Deep Sequential Context Networks for Action Prediction

Yu Kong[1] Zhiqiang Tao[1] Yun Fu[1,2]
[1]Department of Electrical and Computer Engineering,
[2]College of Computer and Information Science.
Northeastern University, Boston, MA 02115, USA
{yukong,zqtao,yunfu}@ece.neu.edu

Abstract

This paper proposes efficient and powerful deep networks for action prediction from partially observed videos containing temporally incomplete action executions. Different from after-the-fact action recognition, action prediction task requires action labels to be predicted from these partially observed videos. Our approach exploits abundant sequential context information to enrich the feature representations of partial videos. We reconstruct missing information in the features extracted from partial videos by learning from fully observed action videos. The amount of the information is temporally ordered for the purpose of modeling temporal orderings of action segments. Label information is also used to better separate the learned features of different categories. We develop a new learning formulation that enables efficient model training. Extensive experimental results on UCF101, Sports-1M and BIT datasets demonstrate that our approach remarkably outperforms state-of-the-art methods, and is up to $300\times$ *faster than these methods. Results also show that actions differ in their prediction characteristics; some actions can be correctly predicted even though only the beginning* 10% *portion of videos is observed.*

1. Introduction

Human action recognition has gained significant interests due to its broad applications, such as visual surveillance and video retrieval. This task is to infer the action label **after** the entire action execution has been observed. However, in some scenarios, predicting the action label **before** the action execution ends is extremely important. For example, it would be very helpful if an intelligent system on a vehicle can predict a traffic accident before it happens; opposed to recognizing the dangerous accident event thereafter. More importantly, it is essential that the intelligent system can make accurate predictions at the very beginning stage of a video, for instance, when only the beginning 10% frames of a full video is observed.

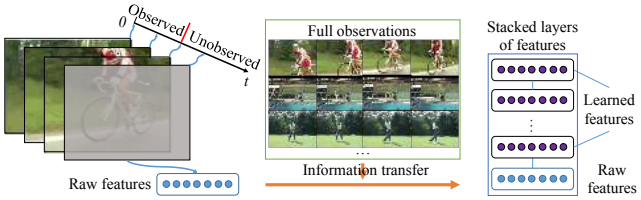


Figure 1. Our DeepSCN predicts the action label given an unfinished action video. Given features extracted from a partially observed video, DeepSCN gains extra discriminative information from fully observed video. Multiple layers of learned features (purple nodes) are stacked to better abstract raw features.

Action prediction is challenging because decisions must be made based on temporally incomplete action executions. However, certain actions are predictable at their early stage if particular temporal patterns are observed and temporal context is available. Consider, for example, a video of a triple jump. We could imagine that a player is very likely to jump after running since we have seen this type of sport elsewhere[1]. The sequential context of the full video observation provides us with the knowledge that the triple jump action consists of running and jumping, and how the action appearance evolves in the temporal domain. This crucial information transferred along temporal axis is the key to action prediction as it helps us to understand the action evolution in the full action observation.

In this paper, we propose novel deep sequential context networks (DeepSCN) for action prediction. DeepSCN is based on marginalized stacked autoencoder (MSDA) [4], but extends it to accommodate sequential data. Our deep networks utilize rich sequential context information to better capture the appearance evolution and temporal structure of the full video observations. We learn such information from full videos (see Figure 1), and transfer it to the features extracted from partial videos containing temporally incomplete action executions. This enriches the feature representations, and improves their discriminative power even

[1]We acknowledge that the scene gist also plays an important role here.

though they are extracted from incomplete sequences. The amount of the transferred information at different progress levels[2] is temporally ordered for the purpose of modeling the temporal orderings of inhomogeneous action segments. Furthermore, supervisory information is incorporated in DeepSCN in order to improve the discriminative ability of the learned features at different progress levels.

DeepSCN has numerous advantages inherited from MSDA: 1) DeepSCN is remarkably faster than existing prediction approaches [13, 2] in training and testing. Empirical results show that DeepSCN is up to $300\times$ faster than [2] and $60\times$ faster than [13]; 2) DeepSCN stacks multiple layers into a deep network that helps effectively summarize action features at different progress levels; 3) it enjoys layer-wise convexity and can be solved efficiently.

Our work focuses on *short-duration* prediction such as "biking" and "diving", while [17] focuses on long-duration compositional action prediction where an action can be further decomposed into semantic meaningful primitives. For example, an activity "make an omelet" can be decomposed into primitives "crack", "pour", "stir", etc. We simulate sequential data arrival while [2] assumes data are randomly observed in a sequence. We aim at predicting the label for a partially observed video. By comparison, [35, 21, 14] predict what will happen in the future, and [9, 19] localizes the starting and ending frames of an incomplete event.

2. Related Work

Action recognition methods take as input fully observed videos and output labels of human actions. Existing approaches can be roughly categorized into low-level feature-based approaches [15, 33, 30, 22, 31, 20] and mid-level feature-based approaches [12, 26, 34, 36, 18]. Low-level features, such as dense trajectory [31] and poselet key-frames [22], utilize local appearance information and spatio-temporal structures, and have shown great success in action recognition. Mid-level feature-based approaches, such as semantic descriptions [12] or data-driven concepts [34], have shown to be capable of recognizing more complex human actions. Furthermore, some deeply learned features [32, 6] were recently proposed to learn high-level information for classification. However, most existing methods expect to observe temporally complete action executions. Their performances are unknown if they are given videos with temporally incomplete action executions.

Action prediction methods [24, 2, 14, 13] were proposed to predict the action given a partially observed video. Ryoo [24] proposed integral and dynamic bag-of-words approaches for action prediction. The former one models feature distribution variations over time, while the latter technique depicts the sequential nature of human activities. Cao *et al.* [2] generalized human activity recognition. In their work, frames were randomly removed in a video to simulate missing data. They formulated the problem as a posterior-maximization problem, where the likelihood is computed by feature reconstruction error using sparse coding. However, [2] suffers from high computational complexity as the inference is performed on the entire training data. Lan *et al.* [14] designed a coarse-to-fine hierarchical representation to capture the discriminative human movement at different levels, and used a max-margin framework for final prediction. Kong *et al.* [13] proposed a structured SVM learning method to simultaneously consider both local and global temporal dynamics of human actions. By enforcing a label consistency of temporal segments, the performance of prediction can be effectively improved.

The proposed approach is significantly different from existing action prediction and early detection approaches [24, 13, 2, 14, 9, 19]. The proposed DeepSCN elegantly gains extra sequential context information from full videos to partial videos, while [13, 9, 19] capture increasing confidence score or decreasing detection loss in temporal sequence. Action models are computed by averaging action representations in training data [24], building action dictionaries [2] or describing actions at both coarse and fine levels [14]. By comparison, we build action models by transferring information from full videos in order to improve the discriminative power of partial videos. In addition, our approach stacks multiple feature layers to better summarize action features, while [24, 2, 14, 13] only use hand-crafted features.

The prediction of future events was also investigated in other applications, such as predicting events in recommendation systems [16, 23], predicting future visual representation [29], and reasoning about the preferred path for a person [11, 1]. Their goals are different from our work as we focus on predicting the action labels of a video.

3. Our Approach

Our goal is to predict the action class y of an action video $\mathbf{x}$ before the ongoing action execution ends [13, 2, 24]. We follow the problem setup described in [13, 2, 24, 14]. A complete video $\mathbf{x}$ containing T frames is uniformly segmented into K segments ($K = 10$ in this work), mimicking sequential video arrival at various observation ratios. Each segment contains $\frac{T}{K}$ frames. Note that for different videos, their lengths T may vary, causing different lengths in their segments. The k-th segment ($k \in \{1, \cdots, K\}$) of the video ranges from the $[(k-1)\cdot\frac{T}{K}+1]$-th frame to the $(\frac{kT}{K})$-th frame. A temporally *partial video* or *partial observation* $\mathbf{x}^{(k)}$ is a temporal subsequence that contains the beginning k out of K segments of the video. The *progress level* g of the partial video $\mathbf{x}^{(k)}$ is k: $g = k$, and its *observation ratio*

[2]The progress level k is the number of observed temporal segments in a video, ranging from 1 to K: $k \in \{1, \cdots, K\}$, where K is the total number of segments in a full video.

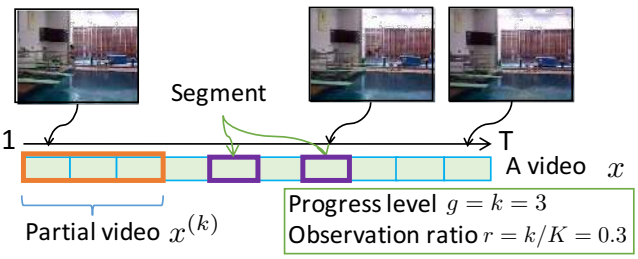


Figure 2. Example of a temporally partial video, and graphical illustration of progress level and observation ratio.

r is $\frac{k}{K}$: $r = \frac{k}{K}$ (see Figure 2). For a given partial video, its progress level (or level) g and observation ratio r have $g = r \times K$.

Given N training videos $\{\mathbf{x}_i, y_i\}_{i=1}^N$, we simulate sequential data arrival in this work, and temporally decompose each training video $\mathbf{x}_i$ into partial observations $\{\mathbf{x}_i^{(k)}\}|_{k=1}^K$ at various progress levels. Note that $\mathbf{x}_i^K$ and $\mathbf{x}_i$ are the same full video: $\mathbf{x}_i^K = \mathbf{x}_i$. We would like to learn a feature mapping function $G : \mathbf{x}^{(k)} \rightarrow \mathbf{z}$ and a prediction function $F : \mathbf{z} \rightarrow y$, where $\mathbf{x}^{(k)} \in \mathcal{R}^d$ is a partial video at progress level k, $\mathbf{z} \in \mathcal{R}^D$ is the learned feature vector with high discriminative power, and $y \in \mathcal{Y}$ is the action label.

Visual Features: Our approach works with both deep features and handcrafted features. We extract C3D features [27], or spatiotemporal interest points [5] and dense trajectory features [30] from a partial video $\mathbf{x}^{(k)}$. Bag-of-words model is used to encode handcrafted features.

3.1. Single-layer Feature Learning

Sequential context. As shown in [13, 2], it is essential to improve the discriminative power of features extracted from partial observations in order to achieve high prediction performance. This is even more important for predicting the beginning portion of a video since a large amount of useful cues for classification are not observed in the early stage of the video. Furthermore, the features extracted from the beginning portion of a video cannot fully convey the information of the entire video.

Intuitively, people are more confident about the action category if more frames are observed. Recent studies [13, 2, 24, 14] show that the best prediction performance is generally made when all the frames are observed. This suggests that full observations contain all the useful information for classification. Motivated by this observation, in this work, we propose to improve the discriminative power of partial videos by gaining extra information from full videos. Our assumption is that if the features from a partial video can be geometrically close to the features from the full video, then their discriminative abilities would be similar. We define the discrepancy between partial observations $\{\mathbf{x}_i^{(k)}\}$ and their corresponding full observations $\{\mathbf{x}_i^{(K)}\}$ as

$$\sum_{i=1}^{N}\sum_{k=1}^{K} \|\mathbf{x}_i^{(K)} - \mathbf{W}\mathbf{x}_i^{(k)}\|_2^2 = \|\bar{\mathbf{X}}^{(K)} - \mathbf{W}\bar{\mathbf{X}}\|_F^2, \quad (1)$$

where $\mathbf{W}$ is a feature transformation matrix of size $d \times d$ learned during model training, and $\|\cdot\|_F$ is the Frobenius norm. Matrices $\bar{\mathbf{X}}^{(K)}$ is a $d \times KN$ matrix containing all the full observations and $\bar{\mathbf{X}}$ is also a $d \times KN$ matrix containing all the partial observations:

$$\begin{aligned} \bar{\mathbf{X}}^{(K)} &= (\underbrace{\mathbf{x}_1^{(K)}, \cdots, \mathbf{x}_1^{(K)}}_{K \text{ times}}, \cdots, \mathbf{x}_N^{(K)}, \cdots, \mathbf{x}_N^{(K)}), \\ \bar{\mathbf{X}} &= (\mathbf{x}_1^{(1)}, \cdots, \mathbf{x}_1^{(K)}, \cdots, \mathbf{x}_N^{(1)}, \cdots, x_N^{(K)}). \end{aligned} \quad (2)$$

By minimizing the discrepancy defined in Eq. (1), a partial observation $\mathbf{x}_i^{(k)}$ is mapped onto a feature space using the learned projection matrix $\mathbf{W}$ under the guidance of its corresponding full observation $\mathbf{x}_i^K$ or $\mathbf{x}_i$. The reconstructed feature $\mathbf{W}\mathbf{x}_i^{(k)}$ is expected to be geometrically closer to its corresponding full observation $\mathbf{x}_i^K$. Therefore, the learned feature vector $\mathbf{W}\mathbf{x}_i^{(k)}$ will gain extra crucial information for action prediction from the full observation $\mathbf{x}_i^{(K)}$, and its discriminative power is thus enhanced.

Note that we use one single feature transformation matrix W here rather than using K transformation matrices, one for each progress level, in order to make our approach compact and practical in testing. If we use K transformation matrices, then we need to know the progress level k of a testing video to pick the right $\mathbf{W}$, which is infeasible in practical scenario.

Robust features. During information transfer, noise could be introduced to partial observations, which may degrade the prediction performance. We overcome this problem by regularizing $\mathbf{W}$ and constructing robust features for partial videos that are insensitive to noise. Recent work in robust feature learning [4, 28] shows that robust features should be able to be reconstructed from partial and random corruption. Inspired by this idea, we reconstruct features of partial observations with the mapping matrix $\mathbf{W}$:

$$\sum_{i=1}^{N}\sum_{k=1}^{K} \|\mathbf{x}_i^{(k)} - \mathbf{W}\tilde{\mathbf{x}}_i^{(k)}\|_2^2 = \|\bar{\mathbf{X}} - \mathbf{W}\tilde{\mathbf{X}}\|_F^2, \quad (3)$$

where $\tilde{\mathbf{x}}_i^{(k)}$ is the corrupted version of the original data $\mathbf{x}_i^{(k)}$ obtained by setting a fraction of the feature vector $\mathbf{x}_i^{(k)}$ to 0 with probability $p \geqslant 0$. Matrix $\tilde{\mathbf{X}}$ is the corrupted version of $\bar{\mathbf{X}}$ defined as

$$\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1^{(1)}, \cdots, \tilde{\mathbf{x}}_1^{(K)}, \cdots, \tilde{\mathbf{x}}_N^{(1)}, \cdots, \tilde{\mathbf{x}}_N^{(K)}). \quad (4)$$

To reduce data variance, "infinite" passes of corruptions are performed over the training data [4].

Effectively using "infinitely" many copies of noisy data allows us to learn features robust to noise [4]. In this work, noise may originate from two sources, a partial video itself and the corresponding full video. Although current feature extractors (e.g., C3D and spatiotemporal interest point detectors) have shown to be robust to background noise, the detected features may still be associated with dynamic background that is irrelevant to human actions. In addition, features extracted from human body may also suffer from illumination changes, pose and appearance variations, and camera jittering, etc. This will undoubtedly degrade the representation power of features and further cause significant reduction in prediction performance. By setting random elements in the feature vector $\mathbf{x}_i^k$ to 0, these feature elements are removed in the feature vector. This essentially simulates appearance variations in videos, and thus helps in learning robust features. This scheme can be considered as a mask-out/drop-out regularization [8, 3].

Label information. Partial observations in the same category may vary greatly in appearance, duration, etc. Therefore, the learned prediction model may not be able to capture complex classification boundaries. We address this problem by incorporating label information to our feature learner, and expect the learned features of partial observations at the same progress level in the same category to be geometrically close to each other. We define the within-class within-progress-level variance here in order to regularize the learning of parameter matrix $\mathbf{W}$:

$$\begin{aligned}\Psi(\mathbf{W}) &= \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j=1}^{N} a_{ij}\|\mathbf{W}\mathbf{x}_i^{(k)} - \mathbf{W}\mathbf{x}_j^{(k)}\|_2^2 \\ &= \sum_{k=1}^{K}\mathrm{Tr}(\mathbf{W}\mathbf{X}^{(k)}\mathbf{L}\mathbf{X}^{(k)\mathrm{T}}\mathbf{W}^{\mathrm{T}}),\end{aligned} \tag{5}$$

where

$$\mathbf{X}^{(k)} = (\mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_N^{(k)}) \in \mathcal{R}^{d\times N}, \tag{6}$$

$\mathbf{L} \in \mathbb{R}^{N\times N}$ is the label Laplacian matrix: $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Here, $\mathbf{D}$ is the diagonal degree matrix with $\mathbf{D}_{ii} = \sum_{j=1}^{N} a_{ij}$, and $\mathbf{A}$ is the adjacency matrix that represents the label relationships between training videos. The (i, j)-th element a_{ij} in $\mathbf{A}$ is 1 if $y_i = y_j$ and $i \neq j$; and 0 otherwise.

Putting Eq. (1), Eq. (3), and Eq. (5) together, optimal parameter matrix $\mathbf{W}$ can be learned by

$$\begin{aligned}&\min_{\mathbf{W}} \|\bar{\mathbf{X}} - \mathbf{W}\tilde{\mathbf{X}}\|_F^2 + \alpha\|\bar{\mathbf{X}}^{(K)} - \mathbf{W}\mathbf{X}\|_F^2 + \beta\Psi(\mathbf{W}), \\ &\text{s.t. } \Delta^{(k+1)} \leqslant \Delta^{(k)}, k = 1, \cdots, K-1,\end{aligned} \tag{7}$$

where α and β are trade-off parameters balancing the importance of the corresponding terms, and $\Delta^{(k)}$ is the reconstruction error $\Delta^{(k)} = \|\mathbf{X}^{(K)} - \mathbf{W}\mathbf{X}^{(k)}\|_F^2$. As progress level k increases, the partial video feature $\mathbf{x}^{(k)}$ is geometrically approaching the corresponding full video $\mathbf{x}^{(K)}$. Consequently, the amount of information transferred from the full observation should be decreasing. Such prior knowledge is incorporated using the constraints in optimization problem (7). These constraints also implicitly capture temporal ordering information of inhomogeneous temporal units.

3.2. Model Learning

Eq. (7) is a convex problem, and can be solved by the augmented Lagrange method:

$$\begin{aligned}\mathcal{L} = &\|\bar{\mathbf{X}} - \mathbf{W}\tilde{\mathbf{X}}\|_F^2 + \alpha\|\bar{\mathbf{X}}^{(K)} - \mathbf{W}\mathbf{X}\|_F^2 + \beta\Psi(\mathbf{W}) \\ &+ \sum_{k=1}^{K-1} u_k(\Delta^{(k+1)} - \Delta^{(k)}) + \frac{v}{2}\sum_{k=1}^{K-1}(\Delta^{(k+1)} - \Delta^{(k)})^2 \\ &\text{s.t. } u_k \geq 0, \forall k = 1, \cdots, K-1,\end{aligned} \tag{8}$$

where u_k is the Lagrange multiplier corresponding to the k-th constraint defined in Eq. (7), and v is a penalty parameter. By introducing $\mathbf{\Delta} = [\Delta^{(2)} - \Delta^{(1)}, \cdots, \Delta^{(K)} - \Delta^{(K-1)}]^{\mathrm{T}}$ and $\mathbf{u} = [u_1, \cdots, u_{K-1}]^{\mathrm{T}}$, Eq. (8) is reduced to a compact form:

$$\begin{aligned}\mathcal{L} = &\|\bar{\mathbf{X}} - \mathbf{W}\tilde{\mathbf{X}}\|_F^2 + \alpha\|\bar{\mathbf{X}}^{(K)} - \mathbf{W}\mathbf{X}\|_F^2 \\ &+ \beta\Psi(\mathbf{W}) + \mathbf{u}^{\mathrm{T}}\mathbf{\Delta} + \frac{v}{2}\|\mathbf{\Delta}\|_2^2 \\ &\text{s.t. } u_k \geq 0, \forall k = 1, \cdots, K-1.\end{aligned} \tag{9}$$

Optimization problem in Eq. (9) can be solved by minimizing $\mathcal{L}$ with respect to $\mathbf{W}$ and maximizing the dual function of $\mathcal{L}$ with respect to $\mathbf{u}$ iteratively. Thus, $\mathbf{W}$ can be updated by directly setting $\frac{\partial\mathcal{L}}{\partial\mathbf{W}} = 0$. Please refer to the supplementary material for further details.

3.3. Deep Architecture

As suggested by [7, 28, 4], a deep architecture consisting of multiple layers of nonlinearity can improve the representation power of features, especially for classification tasks. Inspired by these studies, we also design a deep structure and nonlinearly map features of partial observations in a layer-wise fashion. This allows us to enhance the effectiveness of matrix $\mathbf{W}$ in reconstructing the features at various progress levels. Specifically, using a nonlinear squashing function $\sigma(\cdot)$ and the learned transformation matrix $\mathbf{W}$ at one layer, the new representations of all the partial videos $\bar{\mathbf{X}}$ can be computed by: $\mathbf{Z} = \sigma(\mathbf{W}\bar{\mathbf{X}})$, where $\sigma(\cdot)$ is defined as $\tanh(\cdot)$ in this work.

To learn a deep architecture, we stack our single-layer model in Section 3.1 as multi-layer deep networks, and learn the transformation matrices $\{\mathbf{W}_m\}_{m=1}^{M}$ of M layers using a greedy layer-wise scheme. Specifically, the output of the m-th layer, denoted by $\mathbf{Z}_m$, is used as the input for the $(m+1)$-th layer (Fig. 3(a)). The transformation matrix $\mathbf{W}_{m+1}$ of $(m+1)$-th layer is then trained after the m-th layer has been trained. For the first layer, the input $\mathbf{Z}_0$ is the raw features of all the partial observations $\bar{\mathbf{X}}$: $\mathbf{Z}_0 = \bar{\mathbf{X}}$.

3.4. Action Prediction

Given training videos, DeepSCN is first learned to generate the features of the videos. SVMs with intersection kernel (IKSVM) are then trained to be action predictors using all the feature layers $(\mathbf{Z}_0, \cdots, \mathbf{Z}_M)$ given by DeepSCN.

In order to train IKSVM action predictors appropriately, we consider the following two testing scenarios. **Scenario 1:** Progress level k of a testing video is unknown. This is practical, since in real-world applications, progress levels are always unknown in streaming videos. All the training partial videos are treated to be at the same progress level, and only one IKSVM model is trained. In testing, we use a single IKSVM to predict action labels (Figure 3(b)). **Scenario 2:** Progress level k of the testing video is known, which was used in [13, 2] but it is impractical in real-world applications. K support vector machines with the intersection kernel are trained for action prediction. The k-th SVM corresponds to partial observations at progress level k. In testing, progress level k of a testing video $\mathbf{x}$ is required in order to use the k-th SVM to make predictions (Figure 3(c)). Note that DeepSCN does not require the progress levels of testing videos to be known.

Testing. Given a testing video $\mathbf{x}$, DeepSCN is first used to generate the features of the video. Feature $\mathbf{z}$ of a testing video is built in a layer-wise fashion (see Figure 3(a)). Specifically, the feature $\mathbf{z}_1$ at the first layer is computed by $\mathbf{z}_1 = \sigma(\mathbf{W}_1\mathbf{z}_0)$ ($\mathbf{z}_0 = \mathbf{x}$), and then fed into the second layer with parameter $\mathbf{W}_2$. This procedure is repeated until all the features at M layers are computed. Then, the raw feature $\mathbf{x}$ and all these learned features are concatenated: $\mathbf{z} = (\mathbf{z}_0, \cdots, \mathbf{z}_M)$. Given $\mathbf{z}$, in Scenario 1, the one SVM is adopted to predict the label. If it is in Scenario 2, the k-th SVM from K SVMs will be used for prediction, where k is the progress level of the testing video $\mathbf{x}$.

4. Experiments

4.1. Dataset and Experiment Setup

We evaluate our approach on three datasets: UCF101 dataset [25], Sports-1M dataset [10], and BIT-Interaction dataset [12]. UCF101 dataset consists of $13,320$ videos distributed in 101 action categories. Sports-1M dataset contains $1,133,158$ videos divided into 487 classes. BIT dataset consists of 8 classes of human interactions, with 50 videos per class. It should be noted that N videos will be $10N$ videos to action prediction approaches due to the modeling of 10 progress levels. This larger volume of data increases the complexity of the prediction problem. Therefore, we use the first 50 classes in the Sports-1M datasets, and sample $9,223$ videos. This results in $92,230$ partial videos to prediction approaches.

Our approach works with both deep features and handcrafted features. We extract C3D features [27] from partial videos in UCF101 dataset and Sports-1M dataset as C3D model generates features for both segments and full videos. Pre-trained C3D model on Sports-1M dataset is used on UCF101 and Sports-1M datasets. Spatiotemporal interest points (STIPs) [5] and dense trajectory features (DTs) [30] are extracted from partial videos in BIT dataset. Bag-of-words model (with 500 visual words) is adopted to encode STIPs and DTs features.

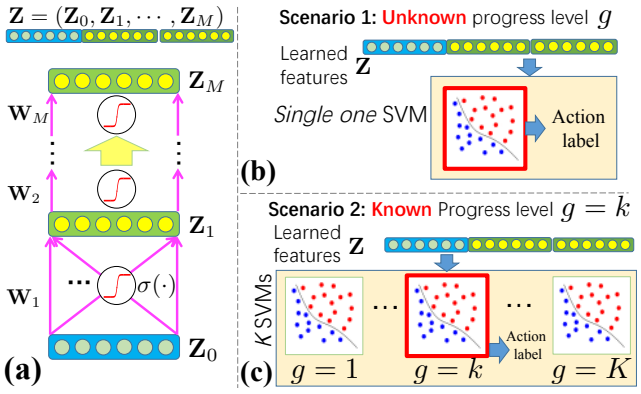


Figure 3. (a) Stacking features for a training/testing video in a layer-wise fashion. Action prediction in two testing scenarios: (b) progress level k is known, and (c) progress level k is unknown.

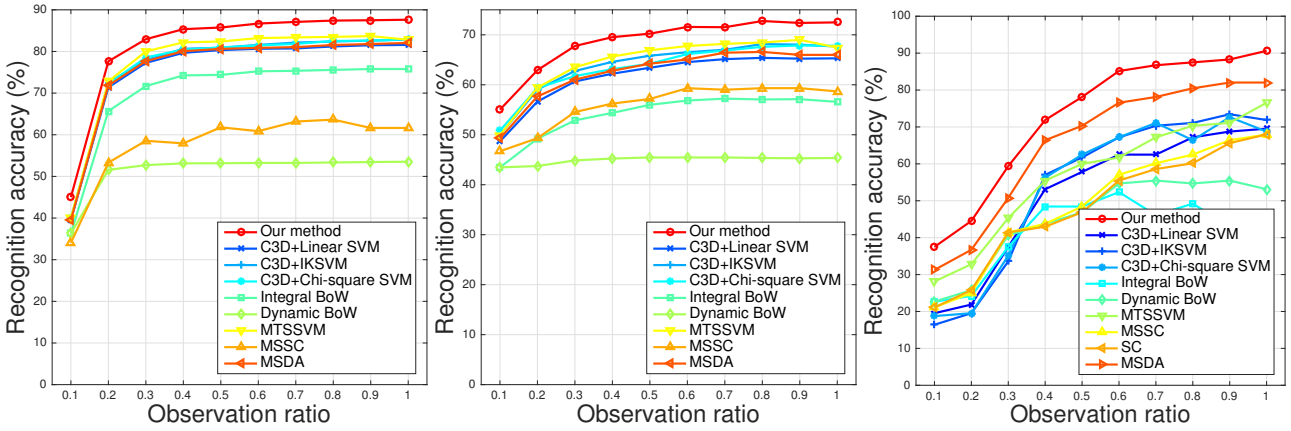
We follow the split scheme of [27] for UCF101 (split-1) and [10] for Sports-1M datasets, respectively. The first 15 groups of videos in UCF101 are used for training; the next 3 groups for cross-validation; and the remaining 7 groups for testing. We also follow the same experiment settings in [13] for BIT dataset, and use the first 34 videos in each class for training (in total 272 training videos) and use the remaining for testing. Default parameters settings are $M = 2, p = 0.5, \alpha = 0.1, \beta = 0.001$ for our method and $C = 1$ for IKSVMs on all the three datasets if not specified.

4.2. Prediction Performance

We compare with Dynamic BoW (DBoW) and Integral BoW (IBoW) [24], MSSC and SC [2][3], and MTSSVM [13]. SVMs with linear kernel, intersection kernel (IKSVM), chi-square kernel, and marginalized stacked autoencoder (MSDA) [4] are used as baselines. IBOW, DBOW, MTSSVM, and all baselines require the ground-truth progress levels to be known in testing. To perform fair comparison, the ground-truth progress levels of testing videos are known to all comparison methods, and all the comparison methods on one dataset are fed with the same features. Scenario 2 in Section 3.4 is adopted and K IKSVMs are trained for our method and MSDA. Note that our method also works without knowing the progress levels.

UCF101 dataset. Results in Figure 4(a) show that our method consistently outperforms all the comparison methods, especially at the beginning 5 progress levels. Our

[3]The code is available at http://www.visioncao.com/publications.html.



(a) UCF101 dataset (b) Sports1M dataset (c) BIT-Interaction dataset

Figure 4. Prediction results on (a) UCF101, (b) Sports-1M, and (c) BIT dataset. Note that these prediction approaches are optimized for partial videos and thus cannot be directly compared to action recognition approaches given full videos (observation ratio $r = 1.0$).

method achieves an impressive 45.02% prediction results when only 10% frames are observed, which is 4.97% higher than the MTSSVM. This demonstrates the effectiveness of learning information from full observations in Eq. (1). The performance of our method at observation ratio 0.4 is already higher than the best performance of all the other comparison methods, demonstrating the superiority of our method. It should be noted that DBoW achieves extremely low performance on this dataset. This is possibly because its action models computed by averaging features are not expressive enough to capture highly diverse action dynamics in the same category.

Sports-1M dataset. Results in Figure 4(b) demonstrate the superiority of our method over all the comparison methods. Our DeepSCN achieves an impressive 70.23% when only 50% frames are observed, higher than the best performance in 10 cases of all the other comparison methods. Note that DeepSCN makes accurate predictions at an early stage, demonstrating the effectiveness of its deep architecture. DeepSCN outperforms MSDA, showing the benefits of learning extra information from full videos and the monotonic error function. Our method consistently outperforms MTSSVM, MSSC, DBoW and IBoW, suggesting the benefit of learning sequential context information.

It should be noted that the performance of our method and C3D+SVM method given full video cannot be directly compared with the original C3D method [27]. We use all the frames in a video in this work while [27] randomly sampled 5 two-second clips from a video.

BIT dataset. Results in Figure 4(c) show that our method significantly outperforms all the other comparison methods, especially when 40% or more frames are observed. Our method achieves 71.88% at observation ratio $r = 0.4$, 5.47% higher than the runner-up MSDA method. At $r = 0.6$, our method achieves an impressive result of 85.16%, higher than the best performance of all the other comparison methods on 10 observation ratios. Our method remarkably outperforms MSSC and SC [2] in all the 10 cases, demonstrating its ability of learning more discriminative features for action prediction. The most noticeable improvement occurs at $r = 0.5$ where the performance increases over MSSC and SC are 29.69% and 31.25%, respectively. Our DeepSCN achieves noteably higher performance compared with DBoW and IBoW. We achieve 71.88% accuracy with only the first 40% frames of testing videos being observed, which is higher than DBoW and IBoW at all observation ratios.

4.3. Running Time

Our method is also compared with MTSSVM [13] and MSSC [2] in terms of running time. We executed their authors' MATLAB code on a 3.4GHz CPU with 64GB RAM, and report the total training and testing time[4] in Table 1. Results indicate that our method is significantly faster than the state-of-the-art MTSSVM and MSSC methods. On UCF101 dataset, our method (2 layers) spends 4 hour in training and testing, which is $35\times$ faster than MTSSVM and $105\times$ faster than MSSC. On Sports-1M dataset, our method (2 layers) is $20\times$ faster than MTSSVM, and $308\times$ faster than MSSC. The majority of our time spends in training the SVM action predictors. Training our DeepSCN method only costs 0.05 hours per layer on UCF101 dataset. Our method (3 layers) only takes 7 seconds (0.002 hours) in training and testing on BIT dataset, $60\times$ faster than the MTSSVM and $100\times$ faster than the MSSC method.

4.4. Instantly, Early, and Late Predictable Actions

It should be noted that actions differ in their prediction characteristics. Discriminative patterns of actions may ap-

[4]The training times of our method includes the training of DeepSCN and intersection kernel SVMs.

Table 1. Training and testing time (hours) of comparison methods on UCF101, Sports-1M, and BIT datasets. The number of layers in our method is set to 2 on UCF101 dataset, 2 on Sports-1M dataset, and 3 on BIT dataset, respectively.

Methods	UCF101	Sports-1M	BIT
MTSSVM [13]	140h	50h	0.12h
MSSC [2]	420h	770h	0.2h
Ours	**4h**	**2.5h**	**0.002h**

pear early or late in an action video. This affects the portion of a video that needs to be observed before being classified correctly, i.e., the *predictability* of an action.

We analyze the predictability of actions in UCF101 dataset, and study at what stage an action can be predicted. We define three categories of action videos according to their predictability: *instantly predictable* (IP), *early predictable* (EP), and *late predictable* (LP). An action video is IP means that the video can be predicted after only observing the beginning 10% portion of the video. EP means that an action video is not IP but can be predicted if the beginning 50% portion of the video is observed. LP means that an action video is neither IP nor EP, and can only be predicted if more than 50% portion of the video is observed.

Top 10 IP, EP, and LP actions in UCF101 dataset are listed in Figure 5. Results show that actions "Billiards" and "IceDancing" are the easiest to predict; all of their testing samples are instantly predictable. In our experiment, there are 33 action categories having over 50% of their respective testing videos instantly predictable (correctly classified after only observing the beginning 10% frames). Figure 5 also shows that 4 actions have all their testing samples early predictable. In fact, there are 38 actions out of 101 actions having over 50% of their respective testing videos that are early predictable (less than 50% video frames need to be observed). The action "JavelinThrow" can be considered as the most challenging class to predict as 29% of its testing samples are late predictable (more than 50% video frames need to be observed), higher than all the other actions. In all the $37,830$ testing partial videos, 35.45% of them are instantly predictable, and 43.78% are early predictable; only 2.09% are late predictable. The remaining 18.69% partial videos cannot be correctly predicted. This suggests that a majority of action videos can be correctly classified using our approach after observing the beginning 50% frames of the videos. On Sports-1M dataset, "equestrianism" is the easiest action to predict and "artistic gymnastics" is the most challenging action to predict. On BIT dataset, "bow" and "pat" are the easiest and the most challenging actions to predict, respectively. Please refer to the supplemental material for the results on Sports-1M and BIT datasets.

4.5. Unknown vs Known Progress Level

Existing methods in [13, 24, 2] assume that the progress levels of videos are known in testing (Scenario 2 in Section 3.4), which is impractical. In this experiment, we evaluate our method in a practical scenario (Scenario 1 defined in Section 3.4), i.e. progress levels are *unknown* in testing. All the partial videos are treated to be at the same progress level. DeepSCN and only a single SVM model are trained (the ONE method). Its performance is compared with two other methods that require progress levels to be given in testing: the RAND method (progress levels are randomly generated) and the TRUE method (ground truth progress levels are used in the testing phase). Both of the two methods train DeepSCN and K SVMs

Instantly Predictable	Early Predictable	Late Predictable
Billiards	Fencing	JavelinThrow
IceDancing	FrisbeeCatch	HighJump
RockClimbingIndoor	SoccerPenalty	FrontCrawl
PlayingPiano	VolleyballSpiking	HeadMassage
PommelHorse	HulaHoop	Haircut
Rowing	FieldHockeyPenalty	PlayingViolin
Skijet	BasketballDunk	HandstandWalking
JugglingBalls	CliffDiving	PoleVault
SoccerJuggling	Bowling	CricketBowling
TaiChi	TennisSwing	ThrowDiscus

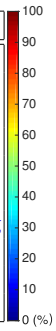


Figure 5. Top 10 instantly, early, and late predictable actions in UCF101 dataset. Action names are colored and sorted according to the percentage of their testing samples falling in the category of IP, EP, or LP. This figure is best viewed in color.

Performance variations of the three methods on UCF101, Sports-1M, and BIT datasets are shown in Table 2. Results show that the average performance variation between the TRUE method and the ONE method is within 1% on UCF101 and Sports-1M datasets, and it is within 3.12% on BIT dataset. This demonstrates that ONE method can be used in practical scenarios without significant performance decrease where the progress levels are unknown. Thanks to the proposed DeepSCN, partial videos at various progress levels can be accurately represented, thereby making one SVM powerful enough for predicting these partial videos and making the progress levels unnecessary in testing. In addition, training ONE method is significantly faster than training TRUE method as ONE method only trains one SVM while TRUE method needs to train K SVMs. The average performance gap between RAND method and TRUE method is 2.25% and 0.34% on UCF101 and Sports-1M datasets, respectively, indicating the robustness of our approach to progress levels on the two datasets. The gap increases to 16.02% on BIT dataset as short video clips (most of videos are less than 100 frames) and non-cyclic actions (such as "push" and "handshake") are present in the dataset. Using inaccurate progress levels in testing would confuse action predictors, and thus decreases the performance.

4.6. Effectiveness of Components and Parameters

We evaluate the effectiveness of model components in our method, and the sensitivity to the number of layers M

Table 2. Prediction results (%) on UCF101, Sports-1M and BIT datasets using RAND, TRUE, and ONE methods. Observation ratios $r \in \{0.1, 0.3, 0.5, 0.7, 1.0\}$. The average performance is computed over all 10 observation ratios.

		UCF101						Sports-1M						BIT					
	#SVMs	avg.	0.1	0.3	0.5	0.7	1.0	avg.	0.1	0.3	0.5	0.7	1.0	avg.	0.1	0.3	0.5	0.7	1.0
RAND	K	79.06	43.91	81.37	83.51	83.72	85.23	68.28	54.33	66.58	70.50	72.06	70.71	56.95	13.28	32.81	69.63	78.13	73.44
TRUE	K	81.31	45.02	82.95	85.75	87.10	87.63	68.62	55.02	67.76	70.23	71.52	72.49	72.97	37.50	59.38	78.13	86.72	90.63
ONE	1	80.55	44.31	82.77	85.46	86.34	86.65	68.62	56.15	67.22	70.50	71.57	71.84	69.85	33.63	50.00	81.25	85.94	85.94

on UCF101 dataset. The sensitivity results of our method to corruption probability p, and parameters α and β are shown in the supplemental material.

Components. We compare with several variants of our full method, including the method without self reconstruction in Eq. (3) (no-SR method), the one with $\alpha = 0$ in Eq. (7), the one with $\beta = 0$, and the one without the constraints in the optimization problem (7) (no-CS method). The averaged prediction results over 10 observation ratios and prediction results over observation ratios $0.1, 0.3, 0.5, 0.7, 1.0$ are summarized in Table 3. Our method significantly outperforms the no-SR method by 5.7% on average, demonstrating the effectiveness of learning robust features from the partial observation itself. The performance gap between our method and the ($\alpha = 0$) method shows the importance of learning information from full observations. The variant method ($\beta = 0$) loses label information and fails to successfully separate the learned features in different categories, thereby achieving lower performance. The strength of the constraints in Eq. (7) can be seen from the performance variance between our method and the no-CS method. The constraints implicitly capture temporal orderings over inhomogeneous units, which are beneficial for predicting complex actions.

Table 3. Comparison experiments among variants on partial videos of observation ratios $r \in \{0.1, 0.3, 0.5, 0.7, 1.0\}$. The average performance is computed over all 10 observation ratios.

Methods	avg.	0.1	0.3	0.5	0.7	1.0
no-SR	75.61	39.49	76.98	80.65	81.24	82.02
$\alpha = 0$	76.55	40.29	78.17	81.02	82.34	82.87
$\beta = 0$	76.45	40.10	78.14	81.21	82.18	82.87
no-CS	76.50	39.92	78.14	81.15	82.32	82.87
Ours	**81.31**	**45.02**	**82.95**	**85.75**	**87.10**	**87.63**

Number of layers M. We also study performance variations given various layers of features. Prediction accuracies of our networks with $M \in \{0, 1, 2, 3\}$ layers are shown in Table 4 ($M = 0$ means the proposed DeepSCN is not used). Results indicate that the average performance over 10 observation ratios generally improves when more layers are stacked. With only one hidden layer features ($M = 1$), our method outperforms the method using raw features only ($M = 0$) by 3.57% on average performance. The performance difference increases to 7.07% when $M = 3$ layers of features are stacked. Stacking multiple layers of features has slightly higher positive effect on the ending portions of videos than the beginning portions of videos. At beginning portions of videos (observation ratios $r = 0.1$ and $r = 0.3$), the network with $M = 3$ layers of features outperforms the one with $M = 1$ layers of features by 3.02% and 3.09%, respectively. The performance gap increases to 3.6% and 3.43% at ending portions of videos ($r = 0.7$ and $r = 1.0$), respectively. This is possibly because the raw features are more discriminative in the ending portions of videos than the beginning portions. Stacking more layers in DeepSCN benefits more from the discriminative features, and thus achieves better performance in the ending portions.

Table 4. Accuracy (%) of our method on partial videos of observation ratios $r \in \{0.1, 0.3, 0.5, 0.7, 1.0\}$ with layers $M \in \{0, 1, 2, 3\}$. $M = 0$ means the proposed DeepSCN is not used.

#Layer	avg.	0.1	0.3	0.5	0.7	1.0
$M = 0$	75.39	39.55	77.27	80.33	80.73	81.58
$M = 1$	78.96	42.98	80.97	83.53	84.43	85.28
$M = 2$	81.31	45.02	82.95	85.75	87.10	87.63
$M = 3$	83.36	46.00	84.06	87.21	88.08	88.71

5. Conclusion

This work addresses the problem of predicting the action label of a video before the action execution ends. We have proposed an efficient and powerful approach for recognizing unfinished human actions from videos. Our approach learns extra information from fully observed actions to improve the discriminative power of the features from temporally partial observations. We further improve the representation power of the features by temporally ordering the amount of the information, incorporating label information, and stacking multiple layers of features. Our method is evaluated on UCF101, Sports-1M, and BIT-Interaction datasets, and shows significant improvements with up to $300\times$ faster speed over state-of-the-art methods. An interesting finding shows that actions differ in their *predicability*. This inspires us to further explore the temporal structures of actions for prompt and accurate prediction in future work.

Acknowledgement

This work is supported in part by the NSF IIS award 1651902, ONR Young Investigator Award N00014-14-1-0484, and U.S. Army Research Office Young Investigator Award W911NF-14-1-0218.

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*, 2016.

[2] Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. Siskind, and S. Wang. Recognizing human activities from partially observed videos. In *CVPR*, 2013.

[3] M. Chen, K. Weinberger, F. Sha, and Y. Bengio. Marginalized denoising auto-encoders for nonlinear representations. In *ICML*, 2014.

[4] M. Chen, Z. E. Xu, K. Q. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In *ICML*, 2012.

[5] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, 2005.

[6] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, June 2015.

[7] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[8] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. Technical report, arXiv:1207.0580v1, 2012.

[9] M. Hoai and F. D. la Torre. Max-margin early event detectors. In *CVPR*, 2012.

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

[11] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *ECCV*, 2012.

[12] Y. Kong, Y. Jia, and Y. Fu. Interactive phrases: Semantic descriptions for human interaction recognition. In *TPAMI*, volume 36, pages 1775–1788, 2014.

[13] Y. Kong, D. Kit, and Y. Fu. A discriminative model with multiple temporal scales for action prediction. In *ECCV*, 2014.

[14] T. Lan, T.-C. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *ECCV*, 2014.

[15] I. Laptev. On space-time interest points. *IJCV*, 64(2):107–123, 2005.

[16] B. Letham, C. Rudin, and D. Madigan. Sequential event prediction. *Machine Learning*, 93:357–380, 2013.

[17] K. Li and Y. Fu. Prediction of human activity by discovering temporal sequence patterns. *TPAMI*, 36(8):1644 – 1657, 2014.

[18] S. Ma, L. Sigal, and S. Sclaroff. Space-time tree ensemble for action recognition. In *CVPR*, June 2015.

[19] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *CVPR*, 2016.

[20] B. Ni, P. Moulin, X. Yang, and S. Yan. Motion part regularization: Improving action recognition via trajectory selection. In *CVPR*, June 2015.

[21] M. Pei, Y. Jia, and S.-C. Zhu. Parsing video events with goal inference and intent prediction. In *ICCV*, 2011.

[22] M. Raptis and L. Sigal. Poselet key-framing: A model for human activity recognition. In *CVPR*, 2013.

[23] C. Rudin, B. Letham, A. Salleb-Aouissi, E. Kogan, and D. Madigan. Sequential event prediction with association rules. In *COLT*, pages 615–634, 2011.

[24] M. S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *ICCV*, 2011.

[25] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild. Technical report, CRCV-TR-12-01, 2012.

[26] Y. Tian, R. Sukthankar, and M. Shah. Spatiotemporal deformable part models for action detection. In *CVPR*, 2013.

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

[28] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11:3371–3408, 2010.

[29] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating visual representations from unlabeled video. In *CVPR*, 2016.

[30] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *CVPR*, pages 3169–3176, 2011.

[31] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1):60–79, 2013.

[32] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, pages 4305–4314, 2015.

[33] B. Wu, C. Yuan, and W. Hu. Human action recognition based on context-dependent graph kernels. In *CVPR*, 2014.

[34] Y. Yang and M. Shah. Complex events detection using data-driven concepts. In *ECCV*, 2012.

[35] Y. Zhou and T. L. Berg. Temporal perception and prediction in ego-centric video. In *ICCV*, 2015.

[36] Y. Zhou, B. Ni, R. Hong, M. Wang, and Q. Tian. Interaction part mining: A mid-level approach for fine-grained action recognition. In *CVPR*, June 2015.