathrm{P}}$, is in MATLAB, and is based on the pseudo-code of Algorithms 1, 2 and 3, respectively[2].

For both $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$, we fixed the distribution ρ to be a Geometric with parameter 0.9 - a full justification fro this choice is provided in Section 4.4 - and $\mathcal{P} = \mathcal{N}(0, \sigma^2)$ is a Gaussian distribution. We carried out experiments for $\nu = 10$, 20, 50, 100, 200, 500, 1000, 2000, 5000 and we averaged among 10 repetitions of each experiment, to account for the random nature of the approach.

For $\boldsymbol{\phi}_{\mathrm{P}}$ we used the architecture of [7] which is also fed with log-projected covariance representations. With such input we trained a one-hidden layer perceptron with sigmoid non-linearities and cross-entropy loss using scaled conjugate gradient descent for all datasets except to the NTU RGB+D, for which we used ADAM optimizer with mini-batches of size 1024. The size of the hidden layer was cross-validated among $10^2, 10^3, \ldots, 10^9$ as the one which gives the lowest objective value. Once the network is trained, we extracted the parameter of the hidden layer and built $\boldsymbol{\phi}_{\mathrm{P}}$ as in Algorithm 3. For all cases - $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$, $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ and $\boldsymbol{\phi}_{\mathrm{P}}$ - we use the linear SVM implementation of [27].

[2] We used MATLAB R2017a installed on an Intel Xeon(R) CPU E5645 @2.40GHz, 12 cores, with 12GB RAM.

4.3. Comparing the bounds on the variance

A direct comparison among the bounds of the variance between the proposed approximations $\boldsymbol{\phi}_{\mathsf{kron}-\pi}, \boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ and the previous approaches [20, 22, 23, 30, 21, 24, 6] is tricky because the theoretical foundation of each approximation is approached in different manners. Indeed, [24] does not rely on a probabilistic framework, but instead, proposes a simple truncation of the feature map (5). Despite this allows the relative error between the exact and the approximated kernel to be explicitly bounded, the method [24] is only applicable in a case of a small d (see Section 4.4).

Other probabilistic frameworks as [20, 22, 23, 21] also provide an analogous result of Corollary 1. However, the probability $\mathbb{P}[|\langle \boldsymbol{\phi}(\mathbf{X}), \boldsymbol{\phi}(\mathbf{Y})\rangle - K(\mathbf{X}, \mathbf{Y})| > \epsilon]$ has only a weaker $O(1/(\nu\epsilon^2))$ behavior. Moreover, while considering the analogous approximated feature maps of [20, 22, 23, 21], the previous result holds only provided that $\mathbf{X}$ and $\mathbf{Y}$ lie on a common sub-manifold. Despite we analogously assume that $\mathbf{X}$ and $\mathbf{Y}$ have unitary norm, our requirement is easier to satisfy in practice and less restrictive. Moreover, ancillary conditions are needed in those works to achieve results of the form $\mathbb{P}[|\langle \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\pi}(\mathbf{Y})\rangle - K(\mathbf{X}, \mathbf{Y})| > \epsilon]$ and $\mathbb{P}[|\langle \boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}(\mathbf{X}), \boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}(\mathbf{Y})\rangle - K(\mathbf{X}, \mathbf{Y})| > \epsilon]$, while, differently, the assumptions we made are much milder.

In addition, [30] and [6] also provide a strong theoretical foundation similar to ours. Indeed, in [30], the proposed approximation gives an unbiased estimation of (1) and its variance is bounded a $O(1/\nu)$ function. In our case, however, the variances of the approximations $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ are bounded by $O(1/\nu^3)$ and are therefore more rapidly decreasing to zero, ensuring a better approximation for a fixed ν. Finally, the quality of the approximation of [6] is comparable - no bias and $O(1/\nu^3)$ decreasing variance. This is reasonable because, as we proved, [6] is a particular case of our approximation $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$.

4.4. Computational cost

Interestingly, we can observe one common trend which is shared across all the approaches [20, 22, 23, 21, 6]: in computational terms, the number of products

required for computing one component of the feature map is linear with respect to the data dimensionality (which is $O(d^2)$ since log-covariance $d \times d$ matrices are used as input). Among the previously published works, two papers are different: [30] achieves a log-linear complexity, while, unfortunately, [24] has exponential complexity with respect to the data size: this is the reason why we were not able to include [24] among the methods in comparison.

Thus, the cost of calculating $\mathrm{tr}(\otimes_{\kappa=1}^{n} {\mathbf{W}^{(\kappa)}}^{\top} \mathbf{X}) = \prod_{\kappa=1}^{n} \mathrm{tr}({\mathbf{W}^{(\kappa)}}^{\top} \mathbf{X})$ (in the computation of $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$) is linear in both the input data dimensionality and in n. Similarly, the same holds for $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$, thanks to the factorization assumption A$'$.1.

Despite such linear dependence from n may appear as a drawback, we can take advantage of the freedom in choosing ρ in order to keep n small. Indeed, throughout all the experiments, either involving $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ or $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$, we fixed ρ as a Geometrical distribution of parameter $\theta = 0.9$. This ensures that the probability of sampling high values of n from ρ is practically zero. Indeed, through analytical computations, we can also notice that, for each realization of $\varphi_{\mathsf{kron}-\pi}$ or $\varphi_{\mathsf{kron}-\mathsf{e}}$, $\mathbb{P}(n > 3) = 0.04$.

This makes the computational cost of our approach substantially in line with that of other works[20, 22, 23, 21, 6]. Practically, in terms of computational running times, it means that, by either using $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ or $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ to produce a $\nu = 100$ dimensional feature representation, we can process 5-10 instances per second (a more detailed list of results is available in the supplementary material).

4.5. Analysis of action recognition performance

Despite [20, 22, 23, 30, 21, 24, 6] are applicable to a RBF kernel function (1), to the best of our knowledge there is no clear evidence of which method is more effective for classification. Indeed, despite all methods ensure scalability in the big data regime, there is no clear understanding about which method gives superior performance and, in general, how a good feature dimensionality ν should be chosen in practice. Here, we try to answer this question with

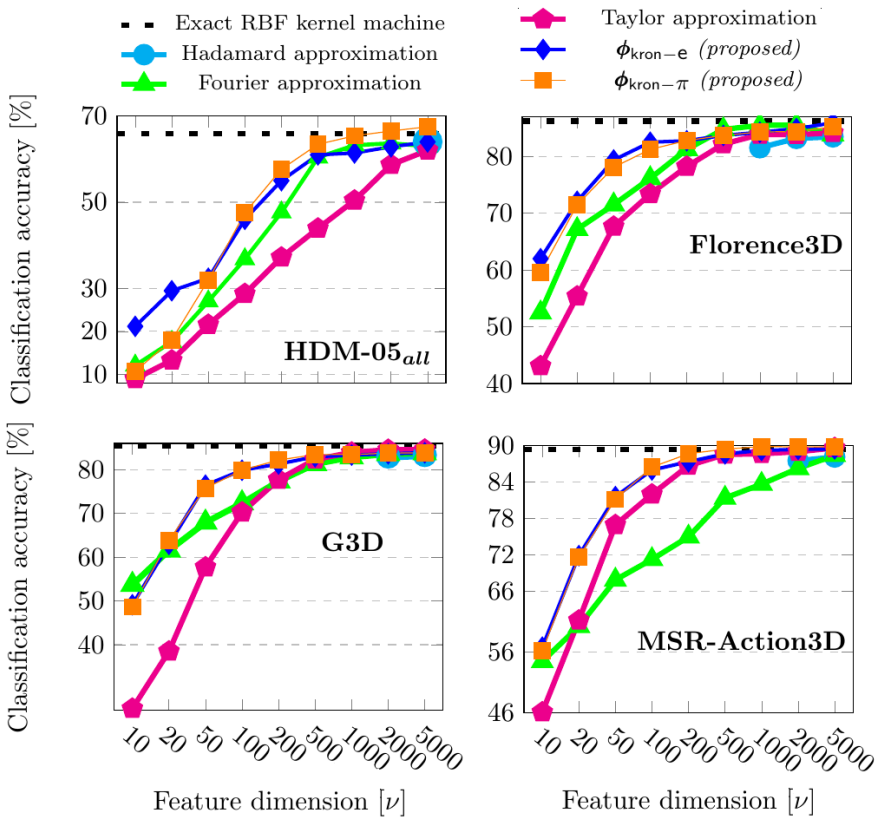

Figure 1: Mean classification accuracy plots averaged over 10 random sampling of the approximating schemes. Check the supplementary material, where the remaining plots are reported. Best viewed in colors.

a detailed validation on 3D action recognition benchmarks, while the feature dimensionality ν assumes one of the following values: 10, 20, 50, 100, 200, 500, 1000, 2000, 5000. We report the results of this analysis in Figures 1 for a few exemplar dataset: the overall trend is confirmed also in the remaining benchmarks as one can see in the supplementary material.

As common trend, we observe that accuracy grows while ν increases. This it theoretically reasonable because $\phi_{\text{kron}-\pi}$ and $\phi_{\text{kron}-\text{e}}$, as well as the alternative methods [20, 22, 23, 30, 21, 24, 6] are guaranteed to provide a better approximation for a bigger ν. For our method is the very same: since the bound on the variance is $O(1/\nu^3)$, when $\nu \to \infty$, we converge towards the expected value

of our approximation which is the exact kernel function.

As another piece of evidence for correctness of the approximations, we can notice that with a feature dimensionality $\nu \geq 1000$, the performance of each single method is close to the remaining ones and, globally, they are able to mimic the classification accuracy of an exact kernel machine: when $\nu > 500, 1000$, we observe a plateau of accuracies since all methods tend to approach the horizontal asymptote given by the exact kernel method (black dotted line).

However, we can observe an interesting pattern which is, in general, common to all datasets for the case $\nu < 200$: at low feature dimensionality (such as 10 or 20), the proposed approximations $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ are remarkably superior in performance with respect to all other competitors which are outperformed by margin. For instance, $+10\%$ on Florence3D for $\nu = 10$, $+14\%$ on MSR-Action3D for $\nu = 20$, $+9\%$ on G3D when $\nu = 50$ and more than $+10\%$ on HDM-05_{all} when $\nu = 100$. Such superiority can be explained by considering the fact that, in our paper, we explicitly provide an upper bound on the variance, as only few methods in the literature do. Consequently, our method is remarkably faster in recovering the original kernel function when increasing the dimensionality of the approximated feature map.

As anticipated, an interesting collateral result of our work consists in the possibility to compare the previously proposed methods [20, 22, 23, 30, 21] within a common benchmark in which we monitor the deviation in performance of the various approximations with respect to the exact kernel machine. In fact, despite [30] shows a solid performance which is always able to match the exact kernel machine and all the other competitors, such approach is limited by the impossibility to obtain a low-dimensional feature representation. Differently, the Fourier [20, 22, 23] and Taylor-based methods [21] show an oscillating performance where, frequently, one outperforms the other, even by margin. In this respect, the solidity of our methods, which is always top scoring, can be concretely appreciated as an advantage in terms of both compactness and superior classification performance.

5. Experimental results: state-of-the-art benchmarks in 3D action recognition

In this Section we will compare our proposed approximating schemes $\varphi_{\mathsf{kron}-\mathsf{e}}$ and $\varphi_{\mathsf{kron}-\pi}$ not only with previously proposed approximations [20, 22, 23, 30, 21], but also against state-of-the-art approaches for 3D action recognition from skeletal data. Additionally, we will provide the results obtained through our proposed perceptron heuristics $\boldsymbol{\phi}_{\mathrm{P}}$.

Before presenting the results, we will briefly discuss the methods involved in the comparison, both kernel methods and feature learning-based approaches.

Kernel methods. We compare against the Fisher vectors-based encoding of [56] and the Lie group representation [8] and related Lie algebra embedding [35] of rototranslations. We also compare against the combination of multiple non-linear RBF kernels (Ker-RP-RBF) [3], the sequence and dynamics compatibility kernels (SCK + DCK) [32] and Hankel matrices combined with either HMM (H-HMM) [33] or geodesic nearest neighbors method with class-protypes (H-prototypes) [34]. Also, we consider the nearest neighbor classification performed in [57] through a spatio-temporal Bayesian kernel similarity. Since our approach is covariance-based, we benchmark the temporal pyramid of covariance descriptors (t-COV-pyramid) of [31], Bregman-divergence [4] and the kernelized covariance operator (Ker-COV) [5]. Despite [18] applies a similar approximated-covariance paradigm, the published results only pertain to image classification. For completeness, we run the original code and applied it to 3D action recognition, denoting with rnd-logHS and QMC-AlogHS the approaches which exploit either random sampling or Quasi-Monte Carlo integration.

Feature learning approaches. We compete against the following recurrent architectures: the RNN fed on the raw joints data (J-RNN) [9] with its body part-aware variant [43] and we consider Long-Short Term Memory units fed by either raw joints (J-LSTM) [9] and its improvements J-LSTM-a [10] and J-LSTM^2-a [11], which adopt either a shallow or a deep attention module, respectively. We compare against the ensemble of deep models given by RNN-tree

	Florence3D*	UTKinect	MSR-Action3D*	MSR-Action3D
SCK [32]	92.98	96.1	90.72	93.5
DCK [32]	93.03	97.5	86.30	91.7
SCK+DCK [32]	95.23	98.2	91.45	94.0
$\boldsymbol{\phi}_{\mathrm{P}}$ *(proposed)*	***97.25***	***98.3***	***96.30***	***97.4***

Table 1: Classification accuracies [%] of $\boldsymbol{\phi}_{\mathrm{P}}$ against [32]. Best results are bold and underlined, the symbol * indicates that we used the alternative training/testing split adopted in [32].

[44] and TSLSTM [45]

We consider the architectures proposed in [13] and [14] which embed a structured input data matrix within a deep net: [13] trains a deep neural network on top of covariance matrices (SPD-Net) and [14] trains on top of rotation matrices. We also compete against LieNet-3B, the 3 blocks configuration that is superior to other investigated in [14].

Also, we benchmark our approach against a few other methods which computes dynamic images (DI), image-like data structures from the joint data to encode the kinematics, and exploit them to train a convolutional neural network. Namely, we consider the J-DI$_E$-CNN [15] that exploits the Euclidean distance function between joints, J-DI$_\theta$-CNN [16] that extract DI from rototranslational representations and J-DI$_v$-CNN [12] that does the same from velocities, approximated with finite differences.

At the same time, we report the best performance obtained from Figures 1 related to the Hadamard- [30], Fourier- [20, 22, 23] and Taylor-based approximations [21], that we indicate with H-approx, F-approx and T-approx, respectively. Ancillary, we also compare with our proposed approximated feature maps $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$.

The results are reported in Table 2, except for the comparison $\boldsymbol{\phi}_{\mathrm{P}}$ versus [32] which is presented in Table 1 due to the different experimental protocol adopted from [32].

	Florence3D	MSR-*pairs*	G3D
H-approx [30]	85.5	72.8	83.8
F-approx [20][22][23]	83.4	72.2	83.4
T-approx [21]	84.2	73.6	**84.6**
$\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ *(proposed)*	***84.9***	*73.1*	*83.9*
$\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ *(proposed)*	*84.3*	***73.7***	*83.8*
rnd-LogHS [18]	88.1	79.4	87.8
QMC-logHS [18]	88.5	79.5	89.5
J-diff-DI-CNN [12]	–	90.3	–
LieNet-3B [14]	–	–	89.1
Lie Group [8]	90.7	91.4	91.1
Lie Algebra [35]	**91.4**	94.7	90.9
$\boldsymbol{\phi}_{\mathrm{P}}$ *(proposed)*	*91.2*	***95.5***	***93.0***

	MSRC-Kinect12	HDM-05_{all}
H-approx [30]	92.4	63.7
F-approx [20][22][23]	92.2	64.0
T-approx [21]	92.8	62.0
$\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ *(proposed)*	*92.3*	*65.0*
$\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ *(proposed)*	***95.6***	***66.5***
t-COV-pyramid [31]	89.2	–
Bregman-div [4]	89.9	58.2
Ker-RP-RBF [3]	92.3	66.2
J-DI$_E$-CNN [15]	93.1	–
Ker-COV [5]	95.0	–
rnd-logHS [18]	97.1	58.1
QMC-logHS [18]	96.2	60.2
SPD-net [13]	–	61.4
$\boldsymbol{\phi}_{\mathrm{P}}$ *(proposed)*	***98.5***	***72.0***

	NTU-×-*subject*	NTU-×-*view*
H-approx [30]	51.5	50.6
F-approx [20][22][23]	50.7	50.6
T-approx [21]	50.8	51.0
$\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ *(proposed)*	*50.7*	*49.4*
$\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ *(proposed)*	***54.0***	***54.1***
Fisher Vectors [56]	38.6	41.4
Lie Group [8]	50.1	52.8
J-RNN [9]	56.3	64.0
J-RNN-parts [43]	59.1	64.1
LieNet-3B [14]	61.4	67.0
J-LSTM [9]	60.7	67.3
J-LSTM-a [10]	69.2	77.7
J-DI$_E$-CNN [15]	73.4	75.2
J-LSTM^2-a [11]	74.4	82.8
TS-LSTM [45]	74.6	81.3
RNN-tree [44]	74.6	83.2
J-DI$_\theta$-CNN [16]	76.2	82.3
J-DI$_v$-CNN [12]	**79.6**	**84.8**
$\boldsymbol{\phi}_{\mathrm{P}}$ *(proposed)*	*60.9*	*63.4*

	MSR-Action3D	HDM-05_{14}	UTKinect
H-approx [30]	88.4	89.2	83.9
F-approx [20][22][23]	88.2	88.6	84.0
T-approx [21]	89.6	88.9	84.0
$\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ *(proposed)*	*89.5*	*89.6*	***84.4***
$\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ *(proposed)*	***89.9***	***89.9***	*84.0*
t-COV-pyramid [31]	74.0	91.5	–
H-HMM [33]	89.0	–	86.8
rnd-logHS [18]	91.5	88.5	89.7
QMC-logHS [18]	90.6	85.4	91.3
H-prototypes [34]	94.7	86.3	**100**
TS-LSTM [45]	–	–	97.0
J-LSTM [10]	94.8	–	97.0
Ker-RP-RBF [3]	96.9	96.8	–
ST-BNN [57]	94.8	–	98.0
Ker-COV [5]	96.8	98.1	–
$\boldsymbol{\phi}_{\mathrm{P}}$ *(proposed)*	***97.4***	***99.1***	*98.3*

Table 2: Classification accuracies [%] for 3D action recognition. For each table, the top part present the performance achieved by $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ against other alternative approximating schemes [20, 22, 23, 30, 21]: within this class of methods, the best accuracy is highlighted in bold. At the same time, in the bottom part of each table, $\boldsymbol{\phi}_{\mathrm{P}}$ is compared against state-of-the-art approaches and, among them, the best performance is marked by bold and underlined. All the performance achieved by methods proposed in this paper ($\boldsymbol{\phi}_{\mathsf{kron}-\pi}$, $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ and $\boldsymbol{\phi}_{\mathrm{P}}$) are in italic.

5.1. Discussion

In Section 4.5, we primarily compared the proposed maps $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ among alternative approximating schemes for different ν values. In Table 2, instead, we can monitor the performance of this class of approximating methods in absolute terms while comparing among methods which have been explicitly designed for 3D action recognition.

First of all, we can notice that our approximation is able to improve upon existing approximating schemes [20, 22, 23, 30, 21] which apply general theoretical frameworks (Bochners Theorem and Fourier analysis in [20, 22, 23, 30] and general properties of a broad family of kernel functions in [21]) to carry out kernel approximation in a top-down fashion. Differently, by means of our bottom-up approach which is directly tailored on the specific kernel function that we care of approximating, we ultimately maximize performance and compactness.

In certain cases, $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ are better than methods which have been explicitly designed for action recognition. It is important to remind here that, in theory, those approximations hold for any type of $d \times d$ data input. For instance, on MSR-Action3D, $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ improves [31] by about $+15\%$ and, on the NTU RGB+D dataset with the cross-subject protocol, the performance of [8] and [56] is improved by $+4\%$ and $+16\%$, respectively. Eventually, on the NTU-$\times$-*subject*, the performance scored by $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ is almost on par with respect to the deep recurrent neural networks J-RNN and J-RNN-parts [43, 9]. Furthermore, in the middle data regime of MSRC-Kinect12 and HDM-05$_{all}$, $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ (and, in general, all the other approximated feature maps hereby considered) are scoring better than [31, 4, 3, 15] on MSRC-Kinect12. For what concerns HDM$_{all}$, $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ is even able to beat by 5% the state-of-the-art deep learning method SPD-net [13]. Such trend can be motivated by the fact that, in the middle data regime ($\sim 10^4$ samples), the data instances are sufficiently rich to allow learning satisfactory decision boundaries in a max margin sense, whereas they are not enough to effectively train deep models due to their over-parametrization.

While moving from either $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}$ or $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ to $\boldsymbol{\phi}_{\mathrm{P}}$, we *always* observe a

growth in performance, the latter being about +2% in the worst case and about +22% in the best one. Precisely, in the small data regime, we improved previously published state-of-the-art classification results by +0.5% on MSR-Action3D, +0.8% on MSR-*pairs*, +1% on HDM-05_{14} and by +2.1% on G3D.

At the same time, the gap in accuracy between $\boldsymbol{\phi}_{\mathrm{P}}$ and $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}, \boldsymbol{\phi}_{\mathsf{kron}-\pi}$ grows as the size of the dataset increases: such correlation is clearly a matter of the well known fact that feature learning benefits from more data. Again, the middle data regime seems the ideal operative setting for $\boldsymbol{\phi}_{\mathrm{P}}$, since, to the best of our knowledge, the previously published state-of-the-art performance on MSRC-Kinect12 by +2.3% (with respect to Ker-RP-RBF [3]) and by +10.6% on HDM-05_{all}. All in all, we explain such trend by observing that $\boldsymbol{\phi}_{\mathrm{P}}$ by rinterpreting $\boldsymbol{\phi}_{\mathrm{P}}$ as a sort of linearized version of the schemes $\boldsymbol{\phi}_{\mathsf{kron}-\mathsf{e}}, \boldsymbol{\phi}_{\mathsf{kron}-\pi}$, where the random sampling of weights is replaced by back-propagation learning.

On the NTU RGB+D experiments, $\boldsymbol{\phi}_{\mathrm{P}}$ improves (by margin) Fisher vectors [56], Lie group representation [8] as well as the deep J-RNN and J-RNN-parts on the NTU-$\times$-*subjects*. However, when comparing with the performance of LSTM- and CNN-based methods, $\boldsymbol{\phi}_{\mathrm{P}}$ shows a suboptimal performance. This trend can be justified in two ways.

On the one hand, we are applying a shallow architecture with just one hidden layer while, for instance, J-DI_v-CNN and JCNN2 uses multiple deep convnets in parallel and J-LSTM^2-a conditions a deep LSTM on the output of another deep LSTM network.

On the other hand, all LSTM-based methods and J-DI_v-CNN access all the raw coordinates for each given timestamps: therefore, since we train the architecture of [7] on covariance matrices, we can say that we are using much less data that are reduced by a factor of approximatively $1/100$, being 100 the typical temporal length for the sequences on the NTU RGB+D dataset.

Despite the previous two points are a drawback in terms of classification accuracies, they results in the following operative advantages.

First, since the architecture of [7] is shallow, there is no need for GPU acceleration neither for inference (which is nevertheless real-time), nor for the training

stage (which, even on CPU, only lasts less than one hour, as opposed to one day, for instance, for the LSTM networks to be trained [11]). Therefore, our system achieves a clear portability for deployment in real-world applications that requires real-time and scalable recognition capabilities.

Second, our representation is very compact: the experiments reported in Table 1, we are able to always overcome SCK and DCK in performance, even using a feature representation which is about 100 times more compact. Even on the NTU RGB+D dataset, we train the coefficients of the support vectors on top of the hidden representation of [7] where its size is fixed to 2^8. Having only two sets of weighted elements is a very favorable operative condition as opposed to stacking several convolutional layers [15, 16, 12] or allocating high-dimensional tensors for back-propagating through times and train the architectures of [43, 9, 10, 11].

This certifies in empirical terms the benefits of learning instead of sampling weights since although being a simple heuristics, the improvements in performance justifies the soundness of our proposed $\boldsymbol{\phi}_{\mathrm{P}}$.

6. Conclusions & future work

This paper presented $\boldsymbol{\phi}_{\mathsf{kron}-\pi}$ a novel approximation scheme for the RBF kernel function, which was shown to be superior to other approximations [20, 22, 23, 30, 21] in terms of better variance bound and classification accuracy, being the computational cost almost equal.

In a broad experimental evaluation among state-of-the-art competitors over publicly available action recognition datasets, our method generally assesses its superiority in terms of classification accuracy. Such favorable performance is also obtained by means of a very compact model, that is characterized by a simple & fast training procedure, especially if compared to deep learning methods.

We opted for a trade-off (ρ to be Geometric distributed of parameter $\theta = 0.9$) which slightly penalizes variance in favor of computational efficiency, but with practically no impact on classification performance.

As future work, we aim at investigating a couple of topics. In theoretical

terms, We will try to devise an improved bound on the variance, while, from the application standpoint, we want to apply our approximation to other computer vision tasks, such as object categorization.

References

[1] M. Vrigkas, C. Nikou, I. A. Kakadiaris, A review of human activity recognition methods, Frontiers in Robotics AI 2 (2015) 28.

[2] J. D. J. Shotton, A. W. Fitzgibbon, Human body pose estimation, uS Patent 9,262,673 (Feb. 16 2016).

[3] L. Wang, J. Zhang, L. Zhou, C. Tang, W. Li, Beyond covariance: Feature representation with nonlinear kernel matrices, in: International Conference on Computer Vision (ICCV), 2015.

[4] M. Harandi, M. Salzmann, F. Porikli, Bregman divergences for infinite dimensional covariance matrices, in: Computer Vision and Pattern Recognition (CVPR), 2014.

[5] J. Cavazza, A. Zunino, M. San Biagio, V. Murino, Kernelized covariance for action recognition, in: International Conference on Pattern Recognition (ICPR), 2016.

[6] J. Cavazza, P. Morerio, V. Murino, A compact kernel approximation for 3d action recognition, in: International Conference on Image Analysis and Processing (ICIAP), 2017.

[7] J. Cavazza, P. Morerio, V. Murino, When kernel methods meet feature learning: Log-covariance network for action recognition, in: Computer Vision and Pattern Recognition (CVPR) workshops, 2017.

[8] R. Vemulapalli, F. Arrate, R. Chellappa, Human action recognition by representing 3d skeletons as points in a lie group, in: Computer Vision and Pattern Recognition (CVPR), 2014.

[9] A. Shahroudy, J. Liu, T.-T. Ng, G. Wang, NTU RGB+D: A large scale dataset for 3D human activity analysis, in: Computer Vision and Pattern Recognition (CVPR), 2016.

[10] J. Liu, A. Shahroudy, D. Xu, G. Wang, Spatio-temporal LSTM with trust gates for 3D human action recognition, in: European Conference on Computer Vision (ECCV), 2016.

[11] J. Liu, G. Wang, P. Hu, L.-Y. Duan, A. C. Kot, Global context-aware attention lstm networks for 3d action recognition, in: Computer Vision and Pattern Recognition (CVPR), 2017.

[12] Q. Ke, M. Bennamoun, S. An, F. Sohel, F. Boussaid, A new representation of skeleton sequences for 3d action recognition, in: Computer Vision and Pattern Recognition (CVPR), 2017.

[13] Z. Huang, L. V. Gool, A riemannian network for SPD matrix learning, in: AAAI Conference on Artificial Intelligence, 2017.

[14] Z. Huang, C. Wan, T. Probst, L. V. Gool, Deep learning on lie groups for skeleton-based action recognition, in: arXiv:1612.05877, 2016.

[15] P. Wang, Z. Li, Y. Hou, W. Li, Action recognition based on joint trajectory maps using convolutional neural networks, in: ACM Multimedia, 2016.

[16] C. Li, Y. Hou, P. Wang, W. Li, Joint distance maps based action recognition with convolutional neural network, in: IEEE Signal Processing Letters, 2017.

[17] M. Ha Quang, M. San Biagio, V. Murino, Log-Hilbert-Schmidt metric between positive definite operators on Hilbert spaces, in: Neural Information Processing Systems (NIPS), 2014.

[18] H. Q. Minh, M. San Biagio, L. Bazzani, V. Murino, Approximate log-Hilbert-Schmidt distances between covariance operators for image classification, in: Computer Vision and Pattern Recognition (CVPR), 2016.

[19] B. Scholkopf, A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, USA, 2002.

[20] A. Rahimi, B. Recth, Random features for large-scale kernel machines, in: Neural Information Processing Systems (NIPS), 2007.

[21] P. Kar, H. Karnick, Random feature maps for dot product kernels, in: Artificial Intelligence and Statistics (AISTATS), 2013.

[22] S. Vempati, A. Vedaldi, A. Zisserman, C. V. Jawahar, Generalized rbf feature maps for efficient detection, in: British Machine Vision Conference (BMVC), 2010.

[23] A. Vedaldi, A. Zisserman, Efficient additive kernels via explicit feature maps, Transactions on Pattern Analysis and Machine Intelligence 34 (3).

[24] An approximation of the gaussian rbf kernel for efficient classification with svms, Pattern Recognition Letters 84 (2016) 107 – 113.

[25] V. Arsigny, P. Fillard, X. Pennec, N. Ayache, Geometric means in a novel vector space structure on symmetric positive-definite matrices, SIAM Journal on Matrix Analysis and Applications 29 (1) (2007) 328–347.

[26] D. Tosato, M. Spera, M. Cristani, V. Murino, Characterizing humans on riemannian manifolds, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (8) (2013) 1972–1984.

[27] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: A library for large linear classification, Journal of Machine Learning Research 9 (2008) 1871–1874.

[28] M. C. Lee, W. L. Chiang, C. J. Lin, Fast matrix-vector multiplications for large-scale logistic regression on shared-memory systems, in: International Conference on Data Mining (ICDM), 2015.

[29] W.-L. Chiang, M.-C. Lee, C.-J. Lin, Parallel dual coordinate descent method for large-scale linear classification in multi-core environments, in: ACM Conference on Knowelgde Discovery and Data Mining, 2016.

[30] Q. Le, T. Sarlos, A. Smola, Fastfood - approximating kernel expansion in loglinear time, in: International Conference on Machine Learning (ICML), 2013.

[31] M. Hussein, M. Torki, M. Gowayyed, M. El-Saban, Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations, in: International Joint Conference on Artificial Intelligence, 2013.

[32] P. Koniusz, A. Cherian, F. Porikli, Tensor representation via kernel linearization for action recognition from 3d skeletons, in: European Conference on Computer Vision (ECCV), 2016.

[33] L. Lo Presti, M. La Cascia, S. Sclaroff, O. Camps, Gesture modeling by Hanklet-based hidden Markov model, in: Asian Conference on Computer Vision (ACCV), 2014.

[34] X. Zhang, Y. Wang, M. Gou, M. Sznaier, O. Camps, Efficient temporal sequence comparison and classification using gram matrix embeddings on a riemannian manifold, in: Computer Vision and Pattern Recognition (CVPR), 2016.

[35] R. Vemulapalli, R. Chellapa, Rolling rotations for recognizing human actions from 3d skeletal data, in: Computer Vision and Pattern Recognition (CVPR), 2016.

[36] Smoothly approximated support vector domain description, Pattern Recognition 49 (2016) 55 – 64.

[37] Unsupervised feature selection based on maximum information and minimum redundancy for hyperspectral images, Pattern Recognition 51 (2016) 295 – 309.

[38] Nystrm-based approximate kernel subspace learning, Pattern Recognition 57 (2016) 190 – 197.

[39] Efficient clustering on riemannian manifolds: A kernelised random projection approach, Pattern Recognition 51 (2016) 333 – 345.

[40] New hermite orthogonal polynomial kernel and combined kernels in support vector machine classifier, Pattern Recognition 60 (2016) 921 – 935.

[41] Parsimonious mahalanobis kernel for the classification of high dimensional data, Pattern Recognition 46 (3) (2013) 845 – 854.

[42] Multiple feature kernel hashing for large-scale visual search, Pattern Recognition 47 (2) (2014) 748 – 757.

[43] Y. Du, W. Wang, L. Wang, Hierarchical recurrent neural network for skeleton based action recognition, in: Computer Vision and Pattern Recognition (CVPR), 2015.

[44] W. Li, L. Wen, M.-C. Chang, S. N. Lim, S. Lyu, Adaptive RNN tree for large-scale human action recognition, in: International Conference on Computer Vision (ICCV), 2017.

[45] I. Lee, D. Kim, S. Kang, S. Lee, Ensemble deep learning for skeleton-based action recognition using temporal sliding LSTM networks, in: International Conference on Computer Vision (ICCV), 2017.

[46] W. Rudin, Real and Complex Analysis, 3rd Ed., McGraw-Hill, Inc., New York, NY, USA, 1987.

[47] A. Shashua, Introduction to machine learning, in: arXiv:0904.3664v1, 2008.

[48] L. Xia, C.-C. Chen, J. Aggarwal, View invariant human action recognition using histograms of 3D joints, in: Computer Vision and Pattern Recognition (CVPR) workshops, 2012.

[49] L. Seidenari, V. Varano, S. Berretti, A. D. Bimbo, P. Pala, Recognizing actions from depth cameras as weakly aligned multi-part bag-of-poses, in: Computer Vision and Pattern Recognition (CVPR) workshops, 2013.

[50] O. Oreifej, Z. Liu., HON4D: Histogram of oriented 4D normals for activity recognition from depth sequences, in: Computer Vision and Pattern Recognition (CVPR), 2013.

[51] W. Li, Z. Zhang, Z. Liu, Action recognition based on a bag of 3d points, in: Computer Vision and Pattern Recognition (CVPR) workshops, 2010.

[52] V. Bloom, D. Makris, V. Argyriou, G3D: A gaming action dataset and real time action recognition evaluation framework, in: Computer Vision and Pattern Recognition (CVPR), 2012.

[53] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, A. Weber, HDM-05 doc., in: Tech. Rep., 2007.

[54] S. Fothergill, H. M. Mentis, P. Kohli, S. Nowozin, Instructing people for training gestural interactive systems, in: ACM Conference on Computer-Human Interaction, 2012.

[55] K. Cho, X. Chen, Classifying and visualizing motion capture sequences using deep neural networks, CoRR 1306.3874.

[56] G. Evangelidis, G. Singh, R. Horaud, Skeletal quads: Human action recognition using joint quadruples, in: International Conference on Pattern Recognition (ICPR), 2014.

[57] J. Y. Junwu Weng, Chaoqun Weng, Spatio-temporal naive-bayes nearest-neighbor for skeleton-based action recognition, in: Computer Vision and Pattern Recognition (CVPR), 2017.