Spatio-Temporal Naive-Bayes Nearest-Neighbor (ST-NBNN) for Skeleton-Based Action Recognition

Junwu Weng Chaoqun Weng Junsong Yuan
School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore 639798
{WE0001WU, WENG0018}@e.ntu.edu.sg, jsyuan@ntu.edu.sg

Abstract

Motivated by previous success of using non-parametric methods to recognize objects, e.g., NBNN [2], we extend it to recognize actions using skeletons. Each 3D action is presented by a sequence of 3D poses. Similar to NBNN, our proposed Spatio-Temporal-NBNN applies stage-to-class distance to classify actions. However, ST-NBNN takes the spatio-temporal structure of 3D actions into consideration and relaxes the Naive Bayes assumption of NBNN. Specifically, ST-NBNN adopts bilinear classifiers [19] to identify both key temporal stages as well as spatial joints for action classification. Although only using a linear classifier, experiments on three benchmark datasets show that by combining the strength of both non-parametric and parametric models, ST-NBNN can achieve competitive performance compared with state-of-the-art results using sophisticated models such as deep learning. Moreover, by identifying key skeleton joints and temporal stages for each action class, our ST-NBNN can capture the essential spatio-temporal patterns that play key roles of recognizing actions, which is not always achievable by using end-to-end models.

1. Introduction

Thanks to the development of commodity depth cameras, skeleton-based action recognition has drawn considerable attention in the computer vision community recently. To date, the leading 3D action classifiers are learning-based classifiers, in particular deep learning based methods (e.g., [5, 24, 35, 21, 14, 13]), which have shown promising results in benchmark datasets.

Despite the great progress of using learning-based methods for 3D action recognition, on the other hand, non-parametric classifiers, which make classification decision directly on the data and require no learning/training of parameters, are not well explored for 3D action recognition.

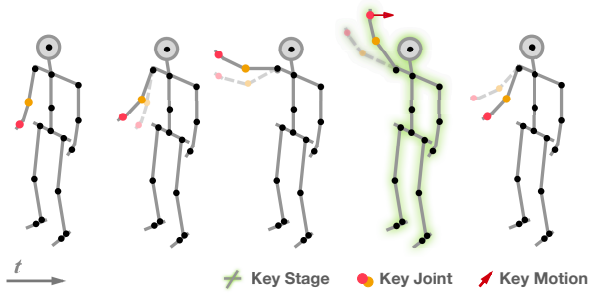

Figure 1. An Illustration of Key Stage, Joints, and Motion for the action of waving right hand action.

Interestingly, for image recognition, non-parametric methods, e.g., *Naive-Bayes Nearest-Neighbor* (NBNN) [2], have shown an impressive performance of classifying images of local visual primitives by using image-to-class distance. Motivated by the previous success of NBNN, in this work, we extend it to recognize actions.

Two observations motivate our exploration of using NBNN for skeleton-based action recognition: (1) similar to images that are composed by local visual primitives, actions are composed by spatio-temporal primitives too, e.g., each action instance is a collection of skeleton poses and each pose is further a collection of spatial joints. We can easily apply *primitive-to-class* distance to perform action recognition, which can generalize according to NBNN. (2) Compared with images and videos which are composed of millions or billions of pixels, the skeleton is composed by only tens of joints which is thus of much less complexity than images and videos. We argue that instead of relying on a sophisticated end-to-end model, a simple non-parametric model may still obtain promising results for such a light-weight problem.

In this work, we propose *Spatio-Temporal Naive-Bayes Nearest-Neighbor* (ST-NBNN), a new variation of NBNN, to classify 3D actions. Each 3D action instance is represented by a collection of temporal stages composed by 3D

poses, and each pose in stages is presented by a collection of spatial joints. Following NBNN, our ST-NBNN applies stage-to-class distance to classify actions. It can well handle the length variations of 3D actions and also the large intra-class variations. However, not every temporal stage and spatial joints are of equal importance to the recognition of the action. Thus it is of great importance to identify the key stages and skeleton joints that matter for the recognition. To extend NBNN for 3D action classification, our ST-NBNN considers the spatio-temporal structure of actions. Instead of simply summing up all stage-to-class distances with a Naive-Bayes assumption, we present these distances as a spatio-temporal matrix of NN distances which represents the action instance. ST-NBNN further adopts a bilinear classifier [19] to identify key joints and stages and classify the spatio-temporal matrix of NN distances. Our proposed formulation can be iteratively optimized to learn the linear classification weight for both spatial joints and temporal stages.

We use Fig. 1 to illustrate the idea of using key spatial joints and temporal stages for action recognition. When performing right hand waving action, only the right hand and arm (key joints) are activated. And when observing the timing (key stage) at which the right hand and arm raise up and move horizontally towards left, we can claim that waving right hand action is performing. Such a spatio-temporal pattern described by key temporal stages and spatial joints is critical to identify action classes. The discovery of such patterns not only can improve recognition accuracy but also answer what composes such an action instance and why we make a recognition of it.

By using both stage-to-class distance and bilinear classifier [19], our proposed ST-NBNN combines the strengths of non-parametric model and parametric model. Although only using a linear classifier, experiments on three benchmark datasets show that ST-NBNN using raw skeleton features can already obtain very competitive performance compared with state-of-the-art end-to-end models which optimize for feature representations. Moreover, by identifying key temporal stages and spatial joints which play key roles of recognizing the action, our ST-NBNN can capture the essential spatio-temporal patterns for each action class, and provide a physical interpretation of the action behavior. Such a spatio-temporal pattern discovery and explicit interpretation, however, is not always available via end-to-end models, which mainly focus on achieving higher recognition accuracy instead of better interpreting patterns.

2. Related Work

Skeleton-Based Action Recognition

In recent years, skeleton-based action recognition problem attracts a lot of attention, and many learning-based methods [5, 24, 35, 21, 14, 13, 15] have been proposed. Due to the tremendous amount of these works, we only limit our review to the spatio-temporal modeling of skeleton-based action recognition.

The modeling in the spatial domain is mainly driven by the fact that an action is usually only characterized by the interactions or combinations of a subset of skeleton joints [35]. Two categories of approaches are often used to model the spatial pattern of actions: part-based model and sub-pose model. In the part-based model, a skeleton is partitioned into several groups, and the joints in each group are skeletal neighbors of each other. In HBRNN [5], skeletons are decomposed into five parts, two arms, two legs, and one torso, and a hierarchical recurrent neural network is build to model the relationship among these parts. Similarly, in [21] a part-aware LSTM is proposed to construct the relationship between body parts. In sub-pose model, the focus is mainly on the informative joints or their interactions. In SMIJ [18], the most informative joints are selected simply based on measures such as mean or variance of joint angle trajectories. The sequence of these informative joints is then used as the representation of actions. In Orderlet [33], interactions between joints are modeled by a few comparisons of joints' primitive feature, and in action recognition only a subset of joints is involved. On the temporal domain, graphical models [11, 30], temporal pyramid matching [28, 32], and dynamic time warping [20] are the common methods for temporal modeling. While in [27], sequential pattern mining method is used to model temporal structures of a set of key poses. Besides spatial modeling or temporal modeling, we also see efforts on spatio-temporal modeling. In [14], LSTM model is extended to spatio-temporal domain to analyze skeletons. Compared with these methods, our proposed ST-NBNN is able to discover key factors of an action on spatial and temporal domain simultaneously.

Naive-Bayes Nearest-Neighbor

NBNN [2] is first proposed for image classification. Two key factors help NBNN achieve remarkably good results. First, it avoids using vector quantization to code primitive features so that rare but discriminative descriptors count in classification. Second, it chooses 'image-to-class' distance rather than 'image-to-image' distance to make a decision so that it bears a good generalization ability. Due to its success, there are a few variations of the original NBNN algorithm. In [1] Behmo *et al.* parameterize NBNN to relax its restrictive assumption that all classes have similar densities in feature space. In the kernelized version of NBNN [22], the independent assumption is criticized by introducing kernel between image representations. Compared with NBNN kernel [22], we also break the independent assumption by incorporating the spatio-temporal structure of the action data. Following NBNN, we also see its

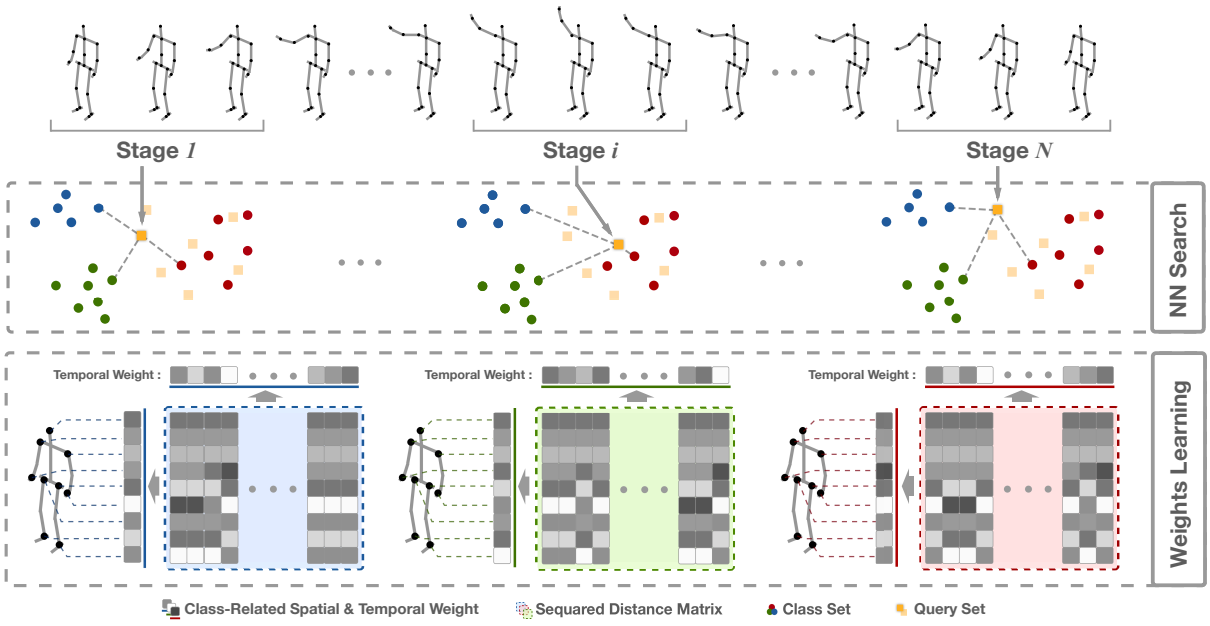

Figure 2. Overview of the Proposed Method. 1) An action video is uniformly divided into a fixed number of stages, and is represented by a set of stage-descriptors (orange query points); 2) Distances of stage-descriptors to action class sets (blue, green and red) are calculated by NN search; 3) Distances of stage-descriptors are gathered in temporal order to generate class-related squared distance matrices (marked by class-related dashed rectangular boxes); 4) Weights on the spatial (left side of the matrix) and the temporal (top of the matrix) domain are learnt to discover key factors of actions and predict action labels

application in video analysis. Yang *et al.* [31] use NBNN to classify 3D actions represented by dimension-reduced action descriptors, EigenJoint. In [34], NBNN is re-designed as *Naive-Bayes based Mutual Information Maximization* (NBMIM) to solve action detection problem. Negative samples are involved in nearest neighbor matching to improve the discriminative ability of descriptors. Recently, the combination of NBNN and CNN [10], as well as the effort to speeding up NN search [9], revive the possibility of NBNN's return in computer vision.

3. Proposed Method

In this section, we introduce how the proposed method predicts actions and discovers key joints and stages as well. The overview of our method is illustrated in Fig. 2. We first introduce a set of stage-descriptors to represent a 3D sequence (Sec. 3.1). Then NBNN [2] is used as a basic framework to classify actions (Sec. 3.2). Finally, the learning of spatial and temporal weights is introduced to discover key poses and spatial joints for action recognition (Sec. 3.3).

3.1. 3D Action Representation

In skeleton-based action recognition, each 3D action is a sequence of 3D poses, but different actions may have different temporal lengths. To provide a unified presentation, we partition each action into N temporal windows of equal length as shown in Fig. 2. Each one of the temporal windows is called a *temporal stage*, which is characterized by the 3D poses in its corresponding window. Assuming each 3D pose has J joints for its skeleton, for a temporal stage descriptor $\boldsymbol{x}$, the 3D pose in its jth frame is denoted as $\boldsymbol{p}_j \in \mathbb{R}^{3J}$, and the related velocity of that pose is denoted as $\boldsymbol{v}_j \in \mathbb{R}^{3J}$. Then the pose part $\boldsymbol{x}_p$ and the velocity part $\boldsymbol{x}_v$ of $\boldsymbol{x}$ is defined as below,

$$\begin{aligned} \boldsymbol{x}_p &= [(\boldsymbol{p}_1)^\intercal, \ \dots, (\boldsymbol{p}_l)^\intercal]^\intercal \\ \boldsymbol{x}_v &= [(\boldsymbol{v}_1)^\intercal, \ \dots, (\boldsymbol{v}_l)^\intercal]^\intercal \end{aligned} \tag{1}$$

Similar to [26], we also normalize $\boldsymbol{x}_p$ and $\boldsymbol{x}_v$ to have l_2 norm equals to one. In our experiment, it shows such a normalization can perform slightly better than using the original features. As we use both original 3D pose and its velocity to represent 3D actions, a temporal stage descriptor $\boldsymbol{x}$ of l frames is presented as:

$$\boldsymbol{x} = [(\boldsymbol{x}_p)^\intercal, (\boldsymbol{x}_v)^\intercal]^\intercal \tag{2}$$

Finally, a 3D action video is described by its N stage-descriptors $V = \{\boldsymbol{x}^i\}_{i=1}^N$.

3.2. NBNN

Given a query action video $V_q = \{\boldsymbol{x}^i\}_{i=1}^N$, the goal is to find which class $c \in \{1, 2, ..., C\}$ the video V_q belongs

to. NBNN follows maximum a posteriori (MAP) rule for classification. When assuming equal prior $p(c) = \frac{1}{C}$, the predicted class is:

$$c^* = \arg\max_c p(c|V_q) = \arg\max_c p(V_q|c) \tag{3}$$

With the Naive-Bayes assumption (data samples are independent with each other), Eq. 3 can be written as,

$$\begin{aligned} c^* &= \arg\max_c p(V_q|c) \\ &= \arg\max_c p(\boldsymbol{x}^1, ..., \boldsymbol{x}^N|c) \\ &= \arg\max_c \prod_{i=1}^{N} p(\boldsymbol{x}^i|c) \end{aligned} \tag{4}$$

Based on the analysis of [2], the probability density of each primitive $\boldsymbol{x}$ in a class c, namely $p(\boldsymbol{x}|c)$, can be estimated according to the distance between $\boldsymbol{x}$ and the nearest neighbor of $\boldsymbol{x}$ in class c, and Eq. 4 is then re-written as

$$c^* = \arg\min_c \sum_{i=1}^{N} \left\| \boldsymbol{x}^i - NN_c(\boldsymbol{x}^i) \right\|^2 \tag{5}$$

where $NN_c(\boldsymbol{x}^i)$ is the nearest neighbor of $\boldsymbol{x}^i$ in class c.

As a non-parametric model, NBNN has no training phase. For a query 3D action $V_q = \{\boldsymbol{x}^i\}_{i=1}^N$, each of its temporal stages will match against C classes separately by finding the best matched temporal stage, i.e., nearest neighbor, in that class. The better the match, i.e., the smaller the NN distance, the stronger this temporal stage will vote for that class c. Otherwise, if a temporal stage cannot find a good match in that class, it will not provide a strong vote to that class. As V_q has in total N temporal stages, the final decision is the summation over all of the N votes towards to C classes, as explained in Eq. 5.

3.3. Spatio-Temporal NBNN

For a specific action, usually only a subset of joints are activated for action performing, and for different actions, their activated joints are different. Hence in action classification, we can only focus on the activated spatial joints, and at the same time suppress those that are not discriminative or easily affected due to occlusions or capturing noises. Meanwhile, for a set of temporal stages, not every stage is of equal importance neither. Depending on action class, a certain temporal stage can be more discriminative than others for classification. As illustrated in Fig. 2, the descriptor (shadowed orange query square) of stage i is more discriminative than the beginning one and the ending one.

To identify important spatial joints and temporal stages simultaneously, we propose to leverage a bilinear classifier [19] to learn spatio-temporal weights for both of them in the framework of NBNN. Although previous works have studied how to pick important spatial joints or temporal poses, not many works can address both of them simultaneously.

Spatio-temporal Matrix

Though we assume that stage descriptors are independent in NBNN, they actually depend on each other in a certain spatio-temporal structure. Therefore, to capture the spatio-temporal structure of 3D actions representation, we represent a 3D action from a set $V = \{\boldsymbol{x}^i\}_{i=1}^N$ to a matrix, as illustrated in Weights Learning block of Fig. 2. For a given video sample with N stages, its spatio-temporal matrix is defined as

$$\boldsymbol{X} = [\boldsymbol{x}^1, \ ... \ , \boldsymbol{x}^N] \tag{6}$$

Stage-descriptors of an action are re-organized column by column following the temporal order. We further define the nearest neighbor matrix of $\boldsymbol{X}$ in c as $\boldsymbol{X}_c^{NN} = [NN_c(\boldsymbol{x}^1), \ ... \ , NN_c(\boldsymbol{x}^N)]$, and the squared distance matrix to class c is defined as

$$\boldsymbol{X}_c = (\boldsymbol{X} - \boldsymbol{X}_c^{NN}) \odot (\boldsymbol{X} - \boldsymbol{X}_c^{NN}) \tag{7}$$

where $\odot$ is an element-wise product. $\boldsymbol{X}_c$ is regarded as a representation of $\boldsymbol{X}$ for class c, and it is a combination of element-wise stage-to-class distances of the testing sample. Summation of all the elements in $\boldsymbol{X}_c$ is equivalent to $\sum_{i=1}^N \left\| \boldsymbol{x}^i - NN_c(\boldsymbol{x}^i) \right\|^2$ in Eq. 5.

Considering elements of $\boldsymbol{X}_c$ bear different contributions to classification, NBNN should be parameterized to emphasize those discriminative ones. We can simply vectorize $\boldsymbol{X}_c$ as $\boldsymbol{\chi}_c$, and the NBNN decision function Eq. 5 is then redefined as $c^* = \arg\min_c \boldsymbol{w}^\intercal \boldsymbol{\chi}_c$, where the weight $\boldsymbol{w}$ can be learnt by linear SVM. However, as $\boldsymbol{X}_c$ is a large matrix, the number of weights to be determined is too many. This strategy is not only time consuming but also has the risk of over-fitting. Hence, a bilinear classifier is adopted in our formulation.

Based on the squared distance matrix in Eq. 7, the classification score of a query matrix $\boldsymbol{X}$ to class c is then determined by a bilinear function $f_c(\cdot)$, which is defined as

$$f_c(\boldsymbol{X}_c) = (\boldsymbol{u}_c^s)^\intercal \, \boldsymbol{X}_c \, \boldsymbol{u}_c^t \tag{8}$$

where $\boldsymbol{u}_c^s \in \mathbb{R}^M$ and $\boldsymbol{u}_c^t \in \mathbb{R}^N$ are the spatial and temporal weights of action class c. As a result, the classification becomes

$$c^* = \arg\min_c f_c(\boldsymbol{X}_c) \tag{9}$$

As can be seen from Eq. 8, the proposed method provides weights for both temporal stages and spatial joints. After a rearrangement, Eq. 9 can be represented as,

$$c^* = \arg\min_c \sum_{i=1}^{N} \boldsymbol{u}_c^t(i) \ \left\| (\boldsymbol{x}^i - NN_c(\boldsymbol{x}^i))^\intercal \sqrt{\boldsymbol{u}_c^s} \right\|^2 \tag{10}$$

where $\sqrt{\cdot}$ is an element-wise square-root of a vector. It is worth noting that, NBNN is a special case of ST-NBNN. When $\boldsymbol{u}_c^s$ and $\boldsymbol{u}_c^t$ are assigned to $\mathbf{1}$, Eq. 10 becomes NBNN in Eq. 5. ST-NBNN generalizes NBNN by breaking the Naive-Bayes rule. Instead of assuming each stage is independent, we introduce the spatio-temporal structure of 3D action into our framework.

Spatial and Temporal Weights Learning
Our objective function is similar to tensor SVM. Following the learning strategy of [3], we adopt the one-vs.-all strategy to classify actions. With empirical loss, the objective function of spatio-temporal weight learning is defined as

$$\begin{aligned} \min_{\boldsymbol{u}_t^s, \boldsymbol{u}_c^t} \quad & \frac{1}{2}\|\boldsymbol{u}_c^s(\boldsymbol{u}_c^t)^\intercal\|^2 + \lambda \sum_{i=1}^{K} \xi_i \\ s.t \quad & \sum\nolimits_{i=1}^{N} \boldsymbol{u}_c^t(i) = N, \quad \boldsymbol{u}_c^t \succeq \mathbf{0} \\ & \xi_i \geqslant max(0, 1 - c_i f_c(\boldsymbol{X}_c^i))^2 \\ & \xi_i \geqslant 0, \quad i = 1, ..., K \end{aligned} \tag{11}$$

in which K is the number of training video samples, and $c_i \in \{-1, 1\}$ is the action label of the corresponding sample. $\boldsymbol{X}_c^i$ is the ith training sample in class c. λ is a parameter for classification error penalty.

The reason why we set linear constraints to temporal weights but not to spatial weights is as following. For the spatial domain we do not know how many key joints will be involved (some joints do not have any contribution in recognition), and how important those activated joints compared with others, while on the temporal domain, each stage of an action counts in classification. Experiment results also show that linear constraints on the spatial domain do not bear any contribution to performance, but the temporal constraints do.

The optimization of Eq. 11 is regarded as an iterative process. There are two steps in each iterative round, 1) fix $\boldsymbol{u}_c^t$ and update $\boldsymbol{u}_c^s$, 2) fix $\boldsymbol{u}_c^s$ then update $\boldsymbol{u}_c^t$. $\boldsymbol{u}_c^t$ is initialized to $\mathbf{1}$.

Fix $\boldsymbol{u}_c^t$ and Update $\boldsymbol{u}_c^s$: With $\boldsymbol{u}_c^t$ fixed, Eq. 11 is treated as a l_2-regularized l_2 loss SVM problem shown below

$$\min_{\boldsymbol{u}_c^s} \quad \frac{1}{2}\beta_1\|\boldsymbol{u}_c^s\|^2 + \lambda \sum_{i=1}^{K} max(0, 1 - c_i f_c(\boldsymbol{X}_c^i))^2 \tag{12}$$

where $\beta_1 = \|\boldsymbol{u}_c^t\|^2$.

Fix $\boldsymbol{u}_c^s$ and Update $\boldsymbol{u}_c^t$: With updated $\boldsymbol{u}_c^s$, Eq. 11 is regarded as a convex optimization problem with linear constraints shown below

$$\begin{aligned} \min_{\boldsymbol{u}_c^t} \quad & \frac{1}{2}\beta_2\|\boldsymbol{u}_c^t\|^2 + \lambda \sum_{i=1}^{K} max(0, 1 - c_i f_c(\boldsymbol{X}_c^i))^2 \\ s.t \quad & \sum\nolimits_{i=1}^{N} \boldsymbol{u}_c^t(i) = N, \quad \boldsymbol{u}_c^t \succeq \mathbf{0} \end{aligned} \tag{13}$$

where $\beta_2 = \|\boldsymbol{u}_c^s\|^2$.

This optimization process is operated iteratively until the objective function Eq. 11 converges.

Method	AS1	AS2	AS3	Ave.
NBNN-∞	85.8	92.0	96.4	91.4
NBNN-15	86.8	92.0	96.4	91.7
NBNN+SVM	90.6	90.3	96.4	92.4
Lie Group [25]	**95.4**	83.9	**98.2**	92.5
SCK+DCK [8]	–	–	–	94.0
HBRNN [5]	93.3	94.6	95.5	94.5
ST-LSTM [14]	–	–	–	**94.8**
Graph-Based [29]	93.6	95.5	95.1	**94.8**
Ours	91.5	**95.6**	97.3	**94.8**

Table 1. Comparison of Results on MSR-Action3D (%)

4. Experiment

In this section, we experiment the proposed method on three 3D action datasets and compare its performance to existing methods. Implementation details are provided in Sec. 4.1. Comparison results on the MSR-Action3D dataset [12], the UTKinect dataset [30], and the Berkeley MHAD dataset [17] are provided and discussed in Sec. 4.2. Experiment results show that though ST-NBNN is simple, it is able to achieve state-of-the-arts performance of 3D action recognition and effectively discover key factors of actions.

4.1. Implementations

3D Action Representation. The one-vs.-all strategy is used in this method. To ensure the responses of linear functions $f_c(\cdot)$ are comparable with each other, each sample $\boldsymbol{X}_c^i$ is mean-centralized by $\mu^i = \sum_{c=1}^{C} sum(\boldsymbol{X}_c^i)/(C \times M \times N)$, where $sum(\cdot)$ sums up entries of the input matrix.

The setting of stage number is indicated in the Sec. 4.2. Considering the variation of action sequences' duration, stages defined in Sec. 3.1 may have overlaps with each other stages when given sequences are not too long.

To ensure that the representation introduced in Sec. 3.1 is location-invariant, each joint of the skeleton is centralized by subtracting coordinates of the hip joint.

Nearest Neighbor Search. To boost the nearest neighbor searching process, KD-tree implementation [16] is used in our method.

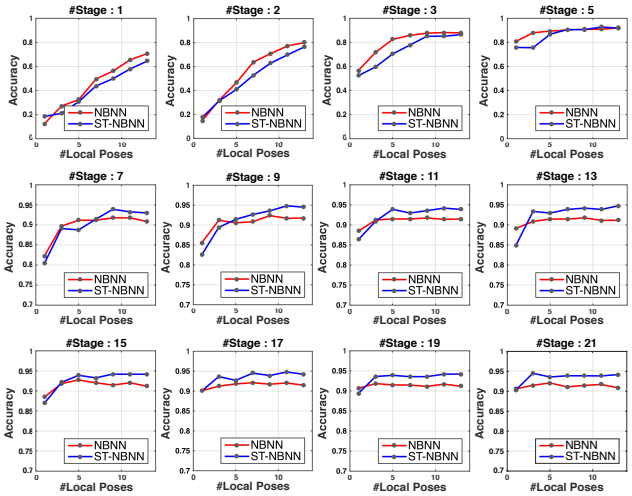

Figure 3. Parameter Sensitivity Analysis on MSR-Action Dataset. The x-axis indicates the chosen number of local poses. The subtitle indicates the chosen number of temporal stages.

Spatio-Temporal Weights Learning. The training matrices $\boldsymbol{X}_c$ are generated by a leave-one-video-out strategy, which means all the stage-descriptors of query training video are excluded in searching region when query stage-descriptors search for nearest neighbors.

In our optimization, $\boldsymbol{u}_c^s$ and $\boldsymbol{u}_c^t$ are learnt iteratively. To solve the SVM problem of Eq. 12, we use a SVM toolbox [4] implemented by Chang *et al.*, and to update $\boldsymbol{u}_c^t$, a convex optimization toolbox [6] is used.

4.2. Results and Analysis

MSR-Action3D

Here we use the evaluation protocol described in [12]. 20 actions are grouped into three subsets AS1, AS2 and AS3. Each of the action set contains eight actions. In this experiment, the number of local pose l is 10 and the number of stage N is 15. We compare ST-NBNN with three baseline methods and five state-of-the-art skeleton-based methods. The results are shown in Table. 1.

Three baseline methods included in the comparison are (1) NBNN without stage setting (NBNN-∞); (2) NBNN with 15 stages (NBNN-15); and (3) NBNN with weights learning by linear SVM (NBNN+SVM). In NBNN-∞, the number of stage N is determined by $N = P - l + 1$, where P is the length of the action video. Table. 1 shows that there is a slight improvement from NBNN-∞ to NBNN-15, which indicates that the informative stages do exist to help differentiate actions. In NBNN+SVM, we use linear SVM to learn weights for every element of $\boldsymbol{X}_c$. As we can see, learnt weights do help improve the performance, especially in AS1. However, it also causes over-fitting. In AS2, although the training accuracy has already achieved 100%, the testing accuracy drops. Besides, NBNN+SVM takes much more time to learn as it has much more parameters compared with the proposed method. The results in Table. 1 show that ST-NBNN bears the ability to achieve even better performance than NBNN+SVM. The comparisons with five state-of-the-art methods show that the proposed method achieves the current best performance. In addition, our method is better than the non-linear models [29, 8, 25], and the deep learning based methods [14, 5].

Method	Accuracy
NBNN-∞	95.5
NBNN-15	95.5
NBNN+SVM	94.0
Key-Motif [27]	93.5
Simplices [26]	96.5
ST-LSTM [14]	97.0
Lie Group [25]	97.1
Graph-Based [29]	97.4
SCK+DCK [8]	**98.2**
Ours	98.0

Table 2. Comparison of Results on UTKinect (%)

We also evaluate the two main parameters, N and l of ST-NBNN on this dataset. We range l from 1 to 13 and set N from 1 to 21. As Fig. 3 shows, ST-NBNN needs a sufficient number of stages to learn the spatio-temporal weights and obtain good performance. When the number of stages is larger than 11, ST-NBNN can still help improve the performance with only 3 poses in each stage. However, further increasing N and l will not improve the performance.

The key stages, joints discovered from MSR-Action 3D dataset are shown in Fig. 6. The results are quite interesting. In this dataset, *Hand Catch* and *Side Boxing* are not easy to be differentiated. However, ST-NBNN focuses on different motions of these two actions to differentiate them. For the *Hand Catch*, ST-NBNN mainly focuses on the x-direction motion of the key joint (right hand). While for the *Side Boxing*, ST-NBNN cares much about the y and z directions though x is the main direction of this action. A similar situation happens in *Forward Kick*. ST-NBNN does not choose the foot joints to focus on, but put more weight to the right hand since in *Forward Kick* performing, the right hand always move upward (y direction) concomitantly. Besides, as shown in Fig. 6 i) and j), the proposed method can also indicate different phases of actions. The two peaks of the temporal weight of *Pick Up and Throw* are related to the *Pick Up* and *Throw* two phases respectively.

UTKinect

We use the leave-one-out validation protocol described

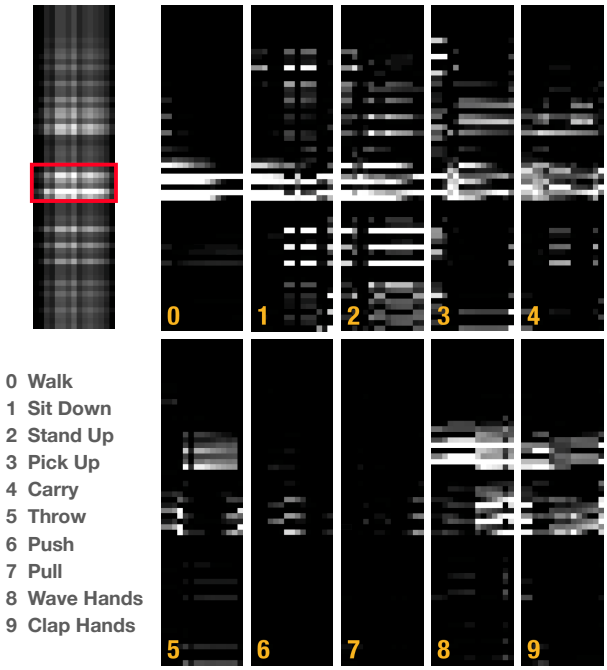

Figure 4. An Example of Spatio-Temporal Weight Matrix and Squared Distance Matrix (One pose feature of *Pull* Action). ST-Weight Matrix is on the top-left corner, and squared distance matrices are on the right side. Each matrix is 60 by 15. The related feature of discovered joints are marked by red box.

Method	Accuracy
NBNN-∞	88.0
NBNN-20	88.0
NBNN+SVM	**100.0**
SMIJ [18]	95.4
Meta-cognitive RBF Network [23]	97.6
Kapsouras *et al.* [7]	98.2
HBRNN [5]	**100.0**
ST-LSTM [14]	**100.0**
Ours	**100.0**

Table 3. Comparison of Results on Berkeley MHAD (%)

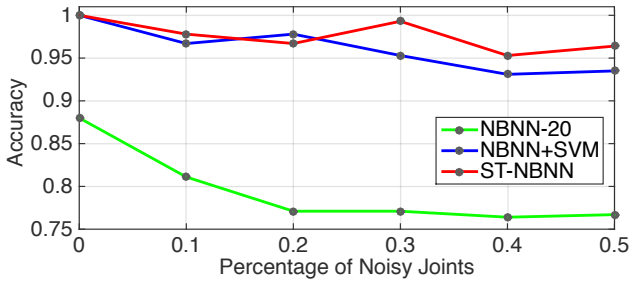

Figure 5. Influence of Noisy Joints on Accuracy of Berkeley MHAD Dataset

in [30] to evaluate our proposed method. Based on the description, there are 20 rounds of testing in our experiment. The parameters chosen for spatio-temporal weights learning are the same in each round. The number of local pose l is set to 3, and the number of stage N is 15.

Table. 2 shows that our method achieves 2.5% improvement from baseline NBNN, and NBNN+SVM is again inferior to ST-NBNN. With a large number of parameters, NBNN+SVM leads to over-fitting. The comparison with existing methods [27, 26, 25, 14, 29, 8] shows that ST-NBNN can achieve competitive performance.

In Fig. 4, we provide an example of learnt spatio-temporal weight matrix and squared distance matrices $\boldsymbol{X}_c$ from *Pull* action in UTKinect dataset. Due to the limitation of space, we only provide the first position feature of each stage and their related weights. Elements a_{ij} of the spatio-temporal weight matrix are determined by $a_{ij} = \boldsymbol{u}_c^s(i) \times \boldsymbol{u}_c^t(j), i = 1, ..., M, j = 1, ..., N$. The brighter the elements of matrix, the larger the value of the elements. The red-square-marked region is related to the x, y, z coordinates of right hand (joint *11* and *12*). In this dataset, subjects are required to perform *Pull* action by right hand, and the figure shows that ST-NBNN can discover it. From the right side, matrix 7 is the darkest one, which means that the testing sample has the smallest distance from action 7 (*Pull*), and therefore the testing sample belongs to *Pull*. The weight matrix selects the commonly most discriminative part of matrices, and it proves that our proposed method is able to discover key factors of actions.

Berkeley MHAD

We follow the experimental protocol described in [17] on this dataset. The sequences performed by the first seven subjects are for training while the ones performed by the rest subjects are for testing. Due to the high sampling rate, most of the data is redundant. We down-sample each sequence by picking one frame of each ten frames. Under this setting, the number of local pose l is 20, and the number of stage N is 20. We compare ST-NBNN with three baseline methods and five state-of-the-art skeleton-based methods. The results are shown in Table. 3.

Table. 3 shows that although the accuracy of NBNN is only 88%, we can achieve 100% accuracy with the help of weights learning. Besides, comparisons with previous works [18, 23, 7, 5, 14] show that our spatio-temporal weights learning method is able to effectively discover key factors of actions and achieve the best performance.

Considering that the skeletal data captured by motion capture system is more accurate than Kinect depth sensor, we evaluate the tolerance and robustness of ST-NBNN to random noise of skeleton data. We further add noise to

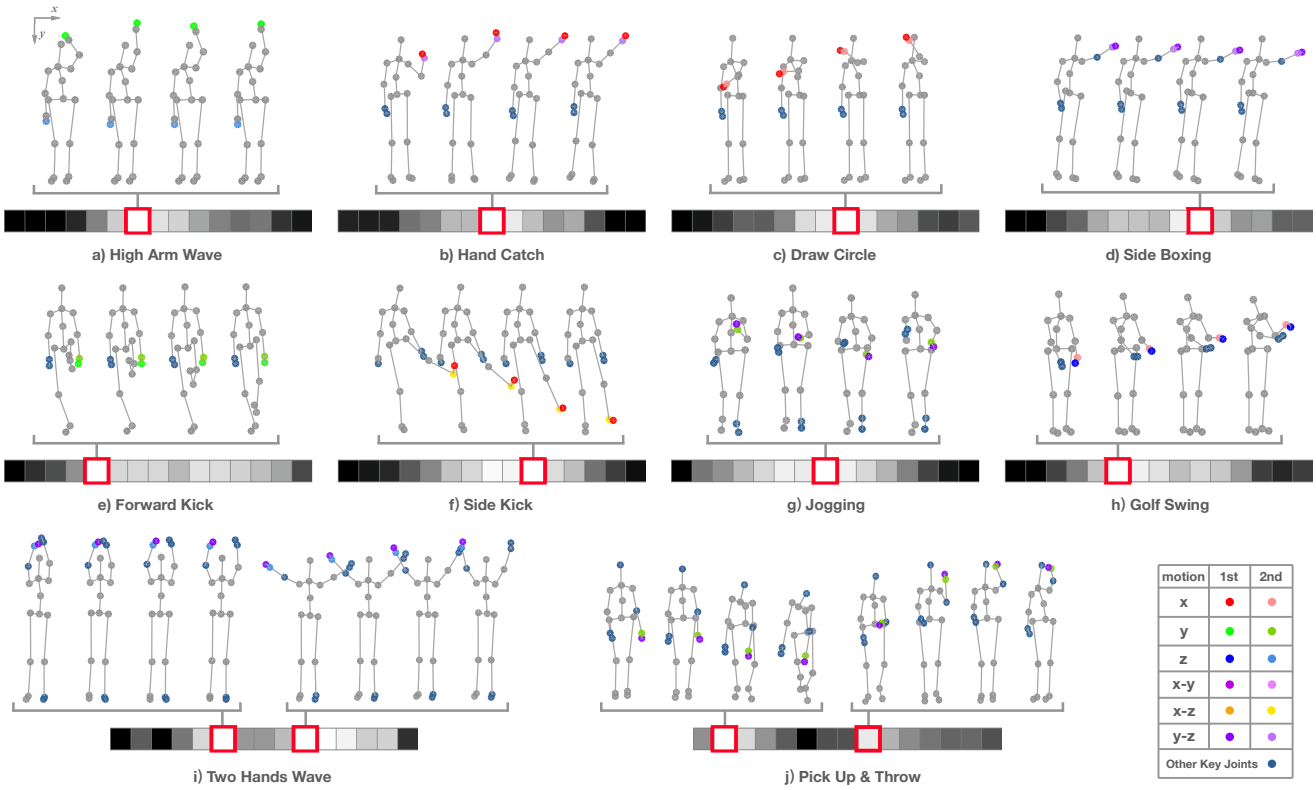

Figure 6. Key Stages and Key Joints with their Key Motions from MSR-Action3D. Colored joints are with weights larger than average weights. The most informative joints are marked by bright color, and the second most informative joints are marked by light color. The global key motions are indicated by different colors. For example, the key motion directing in the x direction is colored by bright red for the 1st most informative joint, and by light red for the 2nd most informative joint. Only the motion of 1st and 2nd key joints are marked. The temporal weights for each action are shown as gray images. Each square in that image represents a temporal stage. The whiter the square, the higher the temporal weight. The key stage is highlighted by a red box. We illustrate each key stage using its 4 representative 3D poses. The bottom two actions have two key stages each.

joints of skeletal data to see whether our method is robust to joint noises and can meanwhile still pick out the informative joints. We randomly choose 10%, 20%, 30%, 40% and 50% joints of 35 joints of a pose, and for each selected joints we add noise ranging from -5 to 5 to each dimension of coordinates of joints. The involvement of noise will result in mismatches of related dimension of feature when searching for nearest neighbors. The influence of noisy joint on accuracy is shown in Fig. 5. The curve shows that with the increasing of noisy joints percentage, the accuracy of NBNN drops dramatically, while ST-NBNN can maintain high performance on this dataset. Meanwhile, NBNN+SVM is again inferior to ST-NBNN under the setting of noisy joints. ST-NBNN can still pick up the informative joints and keep the performance on high level.

5. Conclusion

In this work, we extend NBNN to ST-NBNN for skeleton-based action recognition. Compared with NBNN, ST-NBNN considers the spatio-temporal structure of 3D actions and combines the strength of both non-parametric model and also parametric model to achieve better performance. Despite using only a linear classifier, the proposed method works surprisingly well on three benchmark datasets and achieves competitive results compared with state-of-the-arts using sophisticated end-to-end models. Moreover, our proposed method can discover critical spatial joints and temporal stages, which are essential to capture the spatio-temporal patterns of 3D actions, and which is not always achievable via using end-to-end models such as deep learning. Our results demonstrate the potential of using non-parametric method for skeleton-based action recognition.

Acknowledgement

This work is supported in part by Singapore Ministry of Education Academic Research Fund Tier 2 MOE2015-T2-2-114.

References

[1] R. Behmo, P. Marcombes, A. Dalalyan, and V. Prinet. Towards optimal naive bayes nearest neighbor. In *European Conference on Computer Vision*, pages 171–184. Springer, 2010. 2

[2] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 1, 2, 3, 4

[3] D. Cai, X. He, J.-R. Wen, J. Han, and W.-Y. Ma. Support tensor machines for text categorization. 2006. 5

[4] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011. 6

[5] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1110–1118, 2015. 1, 2, 5, 6, 7

[6] M. Grant and S. Boyd. Cvx: Matlab software for disciplined convex programming. 6

[7] I. Kapsouras and N. Nikolaidis. Action recognition on motion capture data using a dynemes and forward differences representation. *Journal of Visual Communication and Image Representation*, 25(6):1432–1445, 2014. 7

[8] P. Koniusz, A. Cherian, and F. Porikli. Tensor representations via kernel linearization for action recognition from 3d skeletons. *arXiv preprint arXiv:1604.00239*, 2016. 5, 6, 7

[9] M. Kusner, S. Tyree, K. Q. Weinberger, and K. Agrawal. Stochastic neighbor compression. In *Proceedings of the 31st international conference on machine learning (ICML-14)*, pages 622–630, 2014. 3

[10] I. Kuzborskij, F. M. Carlucci, and B. Caputo. When Naive Bayes Nearest Neighbours Meet Convolutional Neural Networks. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, 2016. 3

[11] W. Li, Z. Zhang, and Z. Liu. Expandable data-driven graphical modeling of human actions based on salient postures. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1499–1510, 2008. 2

[12] W. Li, Z. Zhang, and Z. Liu. Action recognition based on a bag of 3d points. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 9–14. IEEE, 2010. 5, 6

[13] Y. Li, C. Lan, J. Xing, W. Zeng, C. Yuan, and J. Liu. Online human action detection using joint classification-regression recurrent neural networks. In *European Conference on Computer Vision*, pages 203–220. Springer, 2016. 1, 2

[14] J. Liu, A. Shahroudy, D. Xu, and G. Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *European Conference on Computer Vision*, pages 816–833. Springer, 2016. 1, 2, 5, 6, 7

[15] J. Liu and G. Wang. Global context-aware attention lstm networks for 3d action recognition. In *CVPR*, 2017. 2

[16] D. M. Mount and S. Arya. Ann: library for approximate nearest neighbour searching. 1998. 5

[17] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy. Berkeley mhad: A comprehensive multimodal human action database. In *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, pages 53–60. IEEE, 2013. 5, 7

[18] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy. Sequence of the most informative joints (smij): A new representation for human skeletal action recognition. *Journal of Visual Communication and Image Representation*, 25(1):24–38, 2014. 2, 7

[19] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Bilinear classifiers for visual recognition. In *Advances in neural information processing systems*, pages 1482–1490, 2009. 1, 2, 4

[20] S. Sempena, N. U. Maulidevi, and P. R. Aryan. Human action recognition using dynamic time warping. In *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*, pages 1–5. IEEE, 2011. 2

[21] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 2

[22] T. Tuytelaars, M. Fritz, K. Saenko, and T. Darrell. The nbnn kernel. In *2011 International Conference on Computer Vision*, pages 1824–1831. IEEE, 2011. 2

[23] S. Vantigodi and V. B. Radhakrishnan. Action recognition from motion capture data using meta-cognitive rbf network classifier. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*, pages 1–6. IEEE, 2014. 7

[24] V. Veeriah, N. Zhuang, and G.-J. Qi. Differential recurrent neural networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4041–4049, 2015. 1, 2

[25] R. Vemulapalli, F. Arrate, and R. Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 588–595, 2014. 5, 6, 7

[26] C. Wang, J. Flynn, Y. Wang, and A. L. Yuille. Recognizing actions in 3d using action-snippets and activated simplices. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 3, 6, 7

[27] C. Wang, Y. Wang, and A. L. Yuille. Mining 3d key-pose-motifs for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2639–2647, 2016. 2, 6, 7

[28] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1290–1297. IEEE, 2012. 2

[29] P. Wang, C. Yuan, W. Hu, B. Li, and Y. Zhang. Graph based skeleton motion representation and similarity measurement for action recognition. In *European Conference on Computer Vision*, pages 370–385. Springer, 2016. 5, 6, 7

[30] L. Xia, C.-C. Chen, and J. Aggarwal. View invariant human action recognition using histograms of 3d joints. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–27. IEEE, 2012. 2, 5, 7

[31] X. Yang and Y. Tian. Effective 3d action recognition using eigenjoints. *Journal of Visual Communication and Image Representation*, 25(1):2–11, 2014. 3

[32] X. Yang and Y. Tian. Super normal vector for activity recognition using depth sequences. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 804–811. IEEE, 2014. 2

[33] G. Yu, Z. Liu, and J. Yuan. Discriminative orderlet mining for realtime recogntion of human-object interaction. In *Asian Conference on Computer Vision*, 2014. 2

[34] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2442–2449. IEEE, 2009. 3

[35] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 1, 2