DUAL-STREAM CNN FOR STRUCTURED TIME SERIES CLASSIFICATION

Shuchen Weng[1*], *Wenbo Li*[2*]

[1]Tianjin University
College of Intelligence and Computing

Yi Zhang[1†], *Siwei Lyu*[2]

[2]University at Albany, SUNY
Computer Science Department

ABSTRACT

The structured time series (STS) classification problem requires the modeling of interweaved spatiotemporal dependency. Most previous methods model these two dependencies independently. Due to the complexity of the STS data, we argue that a desirable method should be a holistic framework that is adaptive and flexible. This motivates us to design a deep neural network with such merits. Inspired by the dual-stream hypothesis in neural science, we propose a novel dual-stream framework for modeling the interweaved spatiotemporal dependency, and develop a convolutional neural network within this framework that aims to achieve high adaptability and flexibility in STS configurations of sequential order and dependency range. Our model is highly modularized and scalable, making it easy to be adapted to specific tasks. The effectiveness of our model is demonstrated through experiments on benchmark datasets for skeleton based activity recognition.

Index Terms— time series, interweaving, dual stream

1. INTRODUCTION

Time series data are produced from a wide range of activities such as weather readings, stock prices and human motions. The observations in many types of time series data are of high dimensional nature. For instance, the financial time series usually include stocks of various companies, and in vision, human actions can be represented as a concatenated vector of skeletal joints' locations. Multivariate time series with explicit statistical dependencies among components are known as structured time series (STS) [1].

Time series classification is the problem of categorizing different time series into pre-defined classes. One important aspect of STS classification is that the statistical dependencies in the spatial and temporal domains are usually intertwined. However, most previous methods model the spatial and temporal dependencies independently. Furthermore, these methods typically focus on improving individual steps of the pipeline, *e.g.*, explicitly modeling spatiotemporal information [2, 3, 4, 5], increasing sequential orders [2, 6, 7], and adjusting sequential dependency ranges [8, 9]. Due to the complexity of the STS data, we argue that a desirable method should be holistic, adaptive and flexible. This motivates us to design a deep neural network (DNN) with such merits.

Inspired by the *dual-stream neural processing hypothesis* of human visual neural system [10], we propose a dual-stream convolutional neural network (CNN) (as shown in Figure 1). Instead of modeling the spatial and temporal dependency independently, the dual-stream CNN explicitly models the interweaved spatiotemporal dependency in STS data jointly, which is not the case for existing CNN methods of the same name, *e.g.*, [11]. The two streams have different emphasis, either the structural or temporal dependencies, and model such dependencies at different scales. This endows our model higher adaptability and flexibility in modeling the spatiotemporal relationships of input elements, compared to the single-stream [2, 3, 4, 5] based methods. Specifically, a STS instance is first transformed into a 3-rank tensor, with the three dimensions correspond to the time steps, spatial structure and descriptive features of each sample, respectively. The two streams are implemented as CNN so that it is not limited to forward or backward directions as typical sequential models, *e.g.*, recurrent neural networks (RNN) [12]. The dual-stream CNN model is based on a set of *dual-stream convolution kernels*, each formed as a tensor product of two 2D convolution kernels, one on the time and feature axes (red side of STS representation in Figure 1(a)), and one on the structure and feature axes (green side in Figure 1(a)), and we refer to the former as the *temporal kernel* and the latter as the *structural kernel*. The convolutional kernels of the dual-stream CNN model are organized into a hierarchical structure, namely, the structural (temporal) kernels are organized into different levels to represent features of different scales (ranges). We apply a gating module for kernels of different sequential dependency ranges so the contributions of different feature ranges can be determined adaptively in a data-driven fashion.

2. RELATED WORKS

Time series can be roughly categorized in terms of the observation dimensionality as the scalar (*e.g.*, $\{(t_1, 0.1), (t_2, 0.6), \ldots, (t_n, 0.3)\}$) or the multivariate time series (*e.g.*, $\{(t_1, <0.1, 0.3, 0.3>), (t_2, <0.2, 0.5, 0.1>), \ldots, (t_n, <0.8, 0.9, 0.6>)\}$). The difference between STS and these two types lies not only

* indicates equal contributions. † indicates the corresponding author.

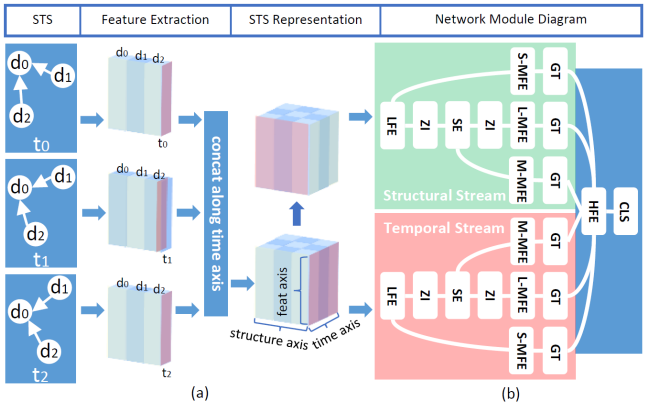

Fig. 1. *(a)* **The STS representation** *is a tensor with three dimensions, which is formed in two steps. We represent each dimension at each time step as a feature vector, and concatenate them according to the dimension and time order. (b)* **Module diagram**. *The red and green blocks represent the temporal and structural stream, respectively. L/M/HFE mean the low/medium/high-level feature extractor, and the prefixes of MFE, S/M/L, mean the short/medium/long range. ZI means the zoom-in module, and SE means the encoder shared by M-MFE and L-MFE. GT and CLS represent the gating and classification module, respectively. See texts for details.*

in its higher observation dimensionality, but also in a key assumption, *i.e.*, there are strong statistical dependencies among the components of each time step. For instance, in skeleton based activity recognition, the components of STS correspond to the joint locations, which can be represented as a tree structure, *e.g.*, $(t_i, <d_1,\ d_2 \rightarrow d_1,\ d_3 \rightarrow d_1,\ d_4 \rightarrow d_2,\ \ldots>)$, where $\rightarrow$ pointing from the children to its parent reveals the tree structure. The spatial/temporal variation of a limb joint is constrained by its parent in the tree.

Classifying STS thus has to take into consideration such intertwining dependencies intra and inter time steps. Previous STS classification methods [13] focus on finding effective hand-crafted features for classification. Most recent works have shifted to deep learning methods, and design DNNs for learning discriminative features automatically. Existing DNN based methods can be roughly divided into three categories.

Integration of Spatial and Temporal Information. [3] incorporated the LSTM with both the spatial and temporal attention mechanisms to adaptively select the discriminative dimensions and time steps for classification. [5] adopted the bilinear classifiers to identify both key time steps and dimensions. [4] formulated the STS as two sequences along the spatial and temporal axes, respectively, and used a two-stream RNN to model these two sequences. A common drawback of these works is that they assume independence between the structural and temporal information in STS.

Sequential Orders. STS instances satisfy the temporal causality, *i.e.*, the observation at the current time step depends on those of the previous time steps, or the other way around. Thus, [6] used a bidirectional RNN structure to model STS. The components of each observation at a time step can be represented by a tree structure. By traversing the tree bidirectionally using the depth-first search, the STS data can also be represented as a sequence spatially. [7] developed the spatiotemporal RNN for modeling such a spatial sequential order as an addition to the temporal sequential order.

Sequential Dependency Ranges. The range of temporal dependency is an important factor in modeling STS. This is commonly modeled using RNN with long-short term memory (LSTM) [14], in which the range is determined by the cooperation of the memory cell and several modulative gates. However, the capability of such an adaptive modeling would be overstretched given the high complexity of the STS data. To this end, several works explored to model the dependency ranges in a more controllable fashion. For example, [9] incorporated the LSTM with multi-scale temporal sliding windows, so as to explicitly control the dependency ranges. [8] represented the STS data as 2D clips, and used the CNN to capture the long-range dependencies.

3. DUAL-STREAM FORMULATION WITH CNN

The temporal and structural dependency are important patterns for STS classification. The temporal dependency is commonly modeled with a chain structure that goes in forward or backward direction, such as in RNN. This is based on the *sequential causality assumption* that the intrinsic temporal dependency of a sequence along the time axis.

However, the sequential causality assumption does not always hold, as suggested by the indefinite causal order theory in quantum mechanics [15], *i.e.*, the causality order does not always obey a specific element permutation, but a mixture of multiple permutations. We refer to such a problem as the *indefinite order problem*, which contradicts the sequential causality assumption made by RNN. In particular, for a RNN, long-range dependencies between two distant elements in a sequence might be affected by many other irrelevant elements on the long path through the chain. It is also hard for RNNs to accommodate such "indefinite" permutations on the fly due to the indifferentiability of permuting operations.

Mitigating the long-range dependency modeling problem and indefinite order problem requires us to avoid using the chain structure but to create a STS classification model that is more flexible and adaptive, and to reduce the impacts caused by different element positions in a sequence. In this work, we model the sequential order of STS using the multi-layer CNN, which creates hierarchical representations over the input STS in which the dependencies of nearby elements are modeled by lower layers while those of distant elements are modeled by higher layers. The replacement of RNN with CNN alleviates the two aforementioned problems, as the sequential dependency modeling is no longer strictly limited by a sequential order or a chain structure, but directly handled by the multi-scale receptive fields of CNN. In the following, we describe the overall processing steps, starting with an augmented STS representation and then on the structure of the model itself.

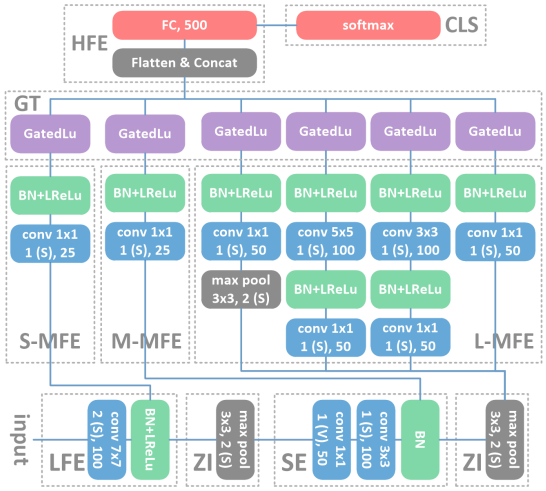

Fig. 2. *The network structure of a single stream of our model.*

3.1. STS Representation

We first augment the original STS data with empirical handcrafted features before the feature learning process to form a rank-3 tensor $R_{d,t,f}$ and $R_{t,d,f}$ as follows. Given a STS $\{(t_i, < d_1, d_2 \rightarrow d_1, d_3 \rightarrow d_1, d_4 \rightarrow d_2, \dots >)\}_{i=1}^{n}$ with each dimension being a point $d_j = (x_1^j, x_2^j, \dots, x_l^j)$ in the l dimensional space, we follow [16, 17] to extract four types of features for each dimension d_j at each time step t_i, and concatenate them to form a feature vector $h_{d_j}^{t_i}$: (1) **Position.** $x_1^j, x_2^j, \dots, x_l^j$ are concatenated to form a l dimensional feature vector; (2) **Angles.** Given multiple edges $\{e_k^j\}_{d_k \in \aleph_j}$ connecting d_j and its neighboring dimensions $\aleph_j$ in the tree, we compute the normalized pairwise angles between these edges; (3) **Offset.** Offsets of elements in d_j between t_i and t_{i-1} are computed and concatenated to form a l dimensional feature vector. (4) **Distance.** We calculate the pairwise distance between d_j and the mean position of all dimensions at t_i.

Then, we pad the extracted features to be equal in length, and concatenate the extracted features to form the STS representation, *i.e.*, $R_{t,d,f} = (h_{d_j}^{t_1}, h_{d_j}^{t_2}, \dots, h_{d_j}^{t_n})_{j=1}^{m}$; similarly, we have $R_{d,t,f} = (h_{d_1}^{t_i}, h_{d_2}^{t_i}, \dots, h_{d_m}^{t_i})_{i=1}^{n}$ which is a transpose of $R_{t,d,f}$. The order of the STS dimensions in $R_{t,d,f}$ is determined by the traversing algorithm [7] starting from d_1.

3.2. Model Structure

The temporal and structural streams in the dual-stream CNN model share similar structures. Each stream focuses on a distinct aspect of the spatiotemporal structure in the STS, which is reflected by different parameterization strategy. Given a STS in the tensor representation described previously, we process the 2D slices constituted by the time and feature elements of the STS tensor with the temporal stream CNN, and process the 2D slices constituted by the structure and feature elements of the STS tensor with the structural stream CNN. For the temporal stream, the convolutional computation (parameterized) on the time and feature axes captures the temporal dependencies, while the spatial dependencies are captured by the addition computation (non-parameterized). The case is reversed for the structural stream. This different characteristics of the two streams suggest that the dual-stream design cannot be simply replaced by a single-stream 3D-CNN. For the choice of the convolutional kernels, we use the 2D kernel rather than using the 3D one. This is because 2D kernel performs the fully connected (shared weight for 3D kernel) computation along the time or structure axis, which makes it beneficial for modeling the long-range dependencies.

The preprocessed STS features are fed for feature learning. The adaptive selection of sequential dependency range is important to STS classification, so we fuse the adaptive learning of the sequential dependency range into the feature learning process. There are ten building blocks within our model as shown in Figure 2, including low/medium/high-level feature extractor (L/M/HFE), two zoom-in modules (ZI), a shared encoder (SE), a gating module (GT) and a classification module (CLS). The MFE is composed of three sub-blocks, *i.e.*, short/medium/long-range MFE.

As shown in Figure 2, we decompose the feature learning process into three stages, *i.e.*, low/medium/high-level feature extractors (L/M/HFE). The low-level features keep more details of the original input STS data, while the high-level features are more conceptual which are used for classification directly. The medium-level features bridge the low-level and the high-level features, so it determines the reliability and the meaningfulness of the high-level features.

LFE consists of a conv layer with kernel size 7 and the batch normalization (BN) followed by a leaky relu (LReLu). We chose a large kernel size because the consecutive elements in a sequence may contain much redundancy. We use LReLu as the nonlinearity because it does not gate the negative values in the STS representation.

MFE plays the key role in feature learning, so its architecture is the most complicated. MFE is decomposed into three sub-stages with each focusing on a specific sequential dependency range corresponding to the short/medium/long-range MFE (denoted as S/M/L-MFE). The MFEs for different dependency ranges are connected by zoom-in (ZI) modules and shared encoders (SE). ZI is implemented as a max pooling layer, and SE is a block shared by the M-MFE and L-MFE. Since the space covered by L-MFE is larger than that of short/medium-range MFEs, we further split L-MFE into four finer scales similarly to the inception module in [18].

GT is posed as the backend of S/M/L-MFE which adaptively determines the contribution of each sequential dependency range to the high-level features. We implement the gating module as the gated linear unit (GatedLu) [19] over the output of the convolution $Y = [A\ B] \in \mathbb{R}^{w,h,2c}$, $v([A\ B]) = A \otimes \sigma(B)$, where $A, B \in \mathbb{R}^{w,h,c}$ are the inputs to the nonlinearity, $\otimes$ is the point-wise multiplication and the output $v([A\ B]) \in \mathbb{R}^{w,h,c}$ is half size the size of Y. The gates $\sigma(B)$ (implemented as the sigmoid) determine the importance of inputs A for learning the high-level features.

HFE is formed with a fully connected layer (FC) which takes the flattened and concatenated medium-level features output by S/M/L-MFE, and outputs a 500 dimensional vector.

Classification. The extracted high-level features are fed into a softmax layer to obtain the probability distributions, which are used to compute the negative log-likelihood loss, $L = -\sum_i y'_i \log(y_i)$, where y_i is the probability distribution output of an input STS by the softmax layer, and y' is its corresponding ground truth one-hot vector representation.

4. EXPERIMENTS

Experimental Settings. We evaluate the dual-stream CNN model on the STS data from three skeleton based activity recognition benchmark datasets: MSR Action3D [20], CharLearn Italian [21] and 3D-SAR-140 [17]. Our method is implemented using Python and TensorFlow[1], and all experiments are conducted on four machines on each of which an NVIDIA TITAN X GPU with 12GB onboard memory is installed. The overall objective function is minimized using back-propagation implemented with the ADAM algorithm [22]. We train the network using mini-batch gradient descent, and set learning rate, momentum and decay rate as 1×10^{-3}, 0.9, 0.999. As usual, we scale the input to be equal in temporal length, so as to enable the mini-batch processing.

Compared Methods. We compared our method against 10 existing methods, *i.e.*, RR [23], HBRNN-L [6], CHARM [24], DBN-HMM [25], Lie-group [26], HOD [27], MP [28], SSS [29], HBRNN-L-T [17] and URNN-2L-T [17].

MSR Action3D. This dataset consists of 20 actions performed by 10 subjects for two or three times, 544 valid samples with 21,462 frames. We follow the experimental protocol presented in [30] on this dataset, which is the most challenging protocol for this dataset. Half of actor subjects are used for training and the rest are used for test. Note that the average number of training samples per class is nearly 14, which is quite limited for training deep neural networks, and poses great potential risks on the overfitting issue. The comparison on MSR Action3D in Table 1 demonstrates the good generalization capability of our method.

CharLearn Italian. This dataset captures 20 Italian cultural signs, and contains 393 labeled sequences with a total of 7,754 gesture instances. We follow the experimental protocol in [25]: 350 sequences for training and the rest 43 sequences for testing. The recognition of sign languages requires the fine-grained recognition ability of the evaluated methods, and always desire the careful feature representation design. As shown by the comparison on CharLearn Italian in Table 1, our method constantly outperforms the evaluated methods without designing any special features for this dataset.

3D-SAR-140. This dataset [17] contains 140 diverse action classes. It is challenging due to two factors: (1) a large variety of movements in various contexts are included, where fine-grained recognition is required; (2) sequence length for individual actions varies significantly (ranging from 5 to 800 frames), which poses the challenges on the adaptive configuration of the sequential dependency range. Notably, URNN-2L-T is designed to recognize fine-grained actions and large-scale dataset. However, our method still performs slightly better than URNN-2L-T. The good performance also demonstrates our method's effectiveness in adaptively configuring the sequential dependency ranges.

[1]https://github.com/SCWengTJU/DualStreamCNN

Table 1. *Classification accuracy.*

Methods	MSR Action3D	ChaLearn	3D-SAR-140
RR	0.891	0.438	0.723
HBRNN-L	0.897	0.559	0.604
CHARM	0.747	0.476	0.618
DBN-HMM	0.735	0.628	0.601
Lie-group	0.866	0.401	0.745
HOD	0.844	0.539	0.657
MP	0.909	0.452	0.203
SSS	0.560	0.413	0.253
HBRNN-L-T	0.915	0.673	0.756
URNN-2L-T	0.931	0.753	0.892
Ours $\ominus$ gating module	0.947	0.766	0.864
Ours $\ominus$ structural stream	0.848	0.677	0.814
Ours $\ominus$ temporal stream	0.934	0.729	0.889
Ours	**0.963**	**0.772**	**0.896**

Ablation Study. We evaluate three components in our method, *i.e.*, the gating module, structural stream and temporal stream. We disable these components one by one and conduct the evaluation. Table 1 shows the comparison results, and it clearly shows that each of these three components is beneficial for the generalization. Generally speaking, the descending order of their influences is structural stream, gating module and temporal stream. An interesting phenomenon is that the structural stream seems to have more important effect than the temporal stream in the final classification performance, which accords with the theory of dual-stream hypothesis in neural science. This strengthens the reasonability of our inspiration drawn from the dual-stream hypothesis, and further shows the necessity of interweaving spatiotemporal modeling for STS classification.

5. CONCLUSION

We propose a novel dual-stream framework for modeling the interweaved spatiotemporal dependency, and develop a CNN within this framework that achieves high adaptability and flexibility in STS configurations of the sequential order and dependency range. Our model is highly modularized and scalable, making it easy to be adapted to specific tasks. The effectiveness of our model is demonstrated through experiments on activity recognition benchmark datasets.

Acknowledgement. W. Li and S. Lyu are supported by the United States Air Force Research Laboratory (AFRL) and the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-16-C-0166.The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. S. Weng and Y. Zhang are supported via National Natural Science Foundation of China (Grant No. 61702359).

6. REFERENCES

[1] Dian Gong, Gérard G. Medioni, and Xuemei Zhao, "Structured time series analysis for human action segmentation and recognition," *TPAMI*, vol. 36, no. 7, pp. 1414–1427, 2014.

[2] Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, and Alex C. Kot, "Global context-aware attention LSTM networks for 3d action recognition," in *CVPR*, 2017, pp. 3671–3680.

[3] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *AAAI*, 2017, pp. 4263–4270.

[4] Hongsong Wang and Liang Wang, "Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks," in *CVPR*, 2017, pp. 3633–3642.

[5] Junwu Weng, Chaoqun Weng, and Junsong Yuan, "Spatio-temporal naive-bayes nearest-neighbor (ST-NBNN) for skeleton-based action recognition," in *CVPR*, 2017, pp. 445–454.

[6] Yong Du, Wei Wang, and Liang Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *CVPR*, 2015, pp. 1110–1118.

[7] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang, "Spatio-temporal lstm with trust gates for 3D human action recognition," in *ECCV*, 2016, pp. 1–8.

[8] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Ahmed Sohel, and Farid Boussaïd, "A new representation of skeleton sequences for 3d action recognition," in *CVPR*, 2017, pp. 4570–4579.

[9] Inwoong Lee, Doyoung Kim, Seoungyoon Kang, and Sanghoon Lee, "Ensemble deep learning for skeleton-based action recognition using temporal sliding LSTM networks," in *ICCV*, 2017, pp. 1012–1020.

[10] Joel Norman, "Two visual systems and two theories of perception: An attempt to reconcile the constructivist and ecological approaches," *Behavioral and brain sciences*, vol. 25, no. 1, pp. 73–96, 2002.

[11] Karen Simonyan and Andrew Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014, pp. 568–576.

[12] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin, "Convolutional sequence to sequence learning," in *ICML*, 2017, pp. 1243–1252.

[13] Jake K. Aggarwal and Lu Xia, "Human activity recognition from 3D data: A review," *PR Letters*, vol. 48, pp. 70–80, 2014.

[14] Alex Graves, *Supervised sequence labelling with recurrent neural networks*, vol. 385 of *Studies in Computational Intelligence*, 2012.

[15] Ognyan Oreshkov, Fabio Costa, and Časlav Brukner, "Quantum correlations with no causal order," *Nature communications*, vol. 3, pp. 1092, 2012.

[16] Vivek Veeriah, Naifan Zhuang, and Guo-Jun Qi, "Differential recurrent neural networks for action recognition," in *ICCV*, 2015, pp. 4041–4049.

[17] Wenbo Li, Longyin Wen, Ming-Ching Chang, Ser-Nam Lim, and Siwei Lyu, "Adaptive RNN tree for large-scale human action recognition," in *ICCV*, 2017, pp. 1453–1461.

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

[19] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier, "Language modeling with gated convolutional networks," in *ICML*, 2017, pp. 933–941.

[20] Wanqing Li, Zhengyou Zhang, and Zicheng Liu, "Action recognition based on a bag of 3D points," in *CVPRW*, 2010, pp. 9–14.

[21] Sergio Escalera, Jordi Gonzàlez, Xavier Baró, Miguel Reyes, Oscar Lopes, Isabelle Guyon, Vassilis Athitsos, and Hugo Jair Escalante, "Multi-modal gesture recognition challenge 2013: Dataset and results," in *ICMI*, 2013, pp. 445–452.

[22] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, 2014.

[23] Raviteja Vemulapalli and Rama Chellappa, "Rolling rotations for recognizing human actions from 3D skeletal data," in *CVPR*, 2016, pp. 4471–4479.

[24] Wenbo Li, Longyin Wen, Mooi Choo Chuah, and Siwei Lyu, "Category-blind human action recognition: A practical recognition system," in *ICCV*, 2015, pp. 4444–4452.

[25] Di Wu and Ling Shao, "Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition," in *CVPR*, 2014, pp. 724–731.

[26] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa, "Human action recognition by representing 3D skeletons as points in a Lie group," in *CVPR*, 2014, pp. 588–595.

[27] Mohammad Abdelaziz Gowayyed, Marwan Torki, Mohammed Elsayed Hussein, and Motaz El-Saban, "Histogram of oriented displacements (HOD): describing trajectories of human joints for action recognition," in *IJCAI*, 2013, pp. 1351–1357.

[28] Mihai Zanfir, Marius Leordeanu, and Cristian Sminchisescu, "The moving pose: An efficient 3D kinematics descriptor for low-latency action recognition and detection," in *ICCV*, 2013, pp. 2752–2759.

[29] Xin Zhao, Xue Li, Chaoyi Pang, Xiaofeng Zhu, and Quan Z. Sheng, "Online human gesture recognition from motion data streams," in *ACM MM*, 2013, pp. 23–32.

[30] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *CVPR*, 2012, pp. 1290–1297.