Global-Local Temporal Saliency Action Prediction

Lai, Shaofan ; Zheng, Wei-Shi; Hu, Jian-Fang ; Zhang, Jianguo

Published in:
IEEE Transactions on Image Processing

DOI:
10.1109/TIP.2017.2751145

Publication date:
2017

Document Version
Peer reviewed version

Global-Local Temporal Saliency Action Prediction

Shaofan Lai, Wei-Shi Zheng, Jian-Fang Hu, and Jianguo Zhang

***Abstract*—Action prediction on a partially observed action sequence is a very challenging task. To address this challenge, we first design a global-local distance model, where a global-temporal distance compares subsequences as a whole and local-temporal distance focuses on individual segment. Our distance model introduces temporal saliency for each segment to adapt its contribution. Finally, a global-local temporal action prediction model is formulated in order to jointly learn and fuse these two types of distances. Such a prediction model is capable of recognizing action of 1) an on-going sequence and 2) a sequence with arbitrarily frames missing between the beginning and end (known as gap-filling). Our proposed model is tested and compared to related action prediction models on BIT, UCF11 and HMDB datasets. The results demonstrated the effectiveness of our proposal. In particular, we showed the benefit of our proposed model on predicting unseen action types and the advantage on addressing the gapfilling problem as compared to recently developed action prediction models.**

***Index Terms*—Action prediction, Gapfilling**

I. Introduction

Action *prediction* is a new dimension towards understanding human activities. Different from action recognition, which has been well studied in the recent decades [1], [2], [3], [4], [5], [6], action prediction is to recognize actions without observing the complete action execution. It is of high demand on deploying action prediction in many real-world scenarios. For example, the tasks like health care assistance, robotic equipment control, and criminal activities surveillance, expect predicting human action before it has been fully executed in order to prevent damage and make prompt reaction. Unlike action recognition, action prediction requires models to discover the temporal-spatial relationship between early segments and unobserved segments. Although existing action recognition methods can be directly applied to action prediction, they are non-optimal for action prediction [2], [3], [7], because they rely on full observation of activity sequence, and are not specifically designed for partially observed sequence.

The majority of the existing action prediction works focus on either developing classification models on partially observed sequences[10], [11], [12], [8] or exploiting reliable features [13], [14], [9], [15]. However, they still have the following limitations.

1) Most of them treat the observed action segments equally [9], [8]. However many actions could be well characterized by only a few segments. Taking the "kicking" action in Figure 1 as an example, the segments in the middle of the sequence are visually more informative in identifying the action than the rest. Thus, better performance could be achieved by assigning rational weights on different segments.
2) Existing models for *action prediction* emphasize too much on discriminative learning of local segments [8], [9]. The global description of an on-going sequence is often overlooked for the prediction. However, the global temporal description is very useful in recognizing some actions which are locally similar. For example, in Fig. 2, the motion information of "patting" and "pushing" is largely shared, and thus it is difficult to discriminate the two actions by just looking at local segment, but they could be well distinguished on global (aggregated) information at a larger scale.
3) Most of the existing action prediction methods rely on template generation [8], [9], [14], or training action-specific SVM classifiers [10], [11]. They cannot learn a metric, and they are not good for potentially generalization to actions types that have not been seen in training set. Actually different human actions could share in certain aspects. For example, patting can be decomposed into "raising

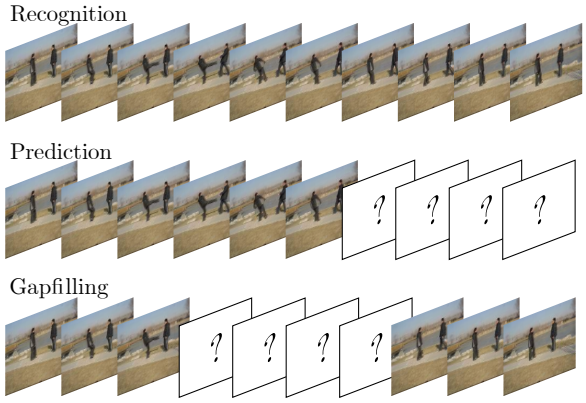


Fig. 1: Illustration of three problems in identifying action from a sequence; (1) action recognition (with fully observed sequence); (2) action prediction, where the unobserved segments is after the observed ones; (3) gapfilling, where some segments of an action sequence is arbitrarily unobservable.

S. Lai, W.-S. Zheng, and J.-F. Hu are with the School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China. W.-S. Zheng is also with the Key Laboratory of Machine Intelligence and Advanced Computing (Sun Yat-sen University), Ministry of Education, China. J.-F. Hu is also with Guangdong Province Key Laboratory of Information Security, P. R. China. W.-S. Zheng is the corresponding author.
E-mail:, laishaovan@gmail.com, zhwshi@mail.sysu.edu.cn, and hujf5@mail.sysu.edu.cn

J. Zhang is with the School of Science and Engineering (Computing), University of Dundee United Kingdom.
E-mail:j.n.zhang@dundee.ac.uk

TABLE I: Difference between our method and the existing prediction models.

Methods	Local/Global Information	Temporal Saliency	Gapfilling	Observation Ratio NOT Available	Generalization to Unseen Classes	Learning Methodology
DBoW[8]	✓/	uniform				Bayesian
IBoW[8]	✓/	uniform				Bayesian
MSSC[9]	✓/	uniform	✓			Sparse Coding
SC[9]	✓/	uniform	✓			Sparse Coding
MTSSVM[10]	✓/✓	implicitly learned		✓		Structured SVM
MMAPM[11]	✓/✓	implicitly learned		✓		Structured SVM
Our Method	✓/✓	explicitly learned	✓	✓	✓	Metric learning

Fig. 2: The first row shows part of action "patting", and the second row is part of action "pushing". They resemble each other in most columns, which represent local segments. But by considering motion in longer time, namely the whole sequence, the two actions would be potentially distinguished by accumulating subtle differences of consecutive segments in pushing and patting.

arms" and "patting the shoulder", while pushing can be decomposed into "raising arms" and "pushing away". Intuitively, the pattern of "raising arms" is shared among many actions related to hands, and assigning certain weights to it can help us to distinguish them from others, like "kicking", "running" and "bowing". Therefore, metric learned from one set of action types could be generalized to recognize unseen action types.

Another branch of partially observed action recognition task is gapfilling[1]. In gapfilling, unobserved frames could appear in arbitrary parts of an action sequence due to camera shaking or occlusion. However, to the best of our knowledge, the gapfilling problem was seldom addressed except the work in [9]. One fundamental difference between gapfilling and prediction is that the gaps separate the observed frames into temporally disjointed segments. Most of existing prediction models assume that the observed action execution must be continuous and thus cannot be applied under this scenario.

To address the aforementioned problems on understanding partially observed action sequence, we propose a global-local temporal action prediction framework. Specifically, we learn distance measures between subsequences, each of which is the accumulated segments since the start of an action. This is termed as *global-temporal distance*. We also learn a distance measure between different segments at the same observation level so as to compare the features from local segments, and we name this as the *local-temporal distance*. Whilst modeling global-temporal distance is helpful in characterizing one ongoing sequence as a whole, the local-temporal distance intends to capture some local action cues for prediction and thus can be used to tackle the gapfilling problem when continuity of sequence cannot be guaranteed. The global- and local-temporal distance measures are learned jointly to accomplish prediction as well as gapfilling. Since the importance of observed segments differs as time goes, we introduce temporal saliency weights to fuse all local-temporal distance measures in a selective way rather than combining them uniformly. Moreover, our saliency weights are allowed to change over time, which means that the importance of each individual local observation can be different as more information about the action is observed. Due to the nature of distance metric learning, our global-local temporal action prediction framework can be applied to help predict action types that are not seen in the training stage.

In this paper, a set of experiments were carried out for action prediction on BIT-Interaction dataset (BIT) [16], UCF11 [17] and HMDB [18] to demonstrate the effectiveness of our proposed model. It was shown that our proposal can 1) conquer gapfilling better than [9]; 2) perform reliably upon the observation ratio estimation; and 3) learn a set of metrics with good generalization ability.

In summary, our contributions are several folds: 1) introducing a global-local distance model with temporal saliency; 2) a novel action prediction method with cost function driven by *push* and *pull* strategies; 3) a model for addressing the gappfiling problem; 4) extensive comparison with state-of-

[1]For consistency, we follow the same terminology introduced in [9]. It refers to the task of recognizing action sequence with a few frames having content missing.

the-arts and comprehensive evaluation of model parameters; 5) demonstrating compatibility on improving the deep neural network for action prediction.

II. Related Work

Action Recognition. There exists a large body of work in action recognition in last decades. Many of these approaches are based on low-level features, for instance, appearance and the spatio-temporal local features [19], [20], [21]. Beyond low-level features, high-level semantic concepts [22], [3], pose-based information[23], [24] and data-driven concepts[25] were also explored to make the video representation more expressive. Other approaches intended to discover some common temporal patterns for human action recognition using sequential models, where action videos were treated as a composition of consecutive segments [26], [27], [28]. Recently, an end-to-end deep two-stream network that combines both RGB and optical flow information[29], [4] achieved state-of-the-art performance for action recognition on several datasets. However, those action recognition models assume that the action sequence is fully observed by the system.

Although action prediction at a given observation ratio[2] can be treated as a conventional action recognition problem, it is naive and non-optimal to simply formulate action prediction as an ensemble of action classification tasks without any further modeling. This is mainly because conventional action recognition holds the assumption that the temporal information of an action is mostly complete, while only partial temporal information of an action is observed in action prediction.

Action Prediction. Action prediction has become popular recently. One representative approach is to formulate a template based model for the prediction. For example, Ryoo [8] generated templates for different actions by averaging features of the same categories, and the likelihood between samples and template is calculated and employed in a Bayesian model to make prediction. However, these templates could be easily affected by the outliers and would perform poorly when actors present large pose variations. To fix this issue, a sparse representation is built for each testing sample and the reconstruction error was used for calculating the likelihood between templates and testing samples in [9]. Lan et al. [14] also exploited templates but at multiple levels of granularities in a hierarchical representation, which can capture and compare human movements at different context levels. In [15], the ongoing sequence is considered as a prefix, whose unobserved parts as well as action type were auto-completed by the model using extracted discriminative patches.

Another line of work focused on discovering temporal characteristics of human actions. In [9], the templates were employed for matching, and a dynamic programming scheme was used to compute optimal likelihood of a sample. By leveraging the fact that the confidence of prediction should increase with more frames gradually observed, a temporal evolving margin was proposed in [10], [11]. Action representation was derived from multiple segmentations and a kernel designed on the best hierarchical structure was applied in [12]. Vondrick et al. [30] developed a deep neural network approach which minimized the difference between the predicted features and the ground truth features of the future frames. By mining temporal sequence patterns and discovering causality, context-cue and predictability of human activities, it was able to predict long-duration complex actions [31]. The depth information was also leveraged to predict action in a soft regression framework to improve the prediction accuracy [32]. Recently, the task of online action detection [33] has been reported, which is another related but different challenging task requiring model to output when to start and end, and what type it is for an action on the fly. In comparison action prediction can be regarded as a sub-procedure after the start of an action is marked.

Note that the holistic feature representation of the whole on-going action sequence alone does not perform well in action prediction, although it is widely used in the conventional action recognition task [2], [7]. Nevertheless, the global information is able to provide an overall description of the observed sequence, albeit not the complete action sequence. It could complement the use of local segment for predicting an action type, especially when local segments of different actions look similar.

Action Gapfilling. Both action prediction and action with gapfilling problem can be generally considered as partial action recognition. In action prediction, the unobserved parts are those consecutive frames following the observed segments of an action, while in gapfilling the unobserved parts (i.e., gaps) can appear anywhere in an action sequence. In general, gaps can occur due to strong camera movement or temporarily blocking of view. Clearly action gapfilling is more general and challenging than prediction; however, it received less attention than the prediction problem. Most of the existing action prediction models [10], [11], [8] cannot be used to address gapfilling because they assume that the observed video must be consecutive.

To the best of our knowledge, gapfilling problem was only addressed very recently by Cao et al [9], where the gapfilling problem was studied by discarding information contained in the unobserved segments. Different from this work, we form a discriminative local-temporal distance metric measuring the similarity between segments at the same observation level so that our model is flexible enough to handle the gapfilling in a more principled way without discarding any action segments.

Metric Learning. Our model also relates to metric learning. Indeed, metric learning is a long-term research topic and many distance metric models have been developed including LMNN[34], NCA[35], MCML[36], and the kernel extension [37]. They have been widely applied in computer vision, such as face recognition[38], tracking[39], person re-identification [40], image retrieval [41] and text retrieval [42].

However, these metric approaches are not particularly designed for action prediction. While most of previous models for action prediction are based on SVM or Bayesian model, our work offers the first attempt of developing a distance metric learning based model in this perspective. We introduced a global-local temporal distance to compute the similarity between the observed sequences, so that it provides both

[2]The observation ratio will be formally defined in Sec. III-A

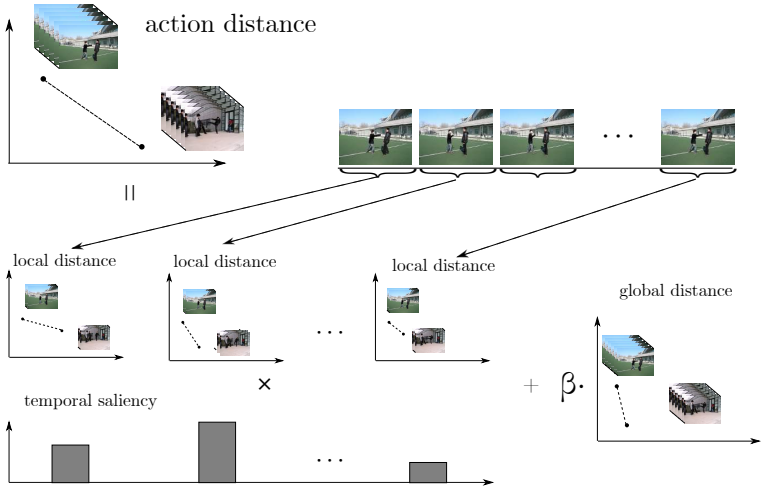


Fig. 3: The similarity of two observations is defined by the sum of local-temporal distance weighted by temporal saliency and global-temporal distance. Local-temporal distance metric is learned in each segmental space $\mathcal{H}^{(t)}$ while the global-temporal distance metric provides a global perspective.

local and global comparison of features from on-going action sequence. Moreover, our learned metric could generalize to unseen action classes, while existing work on action prediction cannot. In one word, we are not focusing on exploring a pure distance metric model, but investigating how distance metric learning can be developed to solve the action prediction problem in a principled way.

III. Approach

In this section, we first present some key notations used in this work. We then introduce our global-local temporal saliency distance model to measure the distance between two partially observed video clips. Finally, the objective function of the proposed prediction model is defined, followed by the optimization algorithm.

A. Segments Extraction

Action prediction can be viewed as a task of recognizing action with partial executions. Following the settings in [11], each action video is uniformly split into G shorter segments. An action video $\boldsymbol{v}$ containing M frames is denoted as $\boldsymbol{v}[1, M]$, and each segment has $\frac{M}{G}$ frames. We use $\boldsymbol{v}^{(t,e)} = \boldsymbol{v}[\frac{M}{G} \cdot (t-1)+1, \frac{M}{G} \cdot e](1 \leq t \leq e \leq G)$ to denote the partial action video starting from the t-th segment and ending at the e-th segment. Especially, when $t = e$, $\boldsymbol{v}^{(e,e)} = \boldsymbol{v}^{(e)}$, denoting the e^{th} segment of video.

After splitting an action video into G segments, an observation level is defined, which is represented by the ratio between the number of observed segments and the total number of segments from that video, termed as *observation ratio*. For instance, if the first i segments of a given video are observed by the system, then its observation ratio is $\frac{i}{G}$. Thus, in total, we could have G observation levels for an action video.

Video features are extracted from each observation $\boldsymbol{v}^{(t,e)}$, and denoted as $\boldsymbol{x}^{(t,e)}$. Therefore $\boldsymbol{x}^{(1,g)}$ is the feature representation of a subsequence of a video from the first segment to the g^{th} segment. Similarly, we have $\boldsymbol{x}^{(e)} = \boldsymbol{x}^{(t,e)}$ when $t = e$. We also use a feature space $\mathcal{H}^{(t)}$ to indicate the space spanned by $\{\boldsymbol{x}_i^{(t)}\}_{i=1}^{N}$, where N is the total number of training samples and $\boldsymbol{x}_i^{(t)}$ is the feature extracted from the t^{th} segment of the i^{th} sequence.

B. Global-Local Temporal Distance

In order to predict actions at different observation levels, a metric $D_g(\cdot,\cdot)$ is introduced to measure the distance between two observations at the g^{th} level (i.e., the sequence observed till the g^{th} segment). If the action sequence is divided into G segments, a total of G action metrics need to be learned.

The Local Temporal Distance. It is indeed that sometimes the underlying action can be probably identified by just providing a glimpse of the sequence [14]. For example, a segment of "shaking fist" would probably signify that the action being performed is "punching". It is expected that the distance between the corresponding segments from the same action should be as small as possible and the one between segments from different actions should be large, so as to mine some discriminant local details for early action classification. For this purpose, the distance between the t^{th} segments of two action sequences can be formulated as:

$$d^{(t)}(\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_j^{(t)}) = (\boldsymbol{x}_i^{(t)} - \boldsymbol{x}_j^{(t)})^T \boldsymbol{M}^{(t)} (\boldsymbol{x}_i^{(t)} - \boldsymbol{x}_j^{(t)}), \quad (1)$$

where $\boldsymbol{M}^{(t)}$ is a positive semi-definite metric matrix.

The Global-Temporal Distance. Since local-temporal distance only measures the similarity between local segments, it could struggle to differentiate the actions that look locally similar, i.e., by just looking at a local segment. To address this issue, a global-temporal distance $d^{(1,g)}(\boldsymbol{x}_i^{(1,g)}, \boldsymbol{x}_j^{(1,g)})$ is employed as a complementary measure to compute the distance between different observations in Eq. (1) as follows:

$$d^{(1,g)}(\boldsymbol{x}_i^{(1,g)}, \boldsymbol{x}_j^{(1,g)}) = (\boldsymbol{x}_i^{(1,g)} - \boldsymbol{x}_j^{(1,g)})^T \boldsymbol{A}_g (\boldsymbol{x}_i^{(1,g)} - \boldsymbol{x}_j^{(1,g)}), \quad (2)$$

where $\boldsymbol{A}_g$ is positive semi-definite matrix. Different from the local feature $\boldsymbol{x}_i^{(t)}$ that captures local action details, the global feature $\boldsymbol{x}_i^{(1,g)}$ provides an overview description of the whole observed sequence.

The Global-Local Temporal Saliency Distance. To incorporate both global and local temporal cues, we build an ensemble of local-temporal distance functions $d^{(t)}(\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_j^{(t)})$, $(t = 1, \cdots, g)$ and global-temporal distance function $d^{(1,g)}(\boldsymbol{x}_i^{(1,g)}, \boldsymbol{x}_j^{(1,g)})$ such that the distance of the samples of the same action is minimized and the distance of samples from different classes is maximized. As shown in Figure 3, the distance between the first g segments of videos $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ is defined as:

$$D_g(\boldsymbol{v}_i, \boldsymbol{v}_j) = \sum_{t=1}^{g} \boldsymbol{\alpha}_g^{(t)} d^{(t)}(\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_j^{(t)}) + \beta d^{(1,g)}(\boldsymbol{x}_i^{(1,g)}, \boldsymbol{x}_j^{(1,g)}), \tag{3}$$

where $d^{(t)}$ is the local-temporal distance measuring the similarity between the t^{th} segments from videos $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$, and $d^{(1,g)}$ measures the similarity of the observations from a global temporal view. A parameter β is used to control the trade-off between the local and global temporal distances.

In the above Eq. (3), we introduce an $\boldsymbol{\alpha}_g$, called the *temporal saliency*, to measure the contribution of each local temporal distance in classification task $T_g(g \in \{1, 2, ..., G\})$, where only the first g segments are observed. We observe that some segments are more important than the others for characterizing an action. The importance of the observed segments could change if more future segments are observed. Taking the action "kicking" presented in Figure 8 as an example, information from only a few segments (in the middle) of the whole action video could be sufficient to tell the underlying action. Hence, we are motivated to weight the local segments differently. Note that, each $\boldsymbol{\alpha}_g$ is a g-dimensional vector, shared among different actions. To ensure that the scale of $\boldsymbol{\alpha}_g$ will not affect the ensemble distance modeling, we add the constraint that $\sum_{t=1}^{g} \boldsymbol{\alpha}_g^{(t)} = 1$ and $\boldsymbol{\alpha}_g^{(t)} \geq 0$.

C. Global-Local Temporal Action Prediction Model

Based on the above formulated Global-Local temporal distance (i.e. Eq.(3)), we now describe how the distance could be used for action prediction. The prediction model is formed by taking use of the holistic feature extracted from the entire observed sequence and the temporal local features extracted from each segment. Two losses are introduced as functions of distances defined on the two types of features, respectively, namely the *successive segments loss* and the *individual segment loss*.

Successive Segments Loss. In order to quantify the global-local temporal distance measure on action prediction of the whole observed sequence, we introduce a joint learning on the global-temporal and the ensemble of local-temporal models weighted by temporal saliency using the large margin idea [34]. Specifically, the successive segments loss function is defined as follows:

$$L_g(\boldsymbol{\alpha}_g, \boldsymbol{A}_g, \{\boldsymbol{M}^{(i)}\}_{i=1}^{g}) = \mu E_{pull_g} + (1 - \mu) E_{push_g}, \tag{4}$$

where μ is the parameter to balance the effects of the "pull" and "push" operations. E_{pull_g} is for "pull" strategy and defined as:

$$E_{pull_g}(\boldsymbol{\alpha}_g, \boldsymbol{A}_g, \{\boldsymbol{M}^{(i)}\}_{i=1}^{g}) = \sum_{(i,j)\in\mathcal{S}} D_g(\boldsymbol{v}_i, \boldsymbol{v}_j), \tag{5}$$

where $\mathcal{S} = \{(i, j) | y_i = y_j\}$, y_i, y_j are the labels of video $\boldsymbol{v}_i$ and video $\boldsymbol{v}_j$. Video $\boldsymbol{v}_j$ is one of the K nearest neighbors of video $\boldsymbol{v}_i$ in the feature space, where the effect of K will be tested in Sec. IV-B. Note that the neighborhood will be changed accordingly when D_g is updated during optimization. Meanwhile, minimizing E_{push_g} achieves the "push" step in order to ensure that the distance between videos $\boldsymbol{v}_l$ and $\boldsymbol{v}_i$ from different classes is larger than the distance between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ from the same class with a margin in the feature space as follows:

$$E_{push_g}(\boldsymbol{\alpha}_g, \boldsymbol{A}_g, \{\boldsymbol{M}^{(i)}\}_{i=1}^{g}) = \sum_{(i,j,l)\in\mathcal{R}} [D_g(\boldsymbol{v}_i, \boldsymbol{v}_j) - D_g(\boldsymbol{v}_i, \boldsymbol{v}_l) + 1]_+, \tag{6}$$

where we denote all tuples for minimizing E_{push_g} as $\mathcal{R} = \{(i, j, l) | (i, j) \in \mathcal{S}, y_i \neq y_l, y_j = y_i\}$.

Individual Segment Loss. In *gapfilling*, some segments might be missing at arbitrary locations. Therefore the global-view feature is not usable in this case. To compensate this, we introduce the *individual segment loss* l_t with respect to each segment:

$$l_t(\boldsymbol{M}^{(t)}) = \mu \varepsilon_{pull_t}(\boldsymbol{M}^{(t)}) + (1 - \mu) \varepsilon_{push_t}(\boldsymbol{M}^{(t)}). \tag{7}$$

Similar to L_g, l_t consists of two parts: the "pull" loss ε_{pull} and the "push" loss ε_{push} defined as below:

$$\varepsilon_{pull_t}(\boldsymbol{M}^{(t)}) = \sum_{(i,j)\in\mathcal{S}} d^{(t)}(\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_j^{(t)}), \tag{8}$$

and

$$\varepsilon_{push_t}(\boldsymbol{M}^{(t)}) = \sum_{(i,j,l)\in\mathcal{R}} [d^{(t)}(\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_j^{(t)}) - d^{(t)}(\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_l^{(t)}) + 1]_+. \tag{9}$$

Successive Segment Loss vs. Individual Segment Loss. The main difference is that in the successive segments loss (Eq. (4)), the distance defined by $D_g(\boldsymbol{v}_i, \boldsymbol{v}_j)$ is a joint distance model formed by the global-temporal and ensemble of local-temporal distances. Therefore the successive segments loss (Eq. (4)) is a loss arising from the collective distance comparing the observed on-going action sequence. In contrast, each *individual* segmental loss (Eq. 7) is specifically defined as a function of each local distance between two corresponding local segments. Minimizing the individual segmental loss is complementary to minimizing the successive segments loss, since the segmental loss is defined based on the features computed from the corresponding segments, and thus unaffected by the the missing segments. Therefore, it is more suitable to address the gapfilling problem.

Algorithm 1 Optimization - Pseudo Code

procedure TRAIN($\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$)
  for each training epoch **do**
    Calculate all losses $l_t(\boldsymbol{M}^{(t)})$ and $L_g(\boldsymbol{\alpha}_g, \boldsymbol{A}_g, \{\boldsymbol{M}^{(i)}\}_{i=1}^g)$
    Update learning rate by comparing the losses to those in last epoch
    for $t = g \in \{1, 2, ..., G\}$ **do**
      Calculate the gradient $\nabla \boldsymbol{M}^{(t)}$
      Calculate the gradient $\nabla \boldsymbol{\alpha}_g$
      Calculate the gradient $\nabla \boldsymbol{A}_g$
      $\boldsymbol{M}^{(t)} \leftarrow \boldsymbol{M}^{(t)} - \eta_{\boldsymbol{M}^{(t)}} \nabla \boldsymbol{M}^{(t)}$
      $\boldsymbol{\alpha}_g \leftarrow \boldsymbol{\alpha}_g - \eta_{\boldsymbol{\alpha}_g} \nabla \boldsymbol{\alpha}_g$
      $\boldsymbol{A}_g \leftarrow \boldsymbol{A}_g - \eta_{\boldsymbol{A}_g} \nabla \boldsymbol{A}_g$
      $\boldsymbol{M}^{(t)} \leftarrow \mathcal{P}_{\mathcal{S}_+}(\boldsymbol{M}^{(t)})$ ▷ Project to $\mathcal{S}_+$
      $\boldsymbol{\alpha}_g \leftarrow \boldsymbol{\alpha}_g^+ / \sum \boldsymbol{\alpha}_g^+$ ▷ Ensure constraints
      $\boldsymbol{A}_g \leftarrow \mathcal{P}_{\mathcal{S}_+}(\boldsymbol{A}_g)$ ▷ Project to $\mathcal{S}_+$

Objective Function. By considering the above two kinds of losses, the final objective function can be written as:

$$L(\{\boldsymbol{\alpha}_i, \boldsymbol{A}_i, \boldsymbol{M}^{(i)}\}_{i=1}^G) = \gamma \cdot \sum_{t=1}^{G} l_t(\boldsymbol{M}^{(t)}) + (1-\gamma) \cdot \sum_{g=1}^{G} L_g(\boldsymbol{\alpha}_g, \boldsymbol{A}_g, \{\boldsymbol{M}^{(i)}\}_{i=1}^g), \quad (10)$$

and the corresponding optimization problem is:

$$\begin{aligned} & \min L(\{\boldsymbol{\alpha}_i, \boldsymbol{A}_i, \boldsymbol{M}^{(i)}\}_{i=1}^G) \quad (11) \\ s.t. \quad & \boldsymbol{\alpha}_i^* \geq 0, \\ & \sum_{t=1}^{i} \boldsymbol{\alpha}_i^{(t)} = 1 \\ & \boldsymbol{A}_*, \boldsymbol{M}^{(*)} \in \mathcal{S}_+ \end{aligned}$$

where γ is a parameter balancing the contribution of successive segments loss and each individual segment loss, and $\mathcal{S}_+$ is the semi-definite matrix space. Parameters are obtained based on minimizing L.

It is worth noting that our distance model formulated in Eq. (3) is quite flexible and can be combined with different types of loss functions. In the experimental section Sec. IV-E1, we have also reported the comparison results obtained by our distance model and other loss functions.

D. Optimization and Inference

The objective function in formula (11) needs to be optimized over a group of parameters including G weight vectors $\boldsymbol{\alpha}_g$, G local metric matrices $\boldsymbol{M}^{(t)}$, and G global metric matrices $\boldsymbol{A}^{(t)}$. To achieve this, we propose an optimization algorithm based on the projected gradient descent approach [34].

1) Initialization of $\boldsymbol{\alpha}_g$: Although $\boldsymbol{\alpha}_g$ can be learned along with $\boldsymbol{M}^{(t)}$, we use a rough estimation of $\boldsymbol{\alpha}_g$ as its initialization. For the t^{th} segment, we roughly estimate its discriminativeness by calculating

$$P^{(t)} = \sum_{i=1}^{N} \sum_{y_i = y_j} \frac{exp(-\|\boldsymbol{x}_i^{(t)} - \boldsymbol{x}_j^{(t)}\|^2)}{\sum_{k \neq i} exp(-\|\boldsymbol{x}_i^{(t)} - \boldsymbol{x}_k^{(t)}\|^2)}. \quad (12)$$

which is inspired from the objective function in [35]. A low $P^{(t)}$ indicates that this segment can hardly tell the difference between actions, and thus less attention should be paid on it. Then $\boldsymbol{\alpha}_g$ can be initialized as

$$\boldsymbol{\alpha}_g^{(i)} = \frac{P^{(i)}}{\sum_{j=1}^{g} P^{(j)}}. \quad (13)$$

2) Optimization of $\boldsymbol{M}^{(t)}$, $\boldsymbol{A}_g$ and $\boldsymbol{\alpha}_g$: We use the gradient decent approach to update $\boldsymbol{M}^{(t)}$, $\boldsymbol{A}_g$ and $\boldsymbol{\alpha}_g$ so that the loss defined by Eq. (10) would decrease. The gradients are computed as follows:

$$\begin{aligned} \nabla \boldsymbol{M}^{(t)} =& \frac{\partial}{\partial \boldsymbol{M}^{(t)}} L(\boldsymbol{M}^{(t)}) \\ =& \mu[\gamma + (1-\gamma) \sum_{g=t}^{G} \boldsymbol{\alpha}_g^{(t)}] \sum_{(i,j) \in \mathcal{S}} \boldsymbol{C}_{i,j}^{(t)} \\ & + (1-\mu)[\gamma \sum_{(i,j,l) \in \mathcal{R}_l^{(t)}} \boldsymbol{C}_{i,j,l}^{(t)} \\ & + (1-\gamma) \sum_{g=t}^{G} \boldsymbol{\alpha}_g^{(t)} \sum_{(i,j,l) \in \mathcal{R}_L^g} \boldsymbol{C}_{i,j,l}^{(t)}], \quad (14) \\ \nabla \boldsymbol{A}_g =& \frac{\partial}{\partial \boldsymbol{A}_g} L_g = \beta[\mu \sum_{(i,j) \in \mathcal{S}} \boldsymbol{C}_{i,j}^{(1,g)} \\ & + (1-\mu) \sum_{(i,j,l) \in \mathcal{R}_L^g} \boldsymbol{C}_{i,j,l}^{(1,g)}], \quad (15) \\ \nabla \boldsymbol{\alpha}_g^{(t)} =& \frac{\partial}{\partial \boldsymbol{\alpha}_g} L_g \\ =& \mu \sum_{(i,j) \in \mathcal{S}} d^{(t)}(\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_j^{(t)}) + (1-\mu) \\ & \sum_{(i,j,l) \in \mathcal{R}_L^g} [d^{(t)}(\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_j^{(t)}) - d^{(t)}(\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_k^{(t)}) + 1]_+, \end{aligned} \quad (16)$$

where $\boldsymbol{C}_{i,j}^{(s,t)} = (\boldsymbol{x}_i^{(s,t)} - \boldsymbol{x}_j^{(s,t)})(\boldsymbol{x}_i^{(s,t)} - \boldsymbol{x}_j^{(s,t)})^T$, $\boldsymbol{C}_{i,j,l}^{(s,t)} = \boldsymbol{C}_{i,j}^{(s,t)} - \boldsymbol{C}_{i,l}^{(s,t)}$, $\mathcal{R}_l^{(t)} = \{(i,j,l) \in \mathcal{R} | d^{(t)}(\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_j^{(t)}) - d^{(t)}(\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_l^{(t)}) + 1 > 0\}$, and $\mathcal{R}_L^{(t)} = \{(i,j,l) \in \mathcal{R} | D_g(\boldsymbol{v}_i, \boldsymbol{v}_j) - D_g(\boldsymbol{v}_i, \boldsymbol{v}_l) + 1 > 0\}$.

Since $\boldsymbol{M}^{(t)}$ and $\boldsymbol{A}_g$ must be positive semi-definite, we need to project the matrix $\boldsymbol{M}^{(t)}$ obtained by a standard gradient descent method onto the positive semi-definite matrix space $\mathcal{S}_+$. Specially, we perform the eigen-decomposition of $\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, where $\boldsymbol{\Lambda}$ is a diagonal matrix with all the eigenvalues being the diagonal terms. Let $\boldsymbol{\Lambda}^+ = max(\boldsymbol{\Lambda}, 0)$ be the diagonal matrix that truncates all negative eigenvalues. Eventually, we have $\mathcal{P}_{\mathcal{S}_+}(\boldsymbol{M}^{(t)}) = \boldsymbol{Q}\boldsymbol{\Lambda}^+\boldsymbol{Q}^T$. The projection

will be performed after every step in order to ensure $\boldsymbol{M}^{(t)}$ and $\boldsymbol{A}_g$ are positive semi-definite. As for $\boldsymbol{\alpha}_g$, we truncate negative value and perform normalization by $\sum \boldsymbol{\alpha}_g^+$ so that the sum will be exactly 1.

3) Inference: In the testing stage, the videos of different observation levels are assumed to be provided manually (or estimated by certain algorithms; please see Section IV-E4 for example.). Once the parameters $\boldsymbol{M}^{(t)}$, $\boldsymbol{A}_g$, and $\boldsymbol{\alpha}_g$ are determined, we then calculate the integrated global-local distance (i.e. Eq. (3)) between training samples and testing samples. Then a k-Nearest Neighbor (KNN) algorithm is used to predict the label of a testing sample. The value of K is the number of nearest neighbors, the same as used in Eq. (5) and Eq. (8) during training. The effect of K is tested in Section IV-E3.

IV. Experiments

Our method is tested on three datasets: UCF11 [17], BIT-Interaction dataset (BIT) [16] and HMDB[18]. The first two were widely used in the action prediction community [8], [9], [11], and HDMB is a new set with a large number of videos. A group of experiments were conducted extensively in the following four settings.

1) The typical action prediction setting [8], where the ending parts of given action sequences are supposed to be unobserved. In this setting, the observation ratio of each ongoing action video is also provided during testing.
2) The same setting as the typical action prediction setting except that the observation ratio is not available when conducting prediction.
3) The gapfilling setting [9], where the middle parts of a sequence are missing.
4) The generalization task, where we directly apply the metric learned from non-target actions to predict related target actions.

A. Action representation

Our model was trained on fully observed videos, and tested on partially observed ones. Therefore, the sequences used for training were fully accessible, and partially observed test sequences were made by varying its observation levels systemically. For each video, we used the STIP (spatio-temporal interest point) [43] and IDT (improved dense trajectory) [44] to extract the spatiotemporal features.

STIP (with HOG/HOF) is a local descriptor capturing local features around selected interest points, while IDT tries to choose a set of salient trajectories to capture the movements depicted in the action. These two features provide both short-term and long-term information about an action and are widely used in video preprocessing[10], [11].

We constructed a visual dictionary for feature type using the K-means algorithm. Then, a bag of words histogram $\boldsymbol{x}^{(t,e)}$ was built to represent each sequence $\boldsymbol{v}^{(t,e)}$. Since we are using histogram features, the explicit kernel mapping for approximating the χ^2 kernel [45] was used to mapping the features into a high dimensional space. Finally, we used PCA to reduce the mapped dimensionality of the histogram to 200 on BIT, or to 800 on UCF11 and HMDB.

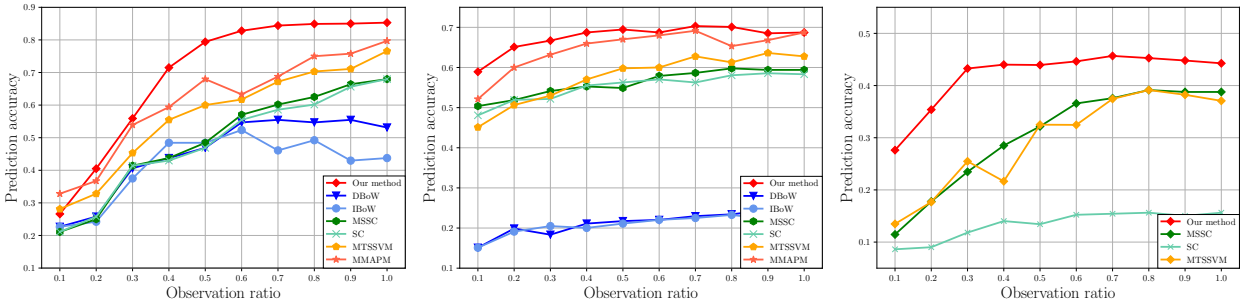
B. Conventional Action Prediction Scenario

In this section, we tested our method under the standard action prediction setting and extensively compared our method with the existing prediction models including DBoW[8], IBoW[8], MSSC[9], SC[9], MTSSVM[10], and MMAPM[11].

BIT-Interaction dataset. The BIT-Interaction dataset contains 8 interactive actions: bow, box, handshake, high-five, hug, kick, pat, and push, with 50 video clips per action class. In each video, only one action instance was captured in a realistic scenario, possibly accompanied with background cluttering or other people as noise. What makes BIT more difficult is that the main actors are often occluded by other people, or cluttered by other irrelevant actions. We followed exactly the same setting as in [11], where 272 videos were randomly picked for training and the rest for testing. We set the parameters $K = 5$, $\mu = 0.9$, $\gamma = 0.90$ and $\beta = 0.20$ through all the experiments on this dataset. Analysis of the effect of key parameters will be presented in Sec. IV-E3. The number of visual words is 500.

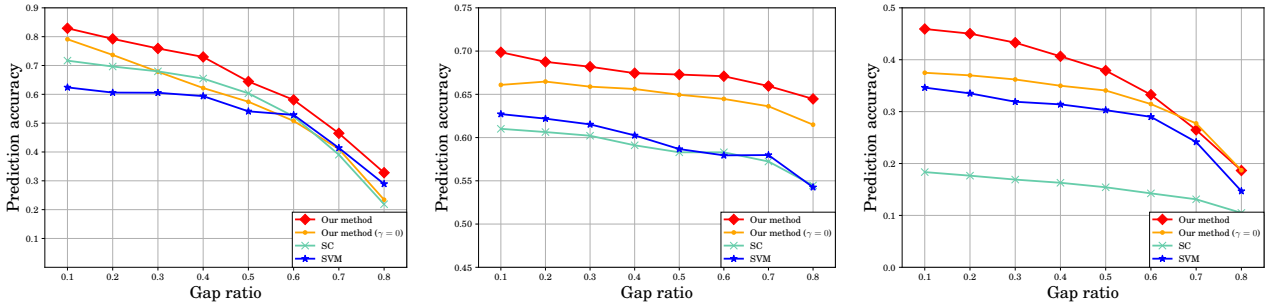
Figure 4(a) shows the comparison results on BIT, and it can be observed that in general, when the observation ratio gets larger, the performances of most competitors increase. This is as expected, because the higher the observation ratio is, the more information we could use for prediction. It is worth noting that our method outperformed the existing methods when the observation ratio drops to as low as 0.5. Those are very promising results. In particular, the prediction accuracy of our method could reach 82.8% at the observation ratio of 0.6, which is close to the accuracy of 85.3% when the sequences are fully observed. When the observation ratio is lower than 0.5, our method performed the best in most cases (or second to the best in a few cases), except at the beginning progress level. An interesting observation is that, when the sequences were fully observed, our method performed significantly better than all other prediction models. This indicates that although our method is specifically designed for action prediction, unlike other models, it copes well with the action recognition of fully observed sequences. It was noted that the methods of DBoW[8], IBoW[8], MSSC[9], and SC[9] treated all segmental representations equally, while ours differs from them in that we use temporal saliency to weight different segments, which may attribute to our superior performance over them. The large margin idea was also explored by MTSSVM[10] and MMAPM[11], similar to our "push" strategy (Eq. 9 and Eq. 6). However, we have introduced "pull" strategy (Eq. 8 and Eq. 5) in the objective function, which contributes in achieving a better overall performance in most cases.

UCF11 dataset. The UCF11 dataset contains 1597 realistic videos of 11 different actions, which were collected from YouTube. All the actions are related to sports: basketball shooting, biking, diving, golf swinging, horse back riding, soccer juggling, swinging, tennis swinging, trampoline jumping, volleyball spiking, and walking with a dog. Hence large degree of camera motion and change of viewpoints, backgrounds and illumination render the dataset challenging in recognition as well as prediction. For each kind of action, there are 25 subsets with more than 4 video clips per class. For a fair comparison,



(a) BIT (b) UCF11 (c) HMDB

Fig. 4: Results for the conventional action prediction task: ours vs. existing action prediction methods. (Best viewed in color)



(a) BIT (b) UCF11 (c) HMDB

Fig. 5: Results for the gapfilling task: ours vs. existing action prediction methods. (Best viewed in color)

we employed the same evaluation setting as in [11], where the first 15 sets of each action class were used for training, and the next 3 sets were for validation and parameter tuning, and the rest 7 sets were used for testing. We set the key hyper-parameters as $K = 11$, $\mu = 0.9$, $\gamma = 0.90$ and $\beta = 1.00$, which were determined by a cross-validation on the validation set. The number of visual words is 500.

Figure 4(b) presents the comparison results on UCF11. As shown, our method outperforms all existing action prediction methods at all observation ratios. Furthermore, when more segments are observed, the prediction accuracies of most of approaches increase slower than that on the BIT set. This is because that salient parts of actions in the BIT set mostly appear in the middle of a video while some of salient parts of actions in the UCF11 set appear at the beginning.

HMDB dataset. The HMDB dataset contains 6849 video clips collected from various sources, mostly from movies, public databases, YouTube and Google videos. There are 51 action classes in this set, which comes from five categories: general facial actions, facial actions with object manipulation, general body movements, body movements with object interaction and body movements for human interaction. Each of the action class contains at least 101 clips. We used the stabilized video to extract IDT features, and the STIP features were provided by the authors of [18]. Also, we followed the splitting policy published along with the videos, where a fraction of samples were saved to adjust hyper-parameter. They were set to be $K = 20$, $\mu = 0.9$, $\gamma = 0.90$ and $\beta = 0.20$ on HMDB. The size of dictionary is 2000 due to the complexity.

Our method was compared with existing competitive models on HMDB, and the results are reported in Figure 4(c). It is shown that our method outperformed others [3] at most of the observation ratios. It was noted that both UCF11 and HMDB were collected in real-world scenarios or from movies. Many actions contain very strong contextual cues and thus their labels can be reliably inferred by observing the context (e.g., a tennis court divulge "playing tennis"). In contrast, most of the actions in BIT were performed in the same background, and thus the motion information is the main cue for prediction.

C. Comparison Under the Gapfilling Scenario

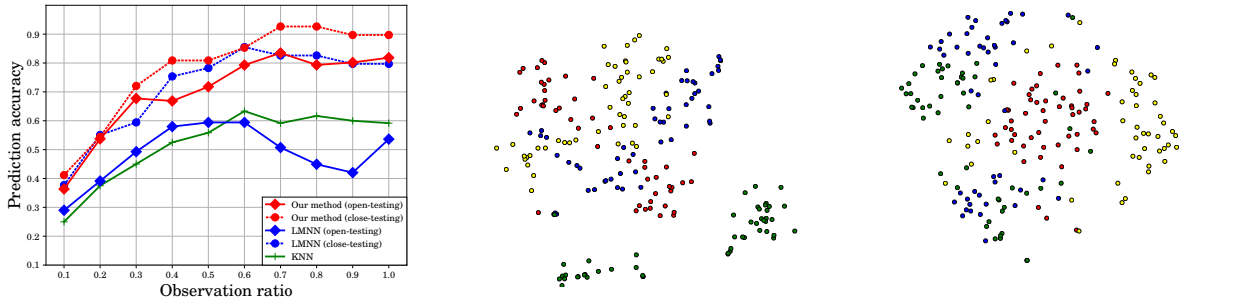
To show that our method can conquer gapfilling better, we conducted our experiments by following the settings in [9], where some segments of action were assumed to be missing.

More specifically, a non-observation interval $[t, e]$ is marked, i.e., the video between the t^{th} and e^{th} segments were unobserved, where $2 \leq t \leq G-1$ and $t \leq e \leq G-1$. The gap ratio r of a partially observed video is defined as the portion of unobserved parts, i.e.,

$$r = \frac{e - t + 1}{G}. \tag{17}$$

The gapfilling problem is simulated by by exhaustively selecting one or a few consecutive segments of any video as the missing segment, the same as in [9]. It is noted that few model can handle gapfilling. Thus, for comparison, we implemented

[3]The results of DBoW and IBoW on HMBD dataset are not presented since the codes of those two methods are not publicly available.



(a) Results on BIT (b) Learned feature space of seen classes (c) Applying learned feature space to unseen classes

Fig. 6: Results of generalization and their visualization. (Best viewed in color)

TABLE II: The effect (%) of temporal saliency policy against the observation ratio ($0.1 \sim 1$) when using our proposed model.

Policy	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
learning-based	**25.8**	**37.5**	**59.4**	**65.6**	**78.1**	**84.4**	**86.7**	**84.4**	**85.9**	**87.5**
uniform	21.9	35.2	53.9	63.3	76.6	81.3	81.3	82.8	83.6	82.8
random	21.1	32.0	43.8	53.9	71.9	77.3	79.7	78.1	78.9	79.7

baselines: SC [9] and a native χ^2 kernel SVM, which are used in action prediction [10], [11]. The accuracy of each method against gap ratio r is reported in Figure 5. It can be observed that our method outperformed the compared methods over all gap ratios on three datasets with a clear margin. Note that we also show the results when only *successive* segment loss was used by setting $\gamma = 0$ in Eq.11 (denoted by *Our method ($\gamma = 0$)* in Figure 5). It could be seen that when adding the *individual* segment loss (denoted by *Our method*), the action prediction performance improved significantly for gapfilling, which confirms the efficacy of the introduced *individual* segment loss in handling the gapfilling problem. It is worth noting that the formulation of the gap-filling problem needs to know the location of the missing segments. In this test, we followed exactly the same assumption as in the existing work [9]. In some practical cases when the missing frames or a set of missing frames (i.e., only frame content is missing) are shown as mosaic of colored patches (e.g., due to coding error), some anomalous detectors [46], [47] would be employed to locate those missing frames[4].

D. Generalization to Unseen Related Actions

Our approach is formulated based on distance metric learning techniques. The metric learned on one set of action classes can be transferable to another task of recognizing related actions in the learning stage, i.e., the metric learned by our model can generalize well to the related actions that have not been seen in the training stage.

In contrast, existing prediction models do not have this property. For example, the SVM-based action prediction models [10], [11] cannot be directly applied to unseen class, because the learned hyperplanes are specific for action type in learning, and there are no hyperplanes available for unseen action types in the training stage.

To evaluate the generalization ability of the action prediction model, we divided all action classes from the BIT dataset into two subsets with no overlap of action types. Only the first subset was used to learn the metric. For testing, one third of samples from the second subset were used as gallery, and the rest were treated as probe samples. In such a setting (termed as *open-setting* in Figure 6(a)), the metric learned in the first subset is applied directly on the second subset to make prediction. When the metric is trained on gallery images and tested on probe images, both from the second subset, we denote it as *close-setting* and plot it with dot-line in Fig 6(a). We compared our method with a LMNN model and KNN under those two setting[5]. The value of K is 5, the same for all cases.

Figure 6(a) shows the results. It was noted that, although there is an expected performance drop of our method from close-setting to the challenging open-setting (i.e., action types unseen in training), the drop is not significant. More importantly, in the open-setting, our method performed much better than the other two methods. All of these suggest our proposed model has good generalization capability in the open-setting.

Using t-SNE[48] with the learned distance metric by our method, we further visualized the distribution of samples used for training (Figure6(b)) (action types are color coded). It is observed that samples from different action types used for training are well separated in the feature space. Figure 6(c) shows the distribution of samples from unseen action types when the learned distance model is applied; and it could be seen that the action types could be distinguished at a good level. This confirms that our learned metric is transferable and thus our model has good generalization capability.

[4]When the missing frames are completely dropped out of the sequence (we would refer this as a frame-dropping problem rather than a gap-filling problem), all of the existing approaches (including ours) are not applicable unless the location of the missing frame information is known.

[5]KNN only operates in close-setting.

TABLE III: Accuracy (%) w.r.t the observation ratio ($0.1 \sim 1$) on BIT datasets; the method named with (u) means the observation ratio is estimated using the algorithm in [11].

methods	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
MMAPM	**32.8**	36.7	53.9	59.4	68.0	63.3	68.8	75.0	75.8	79.7
(u)MMAPM	28.1	32.8	57.0	58.6	68.0	66.4	68.8	72.7	73.4	79.7
Ours	26.6	40.5	**55.9**	**71.5**	**79.4**	**82.8**	**84.4**	**84.9**	**85.0**	**85.3**
(u)Ours	27.6	**44.7**	54.9	63.5	70.9	78.5	83.2	82.2	83.0	85.0

TABLE IV: Performance (%) of our framework with different types of loss functions w.r.t observation ratio ($0.1 \sim 1$).

methods	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Global-Local model with Eq. (11)	**26.6**	**40.5**	**55.9**	**71.5**	**79.4**	**82.8**	**84.4**	84.9	**85.0**	**85.3**
Global-Local model with KL Div	25.8	37.5	54.7	66.4	71.9	78.9	80.5	**85.2**	82.8	82.8

E. Effect of components

*1) **Global-Local Model with KL Div**:* Note that our main idea is to jointly learn and fuse global-local temporal distances in a discriminative model for action prediction. One advantage is that our framework is quite flexible, and capable of embedding other the loss functions.

To demonstrate this potential, we designed and tested a variant of our model with a common loss using KL divergence [36], which minimizes the difference between the estimated distribution and the expected distribution of labels. In this variant, our distance models (Eqs. (1),(2) and (3)) could be optimized directly by minimizing the following objective function:

$$KL[p^*(j|i)|p(j|i)], \tag{18}$$

For each training sample, a conditional distribution over the other samples is defined as

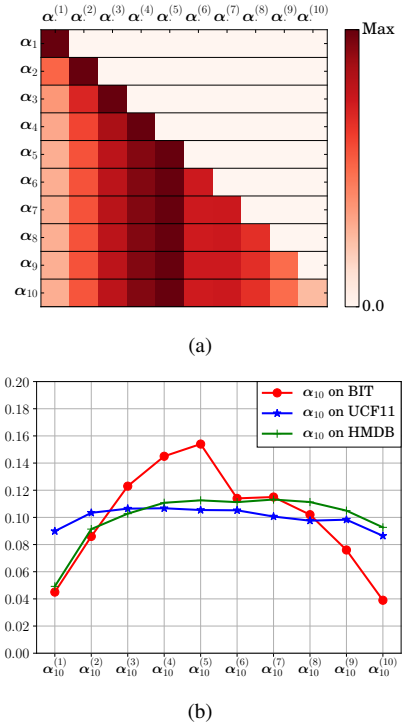
$$p(j|i) = \frac{e^{-d_{ij}}}{\sum_{k \neq i} e^{-d_{ik}}} \quad (i \neq j), \tag{19}$$

$$p^*(j|i) \propto \begin{cases} 1 & y_i = y_j \\ 0 & y_i \neq y_j, \end{cases} \tag{20}$$

where d_{ij} is computed from Eq.(3). The closer two samples are, the smaller d_{ij} is and the larger $p(j|i)$ is. And $p^*(j|i)$ is the ground truth distribution from labels, which is approximated by optimizing the KL divergence in Eq. (18).

Table IV shows the results on BIT, when applying different loss functions in our proposed action prediction framework. The performance of our method is relatively reliable with respect to different loss functions, although with slight difference. It is noticeable that the model with the designed loss function in Eq. (11) performed better at most observation ratios. Similar conclusions could be drawn from other two datasets as presented in our supplementary document.

*2) **Effect of the Temporal Saliency**:* The effect of temporal saliency was tested based on three policies: *uniform*, *learning-based*, and *random*. The *uniform* refers to a policy that all the temporal saliency weights are set as the same value. The *learning-based* means that the saliency is learned by our proposed method. The *random* policy generates the saliency weights randomly.



(a)

(b)

Fig. 7: Visualization of the learned temporal saliency. (a) All temporal saliency $\boldsymbol{\alpha}$ learned on BIT: each row represents one vector of temporal saliency weights $\boldsymbol{\alpha}_g$. (b) The temporal saliency $\boldsymbol{\alpha}_{10}$ learned on BIT, UCF11 and HMDB, respectively when action videos are completely observed. (Best viewed in color)

Results on BIT dataset are tabulated in Table II. It could be seen that the random policy performed worse because of assigning weight arbitrarily, while the learning-based policy outstripped the other two policies mainly because the weight is assigned based on temporal importance of a segment within an action. Interestingly, when the learned saliency by our method is applied to an existing model, SC [9], which originally treated all segment equally, the SC model with our saliency achieved an improvement of 3% on action prediction.

Fig. 8: Four action samples. The first row is from BIT, the second and third rows are from UCF11, and the last is from HMDB.

TABLE V: The prediction (%) accuracy w.r.t observation ratio ($0.1 \sim 1$) with different K

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
K=1	16.0	36.7	54.3	62.5	73.4	77.0	78.1	80.5	79.3	78.5
K=3	18.8	34.0	55.1	66.0	72.3	78.9	82.8	82.0	82.4	80.5
K=5	**26.6**	40.5	55.9	**71.5**	**79.4**	**82.8**	**84.4**	**84.9**	**85.0**	**85.3**
K=7	20.7	**41.0**	**57.4**	65.6	74.2	78.9	80.9	81.2	82.0	80.9
K=9	21.1	38.3	57.0	69.5	75.8	75.8	78.9	78.9	77.3	79.7
K=15	22.7	38.3	53.5	68.4	73.0	77.3	81.2	81.2	80.5	80.5

TABLE VI: The prediction (%) accuracy w.r.t observation ratio ($0.1 \sim 1$) with different β

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
β=0.0	22.7	36.9	**57.6**	66.0	75.4	81.6	81.8	81.6	81.2	82.6
β=0.2	**26.6**	**40.5**	55.9	**71.5**	**79.4**	**82.8**	**84.4**	**84.9**	**85.0**	**85.3**
β=0.4	23.6	38.0	56.9	65.6	75.6	79.8	82.0	82.8	82.8	82.0
β=1.0	22.0	37.2	55.3	64.4	74.2	80.6	81.9	82.0	82.3	81.2
β=10.0	20.8	38.3	50.0	52.6	53.9	61.2	57.6	55.7	62.8	74.2
β=50.0	16.1	31.0	33.9	34.6	33.9	33.3	35.4	37.2	32.0	36.5

Furthermore, we took a deeper look into the temporal saliency we learned. $\boldsymbol{\alpha}_g$ is a vector encoding the importance of local temporal distance models in Eq. (3). If $\boldsymbol{\alpha}_g^{(a)} > \boldsymbol{\alpha}_g^{(b)}$, it means that the a^{th} segment is more important than the b^{th} segment at the g^{th} observation level. We plotted the saliency $\boldsymbol{\alpha}$ learned by our proposed model on BIT in Figure 7(a), where each row represents an $\boldsymbol{\alpha}_g$. The maximum value in each $\boldsymbol{\alpha}_g$ was colored with strong red, while weights close to 0 are colored with light red. It is obvious that different segments were weighed differently at different progress levels.

By investigating $\boldsymbol{\alpha}_{10}$ in Figure 7(b), it can be seen that the last and the first segments of BIT were almost useless. We examined the videos in BIT dataset and found that the most informative action details were always observed in the middle, and it is hard to tell what the person was doing only from frames at the beginning or at the end. Taking the "kicking" in the first row of Figure 8 for example, the most *salient* frames appear in the middle duration. Therefore, on BIT, the temporal saliency weight curve has a peak as shown in Figure 7(b), which means certain segments are more informative for prediction. Our experimental results also suggested that weighting all segments equally on BIT would worsen the prediction performance, as shown in Figure 4(a). In comparison, the temporal saliency weight curve seems flatter on UCF11 in Figure 7(b). It is because every segment of a video is informative to describe the action details. For example, it can be clearly seen that the "jumping" appears in each segment as shown in the second row of Figure 8. Similar observation can be found on "walking", "cycling" and etc.

*3) **Effect of** K **and** β :* We investigated the effect of hyper-parameters: K (its value was set as the same in both the training and testing stage of our model) and β (in Eq.(3)). The results on BIT are reported and similar conclusion can be drawn on the other two datasets (please see the supplementary materials for detailed results on the other two datasets). When one of hyper-parameters was tested, the others were fixed as their default values.

K represents the number of nearest neighbors used in training as well as testing. In Table V, we compared the prediction accuracy of our model with different K. It could be seen that increasing the value of K will generally increase the accuracy. However, the performances will become saturated when K is relatively large (e.g., $K \geq 5$), which means that increasing K further does not improve the performance but with higher computational cost. It is worth noting that the optimal choice of K is very reliable against the observation ratios.

β describes how much the global temporal distance in Eq. (1) contributes to the final distance in Eq. (3) between two samples. When $\beta >> 1$, the global-temporal distance dominates, which means that our model pays more attention to global features than local details. From Table VI, it can be seen

TABLE VII: Prediction accuracy (%) based on using LSTM against the observation ratio ($0.1 \sim 1.0$) on UCF11 and HMDB

Dataset	Loss Function	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
UCF11	Softmax	**68.9**	**70.6**	72.1	72.6	73.0	73.4	74.9	75.5	76.0	77.2
	Softmax+Our Metric	68.5	70.2	**73.0**	**74.5**	**74.3**	**76.6**	**77.9**	**77.9**	**78.3**	**79.1**
HMDB	Softmax	20.5	29.6	33.5	36.7	38.8	40.4	41.7	42.3	42.8	43.4
	Softmax+Our Metric	**24.6**	**34.6**	**37.3**	**39.1**	**41.6**	**44.1**	**45.1**	**47.6**	**49.0**	**50.5**

TABLE VIII: Prediction accuracy (%) based on using two-stream network against the observation ratio ($0.1 \sim 1.0$) on UCF11 and HMDB

Dataset	Loss Function	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
UCF11	Softmax	85.3	86.0	88.1	89.6	89.6	88.9	88.9	88.5	89.6	89.4
	Softmax+Our Metric	**87.5**	**89.3**	**90.2**	**90.2**	**90.5**	**90.9**	**90.5**	**91.0**	**90.9**	**91.1**
HMDB	Softmax	36.4	41.9	45.3	49.0	51.1	52.7	54.6	55.1	55.4	55.9
	Softmax+Our Metric	**38.8**	**43.8**	**49.1**	**50.4**	**52.6**	**54.7**	**56.3**	**56.9**	**57.3**	**57.3**

that the model with a very large β has poor performance since global temporal features alone are not sufficient. However, by comparing the performance at $\beta = 0.0$ versus $\beta = 0.2$ (or $\beta = 0.4$), we can see that including appropriate proportion of global-temporal feature is shown useful.

*4) **Observation ratio unavailable**:* In the previous experiments, we have assumed that the observation ratio of each action sequence is available during testing, which was also adopted by most of the existing prediction models ([8], [9], [11]). However, in real scenarios, it is not easy to obtain the observation ratio. To address this problem, we followed [11] to train an action progressing predictor for estimating the observation ratio of ongoing sequences. The estimated ratio was then used in our prediction model. The deviation of the estimated ratio against the real one is within $\pm 10\%$ on BIT. We report the performance of our model and MMAPM in Table III, which is obtained using the estimated observation ratio. It is shown that our method outperformed MMAPM [11] in this case, and the estimated observation ratio does not have a significant effect on the system performance. It is also interesting to note that with the estimated observation ratio, the model performed comparably to that using the pre-defined ratio.

F. Compatibility with Deep Action Models

Deep neural networks have been empirically proved to be good at extracting temporal features on sequences as compared to conventional hand-crafted temporal features in action recognition [49], [4], [50], [51], [52], [53], [54], [55], [54]. In this section, we show that our proposed metric is not only compatible with deep models and but also leads to improved performance. More specifically, we augment a typical neural network (e.g. two-stream [29], [4] and LSTM [56]) with the proposed loss function in Equ. (10), which acts as a part of the loss of the neural network. In action recognition task, existing deep neural networks were tested on the videos that contain instances of the complete action. In order to employ those networks under action prediction scenario, we have to limit the sample range to the observed segments. And we combine the prediction over multiple segments by averaging them as

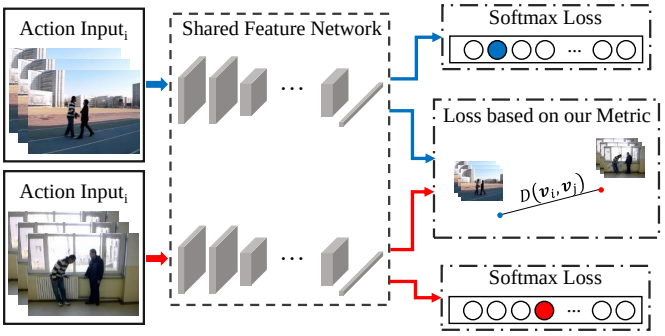


Fig. 9: Illustration of the extended deep neural network using our metric. When training a network (e.g. the two-stream framework), apart from the traditional softmax loss, we sample pairs and calculate the loss based on our metric used in Equ. (10).

similarly done in [29]. All experiments in this section were conducted on UCF11 and HMDB51. It was noted that the size of BIT very small. Thus a deep neural network (e.g., LSTM) working on BIT will need to be carefully fine-tuned based on a pre-trained model on extra data. Although this is out of our focus, we include some evaluations of the effects our metric on the performance of a LSTM on BIT pretrained on extra data in the supplementary material, which shows that our metric still works on a pretrained LSTM on BIT. Note that we are not focusing on developing a state-of-the-art deep structure but on exploring the extensibility of our proposed metric when used as an auxiliary loss function to optimize a deep neural network.

Recurrent neural network (RNN), especially LSTM [56], is widely used to handle sequential tasks because its ability to integrated temporally rated data non-linearly [49]. Therefore we implemented a LSTM-Dense Network with 2 layers of LSTM (3000 and 200 cells respectively) followed by a 500-neurons dense layer to tackle the action prediction problem based on the hand-crafted features. The dropout layer [57] was introduced to alleviate overfitting and we used Adadelta with learning rate of 10^{-3} [58] and a weight decaying rate of

TABLE IX: Prediction Accuracy (%) under Gapfilling based on using two-stream network against the gap ratio ($0.2 \sim 0.8$) on UCF11 and HMDB

Dataset	Loss Function	0.2	0.3	0.4	0.5	0.6	0.7	0.8
UCF11	Softmax	88.9	88.6	88.4	88.1	88.5	87.9	86.8
	Softmax+Our Metric	**90.9**	**90.7**	**90.8**	**90.5**	**90.0**	**89.8**	**89.4**
HMDB	Softmax	55.0	54.0	52.5	51.2	49.1	46.5	43.5
	Softmax+Our Metric	**56.3**	**55.5**	**53.9**	**52.3**	**50.2**	**47.9**	**44.9**

10^{-4}. The dropping ratio was 0.8 for the decision layer and 0.5 for other layers. In Table VII, we used LSTM to achieve a better performance when further incorporating our proposed metric to quantify the deep neural network. On UCF11, the improvement of performance when incorporating our metric increases when the observation ratio becomes larger, and on HMDB, further incorporating our metric will always yield an improvement of at least 4% on all observation ratios. The results show that our approach could improve the performance of LSTM.

We also tested the performance of the-state-of-the-art two-stream framework [29], [4] based on ResNet-50 [59] and concatenate-fusion architecture. We also augmented a loss function formed by our proposed metric with the two-stream framework and evaluated its performance for comparison. Our network was pre-trained on ImageNet [60] and we further decreased the feature dimension to 800 with a fully connected layer for applying our metric. The comparison results of prediction on UCF11 and HMDB datasets are shown in Table VIII. As shown our metric could further improve the performance of the two-stream framework.

In Table IX, we also show that two-stream networks can be easily applied to adapt the setting of gapfilling. LSTM is not reported under gapfilling scenario since it requires a continuous input. Indeed, the performance drop becomes smaller wrt gap ratio using deep learning framework as compared to the results in Figure 5. When further incorporating our metric, the performance drop is alleviated, which is more obvious on UCF11. This is possibly because our proposed individual segment loss in Equ. (7) enables each individual segment to be discriminative.

V. Conclusion

In this paper, we have developed an action prediction method based on a global-local temporal distance model, which introduces the temporal saliency for adapting the contribution of each local-temporal distance. The formulated global-local temporal action prediction model jointly learns and fuses two types of distances: the local-temporal distance and the global-temporal distance. Extensive experimental results show that our method outperforms existing models in recognizing actions of ongoing sequences. We further showed that the proposed model tackles the gapfilling problem better than the compared methods, and the learned metric could be transfered to action types unseen in the training stage. Currently, most existing action prediction efforts focus on a single type of action in one sequence. In future, we will consider action prediction when multiple actions present simultaneously in a video sequence.

Acknowledgments

This work was supported partially by the National NSFC (No. 61472456, No. 61522115, No. 61702567) and the Macau Science and Technology Development Fund of (No. 019/2014/A1, No. 112/2014/A3).

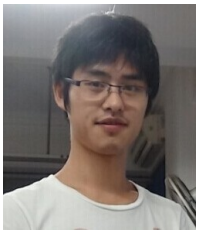
References

[1] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *ICPR*, 2004.
[2] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *CVPR*, 2008.
[3] Y. Kong, Y. Jia, and Y. Fu, "Interactive phrases: Semantic descriptionsfor human interaction recognition," *TPAMI*, vol. 36, no. 9, pp. 1775–1788, 2014.
[4] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *CVPR*, 2016.
[5] C. Jia, Y. Kong, Z. Ding, and Y. Fu, "Rgb-d action recognition," in *Human Activity Recognition and Prediction*. Springer International Publishing, 2016, pp. 87–106.
[6] B. Mahasseni and S. Todorovic, "Regularizing long short term memory with 3d human-skeleton sequences for action recognition," in *CVPR*, 2016.
[7] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj, "Beyond gaussian pyramid: Multi-skip feature stacking for action recognition," in *CVPR*, 2015.
[8] M. Ryoo, "Human activity prediction: Early recognition of ongoing activities from streaming videos," in *ICCV*, 2011.
[9] Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. Siskind, and S. Wang, "Recognize human activities from partially observed videos," in *CVPR*, 2013.
[10] Y. Kong, D. Kit, and Y. Fu, "A discriminative model with multiple temporal scales for action prediction," in *ECCV*, 2014.
[11] Y. Kong and Y. Fu, "Max-margin action prediction machine," *TPAMI*, 2015.
[12] M. Ryoo and L. Matthies, "First-person activity recognition: What are they doing to me?" in *CVPR*, 2013.
[13] M. Ryoo, T. J. Fuchs, L. Xia, J. K. Aggarwal, and L. Matthies, "Robot-centric activity prediction from first-person videos: What will they do to me'," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction (2015)*.
[14] T. Lan, T.-C. Chen, and S. Savarese, "A hierarchical representation for future action prediction," in *ECCV (2014)*.
[15] Z. Xu, L. Qing, and J. Miao, "Activity auto-completion: Predicting human activities from partial videos," in *ICCV*, 2015.
[16] Y. Kong, Y. Jia, and Y. Fu, "Learning human interaction by interactive phrases," in *ECCV*, 2012.
[17] J. Liu, J. Luo, and M. Shah, "Recognizing realistic actions from videos in the wild," in *CVPR*, 2009.
[18] J. H. Kuehne, H., E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *ICCV*, 2011.
[19] B. Wu, C. Yuan, and W. Hu, "Human action recognition based on context-dependent graph kernels," in *ICCV*, 2014.
[20] M. S. Ryoo and J. K. Aggarwal, "Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities," in *ICCV*, 2009.
[21] X. Wu, D. Xu, L. Duan, and J. Luo, "Action recognition using context and appearance distribution features," in *CVPR*, 2011.

[22] J. Liu, B. Kuipers, and S. Savarese, "Recognizing human actions by attributes," in *CVPR*, 2011.
[23] G. Cheron, I. Laptev, and C. Schmid, "P-cnn: Pose-based cnn features for action recognition," in *ICCV*, 2015.
[24] J. F. Hu, W. S. Zheng, J. H. Lai, and J. Zhang, "Jointly learning heterogeneous features for rgb-d activity recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2017.
[25] Y. Yang and M. Shah, "Complex events detection using data-driven concepts," in *ECCV*, 2012.
[26] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification," in *ECCV*, 2010.
[27] A. Vahdat, K. Cannons, G. Mori, S. Oh, and I. Kim, "Compositional models for video event detection: A multiple kernel learning latent variable approach," in *ICCV*, 2013.
[28] Q. Shi, L. Cheng, L. Wang, and A. Smola, "Human action segmentation and recognition using discriminative semi-markov models," *IJCV*, vol. 93, no. 1, pp. 22–32, 2011.
[29] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014.
[30] C. Vondrick, H. Pirsiavash, and A. Torralba, "Anticipating visual representations from unlabeled video," in *CVPR*, 2016.
[31] K. Li and Y. Fu, "Prediction of human activity by discovering temporal sequence patterns," *TPAMI*, vol. 36, no. 8, pp. 1644–1657, 2014.
[32] J.-F. Hu, W.-S. Zheng, L. Ma, G. Wang, and J. Lai, "Real-time rgb-d activity prediction by soft regression," in *ECCV*, 2016.
[33] R. De Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars, "Online action detection," in *ECCV*, 2016.
[34] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *NIPS*, 2015.
[35] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood component analysis," *NIPS*, 2004.
[36] A. Globerson and S. T. Roweis, "Metric learning by collapsing classes," in *NIPS*, 2005.
[37] P. Jain, B. Kulis, J. V. Davis, and I. S. Dhillon, "Metric and kernel learning using a linear transformation," *JMLR*, vol. 13, pp. 519–547, 2012.
[38] K. Etemad and R. Chellappa, "Discriminant analysis for recognition of human face images," *JOSA A*, vol. 14, no. 8, pp. 1724–1733, 1997.
[39] X. Li, C. Shen, Q. Shi, A. Dick, and A. Van den Hengel, "Non-sparse linear representations for visual tracking with online reservoir metric learning," in *CVPR*, 2012.
[40] W.-S. Zheng, S. Gong, and T. Xiang, "Reidentification by relative distance comparison," *TPAMI*, vol. 35, no. 3, pp. 653–668, 2013.
[41] X. Gao, S. C. Hoi, Y. Zhang, J. Wan, and J. Li, "Soml: Sparse online metric learning with application to image retrieval." in *AAAI*, 2014.
[42] J. V. Davis and I. S. Dhillon, "Structured metric learning for high dimensional problems," in *ACM SigKDD*, 2008.
[43] I. Laptev, "On space-time interest points," *IJCV*, vol. 64, no. 2-3, pp. 107–123, 2005.
[44] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *ICCV*, 2013.
[45] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *TPAMI*, vol. 34, no. 3, pp. 480–492, 2012.
[46] A. C. Kokaram, R. D. Morris, W. J. Fitzgerald, and P. J. Rayner, "Detection of missing data in image sequences," *TIP*, vol. 4, no. 11, pp. 1496–1508, 1995.
[47] A. C. Kokaram and S. J. Godsill, "Mcmc for joint noise reduction and missing data treatment in degraded video," *TSP*, vol. 50, no. 2, pp. 189–205, 2002.
[48] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *JMLR*, vol. 9, no. Nov, pp. 2579–2605, 2008.
[49] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014.
[50] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *TPAMI*, 2013.
[51] G. Gkioxari, R. Girshick, and J. Malik, "Contextual action recognition with r* cnn," in *ICCV*, 2015.
[52] M. S. Ibrahim, S. Muralidharan, Z. Deng, A. Vahdat, and G. Mori, "A hierarchical deep temporal model for group activity recognition," in *CVPR*, 2016.
[53] M. Ravanbakhsh, H. Mousavi, M. Rastegari, V. Murino, and L. S. Davis, "Action recognition with image based cnn features," *arXiv preprint arXiv:1512.03980*, 2015.
[54] F. Turchini, L. Seidenari, and A. Del Bimbo, "Understanding and localizing activities from correspondences of clustered trajectories," *CVIU*, 2016.
[55] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *CVPR*, 2015.
[56] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.
[57] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *JMLR*, 2014.
[58] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.
[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.
[60] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, 2015.

Shaofan Lai received his B.S. degree in computer science from the School of Data and Computer Science, Sun Yat-Sen University in 2017. He is currently pursuing his M.S. degree in computer science at the University of Southern California. His research interests include visual surveillance, personal re-identification and generative adversarial models.

Wei-Shi Zheng is now a professor at Sun Yat-sen University. He had been a postdoctoral researcher on the EU FP7 SAMURAI Project at Queen Mary University of London and an associate professor at Sun Yat-sen University after that. He has now published more than 90 papers, including more than 60 publications in main journals (TPAMI,TNN,TIP,TSMC-B,PR) and top conferences (ICCV, CVPR,IJCAI,AAAI). He has joined the organisation of four tutorial presentations in ACCV 2012, ICPR 2012, ICCV 2013 and CVPR 2015 along with other colleagues. His research interests include person association and activity understanding in visual surveillance. He has joined Microsoft Research Asia Young Faculty Visiting Programme. He is a recipient of excellent young scientists fund of the national natural science foundation of China, and a recipient of Royal Society-Newton Advanced Fellowship.



Jian-Fang Hu received the PhD and B.S. degrees from the School of Mathematics, Sun Yat-Sen University, Guangzhou, China, in 2016 and 2010, respectively. His research interests include human-object interaction modeling, 3D face modeling, and RGB-D action recognition. He has published several scientific papers in the international conferences and journals including ICCV, CVPR, ECCV, IEEE TPAMI, IEEE TCSVT, and PR.

Jianguo Zhang is currently a Reader at Computing in the School of Science and Engineering, University of Dundee, UK. He received a PhD in National Lab of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, 2002. His research interests include visual surveillance, object recognition, image processing, medical image analysis and machine learning.