The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Action Prediction from Videos via Memorizing Hard-to-Predict Samples

Yu Kong,[†*] Shangqian Gao,[‡*] Bin Sun,[†] Yun Fu[†§]
[†]Department of Electrical & Computer Engineering, Northeastern University, Boston, MA, USA
[‡]College of Engineering, Northeastern University, Boston, MA, USA
[§]College of Computer & Information Science, Northeastern University, Boston, MA, USA
{yukong,yunfu}@ece.neu.edu, {gao.sh,sun.bi}@husky.neu.edu

Abstract

Action prediction based on video is an important problem in computer vision field with many applications, such as preventing accidents and criminal activities. It's challenging to predict actions at the early stage because of the large variations between early observed videos and complete ones. Besides, intra-class variations cause confusions to the predictors as well. In this paper, we propose a mem-LSTM model to predict actions in the early stage, in which a memory module is introduced to record several "hard-to-predict" samples and a variety of early observations. Our method uses Convolution Neural Network (CNN) and Long Short-Term Memory (LSTM) to model partial observed video input. We augment LSTM with a memory module to remember challenging video instances. With the memory module, our mem-LSTM model not only achieves impressive performance in the early stage but also makes predictions without the prior knowledge of observation ratio. Information in future frames is also utilized using a bi-directional layer of LSTM. Experiments on UCF-101 and Sports-1M datasets show that our method outperforms state-of-the-art methods.

Action prediction is receiving increasing interests in recent years due to its broad and important applications in real-world scenarios, such as visual surveillance and traffic accident avoidance. Different from action recognition, in action prediction, the action label needs to be inferred before the entire action execution has been observed. More importantly, it is crucial that a prediction algorithm can make accurate predictions at the very beginning stage of a video, for example, when only few frames of a video are observed.

Although action recognition approaches (Karpathy et al. 2014; Tran et al. 2015) have made great success, action prediction is still a relatively new research topic and it still has several problems that need to be addressed. In recent years, various attempts have been made in action prediction (Ryoo 2011a; Cao et al. 2013b; Lan, Chen, and Savarese 2014a; Kong and Fu 2016; Kong, Tao, and Fu 2017), but *misprediction* remains. One major reason is that, in some actions, features from the beginning few frames are not discriminative enough to be classified due to visual similarity. Therefore, a classifier learned using these features may not be able to find the correct classification boundaries, and thus prediction approaches would fail. As shown in Figure 1, actions such as "basketball" and "basketball dunk" have the extremely similar appearance at their early stage. It is essential to discover discriminative information for prediction at an early stage. Recent work in (Kong, Tao, and Fu 2017) show that prediction performance usually becomes stable when only half length of videos are observed. This indicates that rich discriminative information is often seen in the middle of a video, thereby limiting the prediction performance at the early stage of videos.

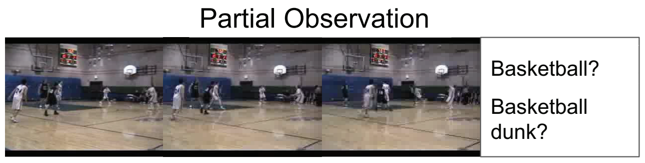


Figure 1: Actions may have very similar appearance in the beginning few frames, for example, "basketball" and "basketball dunk" in UCF-101 dataset, thereby fail action prediction approaches.

Existing work (Kong, Tao, and Fu 2017) enhances the discriminative power of features by transferring future information from full videos to partial videos. MTSSVM in (Kong and Fu 2016) characterizes action evolution and creates a nonlinear prediction model using kernels. MSSC method proposed in (Cao et al. 2013a) learns a new feature representation by enforcing sparse constraints on the features. Still, their prediction performance at the early stage of videos (for example, only $10\% - 20\%$ of the frames are observed) is still relatively low even though the discriminative power of the features from these frames is already enhanced. In addition, these approaches are impractical as they require to know the observation ratio of a testing video.

We introduce mem-LSTM to solve aforementioned problems. We propose to use memory to store hard-to-predict training samples in order to improve the prediction performance at early stage. The memory module used in this work measures the predictability of each training sample, and will store those challenging ones. Given a testing partial video (a query sample), the memory computes the similarity be-

[*]Equal contributions.

tween the query sample and all the memorized samples, and returns the label of the most similar one for the query. As the memory retains a large pool of samples, it allows us to create complex classification boundaries, which are particularly useful for discriminating partial videos at the beginning stage. Our method also utilizes future information in a video for accurate prediction. Using a bi-directional layer of LSTM, information existing in the middle and the late segments of a video is propagated to the features extracted from the beginning frames, thereby improving the discriminative power of the features, and enhancing prediction performance. We use a two-stream framework in this paper, in which RGB and flow streams are fused. In each stream, a convolutional neural network is utilized to extract features from frames and then fed into an LSTM at each time interval. A two layer bi-directional LSTM with both forward connections and backward connections is employed to characterize temporal action evolution and capture future information for prediction. The output of the two streams are then concatenated into a vector, and serve as an input to the global memory to compare with all hard-to-predict samples.

The main contribution of this paper is two-fold. We utilize a life-long memory module to remember hard-to-predict observations. This allows us to learn a more complex classification boundary, which is particularly useful for classifying partial videos with insufficient discriminative information. Furthermore, we utilize information in future frames to further enhance the prediction performance through a bi-directional LSTM layer. This essentially transfers discriminative information from middle and late segments of videos to the beginning segments. Compared with existing methods (Kong, Kit, and Fu 2014; Cao et al. 2013a; Ryoo 2011b), our method does not need to know the observation ratios of testing videos and thus is more practical.

Definitions. We follow the problem setup described in (Kong, Tao, and Fu 2017). To mimic sequential data arrival, we segment a complete video $\mathbf{x}$ with T frames into $K = 10$ segments. Consequently, each segment contains $\frac{T}{K}$ frames. Video lengths T may vary for different videos, thereby causing different lengths in their segments. For a video of length T, its k-th segment ($k \in \{1, \cdots, K\}$) contains frames starting 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 defined as a temporal subsequence that consists of the beginning k segments of the video. The *progress level* g of the partial video $\mathbf{x}^{(k)}$ is defined by the number of the segments contained in the partial video $\mathbf{x}^{(k)}$: $g = k$. The *observation ratio* r of a partial video $\mathbf{x}^{(k)}$ is $\frac{k}{K}$: $r = \frac{k}{K}$.

Related Works

Action recognition is an important research topic in computer vision, which relies on the extracted features from temporally complete action videos. These features, such as space-time interest points (Laptev 2005) and dense trajectory (Wang et al. 2011) are composed of spatiotemporal features and local appearance features. In (Wang and Schmid 2013) the dense trajectory was improved by utilizing camera motion estimation, detection-based noise canceling and being represented by a Fisher vector.

Recent studies demonstrated that action features can be learned by deep learning methods such as convolutional neural networks (CNN) and recurrent neural networks (RNN). Two-stream networks (Simonyan and Zisserman 2014) built on RGB frames and optical flow frames have shown their promising results on various action datasets. In (Ranzato et al. 2014; Srivastava, Mansimov, and Salakhudinov 2015; Donahue et al. 2014; Song et al. 2017; Wu et al. 2015), RNNs have been used to model long-term temporal correlations in videos, and generate video representation for action classification. However, most of the methods are expected to recognize actions through temporally complete videos. Their performance on temporally incomplete action videos is unknown.

Action prediction is another important task to predict the action label contained in a partially observed video. Ryoo *et al.* (Ryoo 2011b) proposed the integral bag-of-words and dynamic bag-of-words approach for action prediction. Cao *et al.* (Cao et al. 2013a) proposed a probabilistic formulation for human activity recognition from partially observed videos. Kong *et al.* (Kong, Kit, and Fu 2014) presented a novel multiple temporal scale model in the support vector machine framework for predicting unfinished actions. From a perspective of interfering social interaction, Lan *et al.* (Lan, Chen, and Savarese 2014b) developed "hierarchical movements" for action prediction, which is able to capture the typical structure of human movements before an action is executed.

Deep learning methods have also shown in action prediction. Ranzato *et al.* (Ranzato et al. 2014) introduced a generative model using the RNN to predict motion in the video. Srivastava *et al.* (Srivastava, Mansimov, and Salakhudinov 2015) proposed an unsupervised learning approach by using LSTM to predict future video sequence.These works mainly focus on how to describe a segment of an action, as a result, they can achieve the propose of action prediction by utilizing these well-defined segment features. Our method does not aim to describe how an action is evolved in the different segment; instead, we focus on remembering the early stage of the action. Moreover, most of action prediction methods need to know the observation ratio of a test video to make a prediction. Our methods do not need to know the observation ratio, and thus can be applied to streaming videos.

Revisiting Long Short Term Memory

Long Short Term Memory (LSTM) (Gers, Schmidhuber, and Cummins 2000) has achieved great success in various sequence learning tasks (Fernandez, Graves, and Schmidhuber 2017). A typical LSTM has three gates, which include an input gate i_t, a forget gate f_t, and an output gate o_t. The three gates are essentially nonlinear summation units. The gates are used to compute activations from inside and outside of the LSTM block, and manage the activation of the cell via

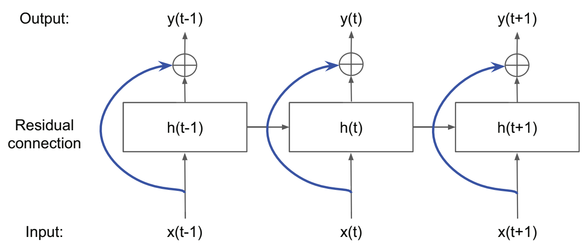


Figure 2: Illustration of the residual connections (in blue color) in LSTMs.

multiplications.

$$\begin{cases} a_t = tanh(W_c x_t + U_c h_{t-1} + b_a) \\ i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ c_t = i_t a_t + f_t c_{t-1} \\ h_t = o_t tanh(c_t), \end{cases} \tag{1}$$

where c_t is the state of a cell at time step t, h_t is the LSTM output, and $\sigma(\cdot)$ is an activation function.

Limitations in LSTM

When training multi-layer LSTM, we found that it is hard to converge and it usually hurts the performance. To solve this problem, we use two strategies to deal with it and create the basic LSTM building block used in this research. First, we combine LSTM with a residual connection (He et al. 2015) to facilitate learning process. A residual connection is added to LSTM so that the input itself is added to the output. Adding the residual connection helps alleviate overfitting and improve prediction accuracy. More formally, adding a residual connection to LSTM can be given by (see Figure 2):

$$h_t = o_t tanh(c_t) + x_t \tag{2}$$

We also regularize the hidden layer activation by adding an additional term to the cost function: $\frac{1}{T}\beta \sum_0^T(\|h_t\| - \|h_{t-1}\|)^2$ (Krueger and Memisevic 2016). This regularization minimizes the difference between the output of LSTM at time t, h_t and the output of LSTM at time step $t-1$, h_{t-1}. It is added to each layer of LSTM, and helps reduce overfitting and performs better than dropout scheme. The performance of our LSTM model can be increased by a considerable margin combining with this regularization, and it is still helpful when we have more LSTM layers. It is quite surprising that this regularization has such effect on our model.

However, in our experiment, a residual connection works poor on flow stream. The performance of LSTM with regularization is inferior to the original LSTM model. This may because of the nature of the flow data. Unlike RGB images, flow images have a lot of variance between two continuous frames, which makes regularization less useful.

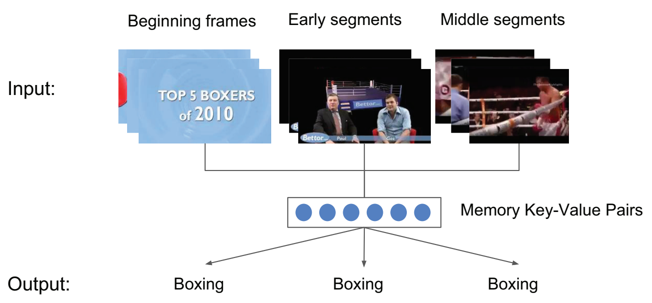


Figure 3: A memory module remembers partial videos and full videos of the same action category in a local neighborhood, thereby allowing a testing video at various stages to find a similar one withe same action label.

Our Method

Overall Architecture

The overall architecture is shown in Figure 4. Our method can be considered as a two-stream network, consisting of a RGB stream and a flow stream. In both of the RGB stream and flow stream, an 18-layer residual network (He et al. 2015) is used to extract features from each frame. Various LSTM layers are applied on top of the features to model temporal correlations of the input frames in a small temporal window. The architecture of the flow stream is similar to the RGB stream, but we only use LSTM without residual connections in the flow stream. The output of the two streams is concatenated and then fed into the proposed memory module at every time interval. The memory module then computes the distance between the query sample and all the memorized samples, and assigns the action label of the closest memorized sample to the query sample. The benefit of using the memory module is that it can remember early observations, thereby improving the generalization power.

The proposed network is particularly developed for action prediction. Compared with existing LSTM networks, (Srivastava, Mansimov, and Salakhudinov 2015; Donahue et al. 2014; Wu et al. 2015; Yang, Molchanov, and Kautz 2016), our method augments LSTMs with a *life-long memory* for the purpose of memorizing hard-to-predict samples. This is particularly important when predictions are made when only few frames are observed. We also enhance the representation power of partial videos by utilizing future information through a bi-directional LSTM layer. Compared with existing action prediction approaches (Kong, Tao, and Fu 2017; Kong, Kit, and Fu 2014) in which the observation ratio should be known to those action predictors, our method does not need such information and thus is more practical in real-world scenarios.

Memorizing Hard-to-Predict Samples

It is essential to predict actions at their early stage, for example, when only the beginning 10% of the frames are observed. Although recent work (Kong, Tao, and Fu 2017; Kong and Fu 2016) has improved the feature representation power, the prediction performance at an early stage is

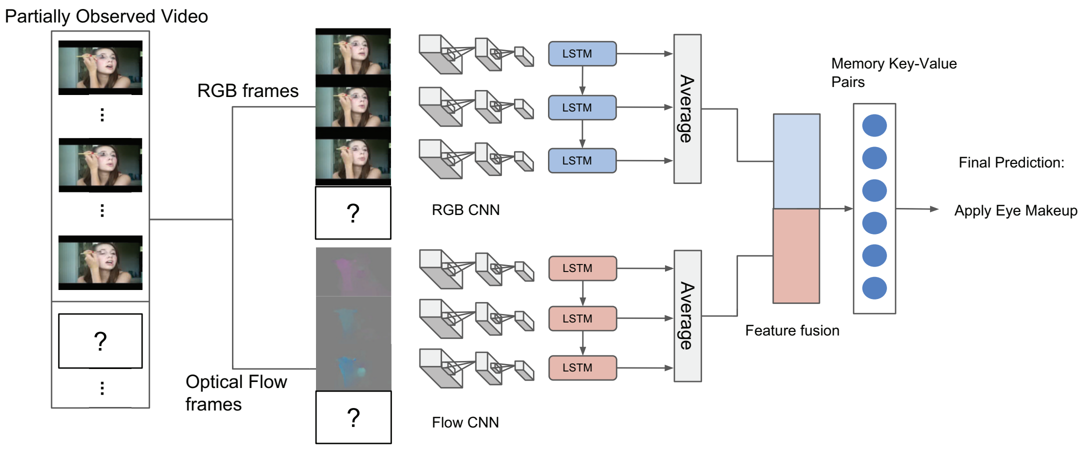


Figure 4: Architecture of the proposed network. A partially observed video is split into two streams, a RGB stream and a flow stream. CNN networks are used on each RGB frame and flow frame. LSTM networks are used to generate latent features for each frame, which are then averaged to generate a representation for each stream. The output representations of the two streams are then concatenated to query the memory Key-Value pairs to produce the final prediction.

still low since the learned features extracted from observed partial videos are hardly distinguishable. Instead of improving the discriminative power of the features, we propose to memorize hard-to-predict training samples in the training phase. As shown in Figure 4, the memory (Kaiser et al. 2017) is added to the last layer of the proposed network, acting as a global memory. The memory is optimized to keep partial videos and full videos of the same class in a local neighborhood, thereby allowing a testing partial video at either early stage, middle stage, or late stage to find a similar one with same action label (Figure 3).

The memory module M consists of a matrix $K \in \mathcal{R}^{m\times k}$ of memory keys, a matrix $V = \{\mathbf{v}\}$ storing m memory values, and an additional vector A of size m that tracks the age of items stored in memory. A memory M is defined as follows:

$$M = (K, V, A). \tag{3}$$

Here, K stores m hard-to-predict samples of dimensional k, and $V = \{\mathbf{v}\} = \{(y, z)\}$ indicates their corresponding action classes y and progress levels z.

Memory query. Basically, the memory is a set of key-value pairs. Memory requires a query as input to look for the most similar instance stored in key matrix K and outputs the corresponding value as result. The question is how to compute the similarity d. In our experiment, we use two similarity metrics, dot product and Gaussian kernel, to compute the similarity, which are defined by

$$d = q \cdot K[i], \text{and } d = \exp(-\frac{\|q - K[i]\|^2}{2\delta^2}). \tag{4}$$

Here q represents the input query, and $K[i]$ represents the i-th key in the memory.

Memory loss is computed based on correct neighbors and incorrect neighbors. Suppose that the corrected value $\mathbf{v} = (y, z)$ is given to the query sample q. Given a series of k nearest neighbors $(n_1, \cdots, n_k)$ to q, the correct neighbor n_u is defined as $V[n_a] = \mathbf{v}$, where a is the smallest index. Incorrect neighbors are the samples that either have incorrect action labels y: $V[n_b](y) \neq \mathbf{v}(y)$, or have incorrect progress levels z: $V[n_c](z) \neq \mathbf{v}(z)$, where b and c are the smallest indices. The memory loss can be expressed as

$$\max([q \cdot K[n_b] + q \cdot K[n_c] - q \cdot K[n_a] + \xi], 0), \tag{5}$$

where ξ is a positive margin. This loss function essentially uses dot product to compute the similarity between the query and the memorized samples. We would like to maximize the similarity to the correct key and also minimize the similarity to the incorrect keys.

Memory update is performed to account for the fact that newly presented query q corresponds to $\mathbf{v}$. The update is performed depending on the returned key value is correct or not. Denote n is the nearest neighbor to q: $n = NN(q, M)$.

If the returned value $\mathbf{v}$ is incorrect (either action label or progress level), i.e., $V[n] \neq \mathbf{v}$, we will update the memory by writing the query $(q, \mathbf{v})$ into the memory. We find the place in the memory for writing the query by $n' = \arg\max_i A[i] + r$, where r is a random number introducing randomness. Then the query q is written into the memory by

$$K[n'] = q, V[n'] = \mathbf{v}, A[n'] = z. \tag{6}$$

Here, the age of a memory item starts at its progress level z as we encourage partial videos at their early stage to be stored in the memory. The age of all items will then be incremented by 1 after every memory update.

If the returned value $\mathbf{v}$ is correct (both action label and progress level are correct), then the key is updated by averaging the item $K[n]$ and q, and normalizing it:

$$K[n] \leftarrow \frac{K[n] + q}{\|K[n] + q\|}. \tag{7}$$

The age of item n is also reset by $A[n] \leftarrow (A[n] + z)/2$, where z is the progress level of the query q.

Integrating Memory into LSTM

Previous work (Kaiser et al. 2017) adds memory module after every time-step of LSTMs. However, we argued that there is no need to remember every time-step output of the LSTMs in the context of action prediction, because of the motion between neighbor frames would be quite similar in the video. Instead, we only use one global memory to remember all the early observations in this work (see Figure 4). We average the outputs of the LSTMs from all time-steps, and feed this output into the memory module. By doing this, the memory module can remember the global output of the LSTM.

The function of global memory is similar to a classifier. However, the major difference is that action label given by the memory is computed by the data similarity while the label given by a classifier is computed by the learned model. Features extracted from partial videos (especially the ones at their early stage) have less discriminative power compared to the features from full videos (Kong, Tao, and Fu 2017; Kong, Kit, and Fu 2014). Using such indistinguishable features may not be able to learn a complex classification boundary. On the contrary, computing sample similarity in a large sample pool using memory can learn a complex classification boundary, and thus can improve the prediction performance even few frames are observed. We will show that in the experiments.

Attention could be an alternative for modeling the correlations between frames within a small period of time and the action categories. However, attention focuses on local short-term information, which may not be able to provide information in a long-term scope. To get better performance of early detection, a life-long memory mechanism is introduced in our framework. By utilizing this mechanism, the model can learn hard-to-predict actions at their early stages.

Modeling Future Context

One limitation of conventional LSTM is that they are unable to make use of future context. We propose to use bidirectional LSTM and enrich current features by integrating future information. As shown in Figure 5, in this work, we add a backward LSTM on top of the forward LSTM without interactions in the hidden layers. The two layers compute their own hidden features *independently*, and their outputs are then summed up at output layer at each time step from $t = 1$ to $t = T$.

The proposed bi-directional LSTM has the capability of utilizing both previous and future context information for action prediction. Different from (Kong, Tao, and Fu 2017), our method is flexible in utilizing bi-directional information, while (Kong, Tao, and Fu 2017) only transfers information from future frames to the current. In addition, our method filters out unnecessary information using the cell in LSTM, while (Kong, Tao, and Fu 2017) does not and thus may introduce noise in the learned features.

Two-Stream Network

We follow the method in LRCN (Donahue et al. 2014) to create a two stream network that contains a RGB stream and a flow stream. ResNet-18 model is trained on RGB images for the RGB stream. A flow frame is calculated by (Brox et al. 2004), and the results are transformed into a ”flow image” by scaling and shifting x and y flow values to a range of $[0, 255]$, and the center of x and y flow is set to 128. After we obtain x and y channels, we compute the third channel by using the magnitude of flow. A ResNet-18 model is then trained on those flow images. We use this ResNet-18 model to extract flow features for training the flow stream. As aforementioned, residual connection and regularization does not improve the result of the flow stream, and thus the bi-directional LSTM is used for training on flow images.

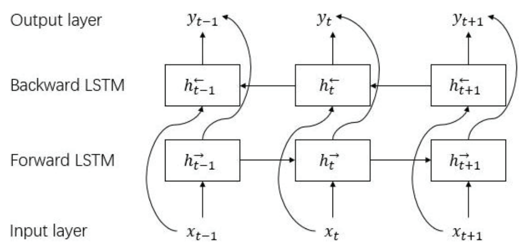


Figure 5: Structure of bi-directional LSTM. The forward LSTM and backward LSTM are separated. The results of two layers are then added to the output layer.

To fuse the results from both RGB and flow streams, we concatenate the output of RGB stream and flow stream, and put this result into a memory module. We also tried several fusion strategies including average pooling, sum pooling and max pooling on the two features, but they result in poor performance. It is possible because the features from partial videos are considerably noisy. If we allow interaction between the features from two streams, noise level will be further enhanced, and thus may confuse the memory and cause incorrect key-value pairs in the memory module. Concatenation, on the contrary, does not allow such interactions, and keeps the original properties of the features, thereby leading to a better prediction performance.

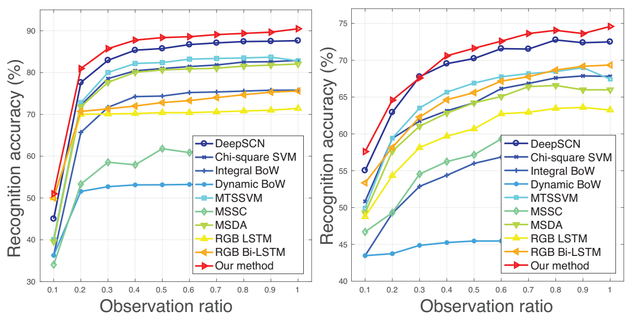
Experiment

Datasets

We use the UCF-101 (Soomro, Zamir, and Shah 2012) and Sports-1M datasets (Karpathy et al. 2014) to evaluate our method. UCF1-101 dataset consists of 13,320 videos distributed in 101 human actions, such as “Baseball pitch” and “Playing Guitar”. Sports-1M dataset consists of $1,133,158$ videos divided into 487 videos. In this work, we follow (Kong, Tao, and Fu 2017), and only test on part of the Sports-1M dataset in order to conduct a fair comparison. We use the first 50 classes in the Sports-1M dataset and sample 9223 videos following (Kong, Tao, and Fu 2017).

Implementation Details

We adopted the ResNet-18 (He et al. 2015) and VGG-19 for RGB and optical flow data separately. The ResNet-18 is pre-trained on ImageNet ILSVRC-2012 (Deng et al. 2009)



(a) UCF-101 dataset (b) Sports-1M dataset

Figure 6: Action prediction results on (a) UCF-101 dataset and (b) Sports-1M dataset.

training set. Then the ResNet-18 is fine-tuned on UCF-101 dataset and it can achieve 69.1% accuracy. Pre-training ResNet-18 on ImageNet can significantly reduce the overfitting problem of the model and increase the accuracy from 47.9% to 69.1%. We train optical flow CNN from scratch.

In our RGB LSTM setting, raw video is split into an image sequence. A fine-tuned ResNet-18 network is used to extract features from video frames. The input to LSTM is features extracted from each video frame. The input and output size of LSTM is 512. The bi-directional LSTM model contains one forward LSTM layer and one backward LSTM layer, each LSTM layer has a residual connection and regularization. Every time step of LSTM follows a linear layer (size: 512, number of classes) and a Logsoftmax layer as the classifier. In our flow LSTM setting, flow features are extracted from flow images by using similar CNN network as RGB channel. The input size of LSTM is 4096 and output size of the LSTM are 512. We only use 1 layer LSTM on top of flow features. Every time-step of LSTM also follows a linear layer and a Logsoftmax layer as the classifier. The detailed training setting is same as RGB LSTM.

Memory in our method can be seen as a classifier. The outputs of both RGB LSTM and flow LSTM from all time steps will be averaged and then fed into the memory module. The output value of memory is the class label. During training and testing, the memory-size is set to 5000, K is set to 16 and the key-size is 1024 (same as the sum of RGB LSTM and Flow LSTM). The entire networks are trained using stochastic gradient descent (SGD) algorithm implemented on a single Titan X GPU.

Results

Prediction Performance We compare our method with dynamic bag-of-words (DBoW) and integral bag-of-words (IBoW) (Ryoo 2011b), C3D (Tran et al. 2015) with Chi-square SVM, MSSC (Cao et al. 2013a), MSDA (Chen et al. 2012), MTSSVM (Kong, Kit, and Fu 2014), and DeepSCN (Kong, Tao, and Fu 2017) on the UCF-101 dataset and the Sports-1M dataset. RGB LSTM and RGB-BiLSTM methods are used as baseline methods. Note that all these comparison methods need to know the observation ratio for accurate prediction, while our method doesn't need it.

UCF-101 dataset. Results in Figure 6(a) show that our method consistently achieves superior results over state-of-the-art action prediction methods on UCF-101 dataset. Our method outperforms DeepSCN by 6% and 3.33% at observation ratios 0.1 and 0.2, respectively. This demonstrates that the memory module in our networks memorize hard-to-predict samples, and thus can help enhance predict performance. As videos in UCF-101 dataset are not difficult to distinguish when more frames are observed, fewer samples are memorized in our network, and thus the performance gap between our method and DeepSCN reduces to around 2% at observations ratios 0.3 to 1.0. Our method also outperforms three existing action prediction methods, IBoW, DBoW and MSSC. All these methods are fed with C3D features. However, all these methods create a model for each observation ratio and do not particularly memorize hard-to-predict samples. In addition, these methods need to know the ratio of a testing sample, and thus is not practically feasible. Our method outperforms IBoW, DBoW and MSSC by 14.72%, 14.72%, and 16.97%, at observation ratio 0.1, respectively. Our method also outperforms C3D+SVM with chi-square kernel. The C3D method performs 3D convolutions on the videos for action recognition, but it is not optimized for action prediction. At observation ratio 0.1, our method outperforms the C3D method by 11.02%, demonstrating that benefits of the memory module and bi-directional LSTM layers in our method for action prediction.

Our method consistently outperforms RGB LSTM and RGB Bi-LSTM, showing that the benefit of using flow stream in the prediction task. At observation ratio 0.1, the performance gap between our method and the two RGB streams (RGB LSTM and RGB Bi-LSTM) is around 1%. However, as more frames are observed, the gap increases to surprisingly 19.06% for RGB LSTM and 15.75% for RGB Bi-LSTM, respectively, demonstrating the importance of using flow stream in the prediction task. This result also shows that flow information may not be very discriminative in the beginning frames of videos. Most videos just have some non-informative motion in the beginning frames, and thus the flow information extracted from these frames are noisy.

Sports-1M dataset. Our method consistently outperforms all the comparison methods. Compared with existing action prediction methods DeepSCN, MTSSVM, MSSC, IBoW and DBoW, our method achieves superior results at all observation ratios. Our method outperforms DeepSCN by 2.58% at observation ratio 0.1. Even though DeepSCN transfers discriminative information from full videos to partial videos, noise may also be transferred which results in negative transfer problem. In addition, the transferring mapping is essentially a linear one, and thus may not fundamentally improve the power of the features from the beginning few frames. By comparison, our method uses the memory module to discover hard-to-predict samples, which allows us to learn a more complex classification boundary. In addition, the bi-directional LSTM in our method uses a nonlinear mapping to transfer information from future frames to current frames, thereby improving the discriminative power of the features. The improvement of our method over MTSSVM and MSSC at observation ratio 0.1

Table 1: Instant prediction results (observation ratio $= 0.1$).

Methods	UCF-101	Sports-1M
MSSC	34.05%	46.70%
Dynamic BoW	36.29%	43.47%
Integral BoW	36.29%	43.47%
MSDA	39.55%	49.27%
C3D + SVM	40.00%	50.83%
MTSSVM	40.05%	49.92%
DeepSCN	45.02%	55.02%
RGB LSTM	50.01%	48.77%
RGB Bi-LSTM	50.01%	53.36%
Our method	51.02%	57.60%

is 7.68% and 10.9%, respectively, which is remarkable. The largest performance gap between our method and IBoW is 18.00% at observation ratio 1.0; and it is 29.24% between our method and DBoW occurred at observation ratio 0.8. Our method also shows significant improvement over RGB LSTM and RGB Bi-LSTM, demonstrating the benefit of using flow stream.

Table 1 shows the *instant* prediction performance where only the beginning 10% frames are observed to all the methods. Our method achieves 51.02% on UCF-101 dataset, and 57.60% on Sports-1M dataset, higher than all the other comparison methods. Thanks to the memory module, our method performs well when fewer frames are given even though they are difficult to be distinguished. Note that our method does not require the observation ratios to be given in testing, while existing methods such as DeepSCN, MTSSVM, MSSC, IBoW and DBoW all need this information. This shows our method is practically feasible.

Table 1 also shows an interesting observation, which is the improvement of our method over the runner-up method on the Sports-1M datatset is slightly greater than the one on the UCF-101 dataset. One possible reason is background information in the Sports-1M dataset is more informative in the beginning frames than that of UCF-101 dataset, and thus can be utilized for discriminating various sports actions. Therefore, the memory on the Sports-1M dataset works more effectively than the one on the UCF-101 dataset as it is expected to remember features from beginning frames.

Comparison with variants We compare our mem-LSTM with several variants in order to show the effectiveness of each module in our method. These variants are ResNet-18, LSTM, Residual LSTM, and Bi-LSTM, flow LSTM, RGB Bi-LSTM+Flow LSTM. All these variants only have RGB stream. Comparison results with our mem-LSTM at observation ratio 0.5 and 1.0 are shown in Table 2.

Our method achieves significant improvement over all the variant methods at observation ratios 0.5 and 1.0. The benefit of using the memory module can be seen in the performance difference between our method and the methods without memory. The improvement is 5.21% and 4.69% at observation ratio 0.5 and 1.0, respectively. This shows that the memory in our method can still memorize some of the challenging samples to be classified in the middle or late stage of videos, thereby creating complex classification boundaries and improving the performance. The benefit of using residual connections in LSTM can be seen from the improvement of Residual LSTM over vanilla LSTM. Adding the residual connections gain about 1% improvement. Bi-LSTM outperforms vanilla LSTM by 2.19% and 3.27%, demonstrating the importance of using a backward LSTM layer on top of the vanilla LSTM. The backward LSTM layer transfers discriminative information from future frames to current frames. Even though the features from current observation may not be discriminative enough, the transferred information can enrich the features, and thus enhance the prediction performance.

Table 2: Comparison results of variants on UCF-101 dataset given the beginning 50% of frames (observation ratio=0.5) and all the frames in testing videos (observation ratio=1.0). $l = x$ means a network contains x LSTM layers.

Networks	Stream	ratio=0.5	ratio=1.0
ResNet-18	RGB	N/A	69.13%
Vanilla LSTM (l=1)	RGB	70.32%	71.37%
Vanilla LSTM (l=2)	RGB	71.58%	73.15%
Residual LSTM (l=2)	RGB	72.51%	74.64%
Bi-LSTM (l=2)	RGB	72.74%	75.47%
Flow LSTM (l=1)	Flow	70.20%	71.65%
Our method (without memory)	Both	83.16%	85.80%
Our method	Both	88.37%	90.49%

We also test the performance of the flow stream in this experiment. A vanilla LSTM is run on a series of flow features extracted using ResNet-18. Results show that the flow stream achieves similar performance to RGB stream (vanilla LSTM). We also test the performance of Bi-LSTM on flow streams. However, its performance is relatively low, possibly because noise from optical flow is transferred from future frames to current frames.

Conclusion

In this paper, we have proposed a novel action prediction network aiming at optimizing the performance at the beginning stage of videos. We introduce a memory module, which remembers hard-to-predict samples. By minimizing the memory loss, the network updates memory keys by either replacing a key with the query or averaging the key and the query. This essentially allows us to learn complex classification boundaries, which is particularly useful for differentiating features from beginning few frames. To further enhance the discriminative power of the features, we propose to utilize information from future by a bi-directional LSTM. We further add residual connections to the LSTM, and regularize the hidden layers by adding a constraint to the LSTM network. Extensive experiments on UCF-101 and Sports-1M datasets show that our method achieves superior prediction performance over state-of-the-art methods, especially when only few beginning frames are observed. Different from most of existing action prediction approaches, our method doesn't need to know observation ratios of testing samples. This appealing feature makes our approach practical feasible.

Acknowledgement

This research is supported in part by the NSF IIS award 1651902, ONR Young Investigator Award N00014-14-1-0484, and U.S. Army Research Office Award W911NF-17-1-0367.

References

Brox, T.; Bruhn, A.; Papenberg, N.; and Weickert, J. 2004. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)*, volume 3024 of *Lecture Notes in Computer Science*, 25–36. Springer.

Cao, Y.; Barrett, D.; Barbu, A.; Narayanaswamy, S.; Yu, H.; Michaux, A.; Lin, Y.; Dickinson, S.; Mark Siskind, J.; and Wang, S. 2013a. Recognize human activities from partially observed videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

Chen, M.; Xu, Z.; Weinberger, K.; and Sha, F. 2012. Marginalized denoising autoencoders for domain adaptation. In Langford, J., and Pineau, J., eds., *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12. New York, NY, USA: ACM. 767–774.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.

Donahue, J.; Hendricks, L. A.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; and Darrell, T. 2014. Long-term recurrent convolutional networks for visual recognition and description. *CoRR* abs/1411.4389.

Fernandez, S.; Graves, A.; and Schmidhuber, J. 2017. Sequence labelling in structured domains with hierarchical recurrent neural networks. *IJCAI*.

Gers, F. A.; Schmidhuber, J.; and Cummins, F. 2000. Learning to forget: Continual prediction with lstm. *Neural computation* 12(10):2451–2471.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Kaiser, L.; Nachum, O.; Roy, A.; and Bengio, S. 2017. Learning to remember rare events. *ICLR*.

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

Kong, Y., and Fu, Y. 2016. Max-margin action prediction machine. *TPAMI* 38(9):1844 – 1858.

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

Kong, Y.; Tao, Z.; and Fu, Y. 2017. Deep sequential context networks for action prediction. In *CVPR*.

Krueger, D., and Memisevic, R. 2016. Regularizing rnns by stabilizing activations. *ICLR*.

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

Lan, T.; Chen, T.; and Savarese, S. 2014b. A hierarchical representation for future action prediction. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*, 689–704.

Laptev, I. 2005. On space-time interest points. *Int. J. Comput. Vision* 64(2-3):107–123.

Ranzato, M.; Szlam, A.; Bruna, J.; Mathieu, M.; Collobert, R.; and Chopra, S. 2014. Video (language) modeling: a baseline for generative models of natural videos. *CoRR* abs/1412.6604.

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

Ryoo, M. S. 2011b. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, 1036–1043. Washington, DC, USA: IEEE Computer Society.

Simonyan, K., and Zisserman, A. 2014. Two-stream convolutional networks for action recognition in videos. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc. 568–576.

Song, S.; Lan, C.; Xing, J.; Zeng, W.; and Liu, J. 2017. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 4263–4270.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR* abs/1212.0402.

Srivastava, N.; Mansimov, E.; and Salakhudinov, R. 2015. Unsupervised learning of video representations using lstms. In Blei, D., and Bach, F., eds., *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 843–852. JMLR Workshop and Conference Proceedings.

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

Wang, H., and Schmid, C. 2013. Action recognition with improved trajectories. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, 3551–3558. Washington, DC, USA: IEEE Computer Society.

Wang, H.; Klaser, A.; Schmid, C.; and Liu, C.-L. 2011. Action recognition by dense trajectories. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, 3169–3176. Washington, DC, USA: IEEE Computer Society.

Wu, Z.; Wang, X.; Jiang, Y.-G.; Ye, H.; and Xue, X. 2015. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, 461–470. New York, NY, USA: ACM.

Yang, X.; Molchanov, P.; and Kautz, J. 2016. Multilayer and multimodal fusion of deep neural networks for video classification. In *Proceedings of the 2016 ACM on Multimedia Conference*, MM '16, 978–987. New York, NY, USA: ACM.