Temporal Domain Neural Encoder for Video Representation Learning

Hao Hu[1], Zhaowen Wang[2], Joon-Young Lee[2], Zhe Lin[2] and Guo-Jun Qi[1]

[1]University of Central Florida
[2]Adobe System Inc
haohu@cs.ucf.edu, {zhawang,jolee,zlin}@adobe.com, GuoJun.Qi@ucf.edu

Abstract

We address the challenge of learning good video representations by explicitly modeling the relationship between visual concepts in time space. We propose a novel Temporal Preserving Recurrent Neural Network (TPRNN) that extracts and encodes visual dynamics with frame-level features as input. The proposed network architecture captures temporal dynamics by keeping track of the ordinal relationship of co-occurring visual concepts, and constructs video representations with their temporal order patterns. The resultant video representations effectively encode temporal information of dynamic patterns, which makes them more discriminative to human actions performed with different sequences of action patterns. We evaluate the proposed model on several real video datasets, and the results show that it successfully outperforms the baseline models. In particular, we observe significant improvement on action classes that can only be distinguished by capturing the temporal orders of action patterns.

1. Introduction

Video representation learning is a very active research area due to its fundamental role in many computer vision tasks. It attempts to close the huge performance gap between the state-of-the-art recognition systems and human beings. Inspired by the tremendous success of the Convolutional Neural Networks (CNN) in learning image representations [15, 21, 25, 11, 8], recent works focus on generalization the CNN architectures to learn high quality video representations [14, 26, 28, 20, 31]. Although these approaches make a significant progress in learning better video features, the performance on many tasks has a large gap to what has been achieved on the image-related tasks such as image classification [15, 21, 25, 8], human face recognition [18, 16] and human pose estimation [2, 30].

One of the possible reasons making video representation learning so challenging is video clips usually contain very rich global and local temporal information which is essential for distinguishing different videos with similar frame level information. For example, videos of opening/closing a door can be easily classified by the temporal information on when the door leaves/approaches the wall, while it is hard to classify them solely based on the visual information of individual frames. Most CNN-based video features handle the challenge by leveraging the local temporal information between consecutive frames [28, 14, 20, 5]. Some other methods employ Long-Short-Term-Memory (LSTM) [9] to capture the long-term dependencies between frames [4, 23]. However, the LSTM model encodes its output as a nonlinear function of frame-level image features, which still limits its capability to model how the temporal orders of action patterns would impact the recognition tasks.

This inspires us to encode global temporal information into video representations. Specifically, we present the Temporal Preserving Recurrent Neural Network (TPRNN) by generalizing the idea behind the First-Take-All framework [10], a novel representation that learns discrete-valued representations of sequential data by encoding the temporal orders of latent patterns. The proposed TPRNN architecture is designated to extract rich patterns over an entire video sequence and encode them as compact video features in ordered temporal structures. Different from the LSTM that is designed to memorize the long-term dependencies between frames, the proposed TPRNN generates video features directly from the ordinal relationships between action patterns in the temporal domain. Compared to frame-level features, the TPRNN features can be more discriminative in recognizing videos captured in the same context (e.g., background, objects and actors) but comprising different sequences of ordered action patterns.

To evaluate the proposed TPRNN model, we conduct extensive experiments on two action recognition datasets, UCF-101 [22] and Charades [19]. To verify the effectiveness of the model in encoding temporal orders of actions,

we construct new action classes by reversing the original videos and test if the model can distinguish the reverse action from its original counterpart. Our experiment results show that the proposed TPRNN model outperforms the LSTM model on both datasets. Moreover, the TPRNN features also significantly improve the performance of the frame-level CNN features and LSTM on recognizing action classes that are only distinguishable by different temporal orders of action patterns.

The rest of this paper is organized as follows. Section 2 reviews related literature on video representation learning. Then we introduce the proposed TPRNN architecture in Section 3. We show our experiment results on video action datasets in Section 4, and summarize the paper in Section 5.

2. Related Works

Deep features acquired by CNN-based architectures achieved great success in many computer vision applications such as image classification [15, 21, 25, 8], face recognition [18, 16], pose estimation [2, 30], etc. Due to its superior performance compared to conventional methods, recent works tend to expand the application of CNN features to a wider range of areas. In terms of video representation learning, several recent works have investigated the question of how to leverage temporal information in addition to the frame level spatial information. These efforts can be roughly divided into two different categories. One is to extend the convolution operation to the temporal axis with 3 dimensional filters learn spatiotemporal features. For example, [12] tries to stack consecutive video frames and extend 2D CNN into time axis. While [14] studies several approaches for temporal sampling shows they cannot encode the temporal information effectively as they only report a marginal improvement over a spatial CNN. Moreover, C3D method [28] introduces a architecture similar to [21] but allowing all filters to operate over spatial and temporal domain. [24] introduces another way to learn spatiotemporal features by factorizing 3D convolution into a 2D spatial and a 1D temporal convolution.

Another way to encode temporal information is represented by the two stream approach originally proposed by [20]. The method decomposes a video clip into spatial and temporal components and feeds them into two separated CNN architectures to learn spatial and temporal representations separately. Then the final prediction is based on the late fusion of the softmax scores from both streams. Rather than performing late fusion at softmax outputs, [5] studies several different fusion strategies including both spatial fusion (sum, concatenation, bilinear, etc.) temporal fusion (3D pooling). The fused features produced by the two stream approach is shown very effective in action recognition and has been deployed into several action recognition methods [1, 6].

Besides CNN-based features, some other methods employ LSTM [9] to encode the long-term dependencies between frames into the video features. A typical way to do this is Long-Term Recurrent Convolutional Network (LRCN) [4], which is the most closely related work to ours. LRCN combines CNN structure and LSTM into a unified framework and can be trained in an end-to-end fashion. Similar works include [31] and [29]: [31] investigates ways to pool temporal features across longer time span, as well as sequential modeling with deeply stacked LSTM, while [29] fuses different types of features including two stream, stacked LSTM, spatial pooling and temporal pooling to make final prediction. Moreover, [23] treats LSTM layer as an autoencoder and learns video representations in an unsupervised fashion. Although LSTM can discover long-range temporal relationships between frames, the resultant video features are still from spatial feature space. On the contrary, First-Take-All hashing [10] encodes temporal order information directly from time space and can achieve good performance on time sensitive datasets.

3. Temporal Preserving Recurrent Network

In this section, we introduce Temporal Preserving Recurrent Network, a novel RNN-based model which is designed to encode temporal structure information between video frames. First, we explain the design principles for the proposed network along with the connection between the aforementioned First-Take-All Hashing [10], then we present the architecture of the proposed network including the definition of each layer and its functionality. Finally, we compare the proposed network with other recurrent networks, which share similar structures.

3.1. Intuition

We design the proposed Temporal Preserving Recurrent Network based on the inspiration from the First-Take-All (FTA) hashing [10], which employs the temporal order information to compactly represent videos. In FTA, a multi-dimensional process is projected to a latent subspace at each time step, generating a set of 1D latent temporal signals. The occurrences of the maximal values are compared among all the latent signals, and the one with the first maximum occurrence is used to index the hash code. However, there are several drawbacks with the FTA formulation. (1) Only the index of the first-appearing patterns is used to represent the sequence. Others are ignored which may also contain useful information. (2) In the FTA comparison, The only learnable parameter is the linear projection. The number of projections and the dimension of each projection are also determined heuristically. Therefore, the learning capability and scalability of FTA is limited. (3) Since FTA is an hashing algorithm, the binary nature of the FTA codes

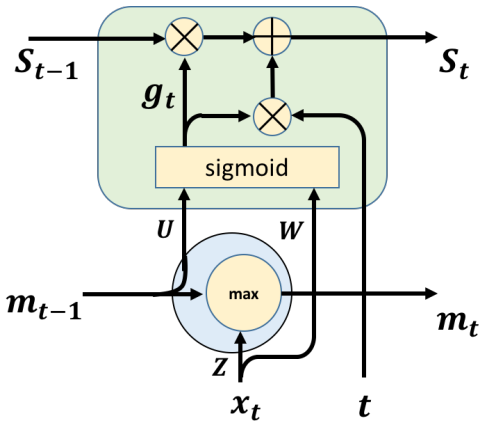


Figure 1: Structure of temporal preserving network, where the circle represents the max-pooling layer and the rounded corner rectangle represents the temporal preserving layer.

prevents them from representing input sequences more accurate than those floating-point features.

To address the weaknesses of FTA hashing, we reformulate the FTA comparison in a recursive fashion, so that it can be implemented with a computational model such as recurrent neural network. Denote the original multi-dimensional process as $\{\mathbf{x}_t\}$, and the linear projection as $\mathbf{Z}$, the running maximum $\mathbf{m}_t$ of projected latent signals can be obtained by

$$\mathbf{m}_t = \max(\mathbf{Z}\mathbf{x}_t, \mathbf{m}_{t-1}). \tag{1}$$

All scalar functions such as $\max(\cdot)$ are applied in an element-wise way unless otherwise stated. The time steps $\mathbf{s}_t$ when the maximal values $\mathbf{m}_t$ first occurred so far can be recorded as

$$\mathbf{s}_t = \begin{cases} \tilde{t}, & \text{if } \mathbf{Z}\mathbf{x}_t > \mathbf{m}_{t-1}, \\ \mathbf{s}_{t-1}, & \text{otherwise.} \end{cases} \tag{2}$$

where $\tilde{t}$ stands for the normalized version of time index in $[0, 1]$. After recursive updating with equations 1 and 2, at the final step T, FTA calculates a binary hash code based on the index $\arg\min_i \mathbf{s}_T(i)$. Note that the conditional expression of $\mathbf{s}_t$ can be controlled by a logic gate $\sigma(\mathbf{Z}\mathbf{x}_t - \mathbf{m}_{t-1})$, where $\sigma(\cdot)$ is a sigmoid function approximating the hard boolean operation.

3.2. Model Architecture

Based on the components of FTA, we propose a Temporal Preserving Recurrent Neural Network (TPRNN) with stronger temporal modeling capability. TPRNN is a RNN-based network with evolving memory cells connected in the same way at each time step. An illustration of the network structure at one time slice is shown in Figure 1. An input $\mathbf{x}_t$ (video frame feature) is fed into the network together with the current time step t, and the hidden states $\mathbf{m}_t$, $\mathbf{s}_t$ are updated with the control of gate $\mathbf{g}_t$. There are several components in the network with distinct functionality as described in below.

Max-pooling Layer The bottom circle in Figure 1 is a max-pooling layer over adjacent time steps. Same as equation 1 for FTA, this layer evaluates the latent patterns of input feature $\mathbf{x}_t$ with linear projection $\mathbf{Z}$, and stores the current maximal projected values in state $\mathbf{m}_t$ by comparing with the previous state $\mathbf{m}_{t-1}$.

Temporal-preserving Layer This layer in the rounded corner rectangle of Figure 1 plays a key role in constructing the temporal preserving representation. It keeps track of important time moments t in state $\mathbf{s}_t$ with a control gate $\mathbf{g}_t$:

$$\mathbf{s}_t = (1 - \mathbf{g}_t) \cdot \mathbf{s}_{t-1} + \mathbf{g}_t \cdot \tilde{t}, \tag{3}$$

The gate $\mathbf{g}_t$ is a sigmoid function which is activated when the most salient moment of a latent event is detected:

$$\mathbf{g}_t = \sigma(\mathbf{W}\mathbf{x}_t - \mathbf{U}\mathbf{m}_{t-1}), \tag{4}$$

where the current input feature $\mathbf{x}_t$ is compared with the previous maximal response $\mathbf{m}_{t-1}$ in latent spaces spanned by matrices $\mathbf{W}$ and $\mathbf{U}$.

The temporal-preserving layer introduces a few extensions based on the FTA comparison in equation 2. First, rather than following a hard activation condition, we employ a soft activation function to monitor latent patterns, which is differential and more sensitive to subtle temporal changes. Similar to the gates in LSTM, $\mathbf{g}_t$ measures the likelihood of a latent pattern occurrence. It controls how much to forget about the previous state $\mathbf{s}_{t-1}$ and how much to remember for the current time step t. Such soft updating scheme enables multiple time steps to contribute as the occurrences of salient events. As a result, the temporal-preserving layer is capable of leveraging more high-order temporal information than solely considering the maximal response moments.

More importantly, two additional parameters $\mathbf{W}$ and $\mathbf{U}$ are introduced in equation 4 for better modeling of complex dynamic visual events. The projection $\mathbf{U}$ is applied on the max-pooling state $\mathbf{m}_{t-1}$ to extract more semantic information from input feature. The projection $\mathbf{W}$ together with the original parameter $\mathbf{Z}$ provide two different latent spaces of $\mathbf{x}_t$ for activating gate and detecting maximal response, sharing similar motivation as the key/value addressing mechanism in Neural Turing Machine [7]. If we set $\mathbf{U} = \mathbf{I}$ and $\mathbf{W} = \mathbf{Z}$, the temporal-preserving layer will reduce to FTA comparison.

Output Layer At the last time step T, the hidden state $\mathbf{s}_T$ accumulates the occurrence information of latent patterns in all previous steps. Thus, each dimension of $\mathbf{s}_T$ represents the expected time step when a latent pattern occurs. To encode the temporal order relationship among different visual patterns, we apply a weighted softmax layer to $\mathbf{s}_T$ to get the output representation $\mathbf{o}$ of video sequence:

$$\mathbf{o} = softmax(\mathbf{Y}\mathbf{s}_T) = \frac{\exp(\mathbf{Y}\mathbf{s}_T)}{\sum_i \exp(\mathbf{Y}(i)\mathbf{s}_T)}, \qquad (5)$$

where $\mathbf{Y}$ is output weight matrix and $\mathbf{Y}(i)$ is the i-th row of $\mathbf{Y}$. Each row of $\mathbf{Y}$ selects two or more latent patterns and compares their relative temporal order through a learned linear combination. The strength of all the ordinary relationships is normalized to a probability vector by a softmax function. In contrast to FTA, equation 5 gives us more flexibility to encode temporal information. The total number of ordinal relationships to encode is controlled by the number of rows in $\mathbf{Y}$, and the number of latent patterns involved in each comparison is controlled by the number of non-zero entries in the rows of $\mathbf{Y}$. An ℓ_1 regularization can be used to control the sparsity of $\mathbf{Y}$.

For a further understanding of the proposed model, we give an intuitive example of how each component works in TPRNN. In the max-pooling layer, the projected value of $\mathbf{x}_t$ indicates the likelihood of visual concepts, such as arm and forearm, appearing in current video frame. The most prominent responses of visual concepts are kept in the state $\mathbf{m}_t$. The temporal-preserving layer projects the visual concepts into some higher-order subspace with $\mathbf{W}$ and $\mathbf{U}$, capturing information such as the pose of elbow (angle between arm and forearm). When the elbow pose changes significantly (with arms stretching or folding), the gate $\mathbf{g}_t$ will be activated and the current event moment will be memorized in $\mathbf{s}_t$. By aggregating all the time steps when elbow pose changes and comparing with other correlated poses such as shoulder movement, the output representation $\mathbf{o}$ can be useful to characterize videos containing boxing activity which requires joint elbow and shoulder motion. The proposed model mainly relies on dynamic order information to represent and distinguish videos, which is why we call it temporal-preserving network.

3.3. Comparison with other RNNs

We compare the proposed network with the other variants in the RNN, which include the conventional LSTM [9] and Gated Recurrent Unit (GRU) [3]. Similar to these models, the proposed TPRNN also employs the activation gate to forget and store useful information in the hidden states. However, compared to both LSTM and GRU, there are several differences in the proposed TPRNN model. A major difference in the proposed TPRNN model is the encoding space. Rather than learning temporal dependencies from the frame-level (spatial) feature space, TPRNN intends to capture temporal order structures directly from the time space. This allows video sequences to be represented from a new perspective that totally different from using the spatial features. If two video sequences contain the same visual concepts but only with different orders (for example, open/close a door), the temporal order information is very useful to distinguish their differences. In such cases, the feature generated by TPRNN can be a great complement for spatial features.

	# of gates	# of parameters
LSTM	4	$4 * (n * d + n^2)$
GRU	2	$3 * (n * d + n^2)$
TPRNN	1	$2 * n * d + n^2$

Table 1: Structural differences between LSTM, GRU and TPRNN. Here n represents hidden states dimension and d represents input dimension.

Another straightforward difference is the proposed TPRNN has a simpler structure than LSTM and GRU. Table 1 summarizes the distinction between three structures in terms of the number of activation gates and the number of parameters. We can see that LSTM has the most complicated structure with the most activation gates and parameters, while TPRNN only contains less than half of its parameters with only one gate. This makes TPRNN more invulnerable to overfitting.

4. Experiments

We evaluate the proposed TPRNN representations by performing video classification tasks on two public available datasets: UCF-101 [22] and Charades [19]. The evaluation aims to validate the properties of TPRNN from three aspects. First, we compare TPRNN with conventional LSTM structure to demonstrate the advantage of TPRNN encoding temporal order structures. Then, we analyze the performance of TPRNN on time-sensitive classes. At last, we prove that the TPRNN feature can be a good complement of the spatial features by fusing TPRNN features with frame level features.

4.1. Datasets

We employ two video benchmarks to evaluate the proposed TPRNN model: UCF-101 [22] and Charades [19]. UCF-101 dataset includes 13320 videos from 101 action categories with average over 150 frames per video. Each video clip in dataset is segmented to exactly contain one of 101 categories. Charades dataset contains 9848 videos of daily indoors activities with temporal annotations for 157

action classes. Unlike UCF-101, each video clip in Charades dataset may contain multiple action classes in different temporal locations. For UCF-101, We use split-1 with 9537 and 3783 videos as training and testing samples, to conduct our experiments. And for Charades, we exclude videos without any action labels so the final version of the dataset in our experiments contains 7811 training and 1814 testing samples, respectively.

4.2. Implementation and Training

We implement both the conventional LSTM and the proposed TPRNN with Theano [27] python math library. Note that the time scale is normalized to 1 for all video clips such that the occurrences of all visual concepts are in $[0, 1]$. The deep features used in the experiments come from several different Convolutional Neural Network (CNN) models including AlexNet [15], VGG [21] and LRCN-single-frame [4]. It is worth mentioning that the VGG models we use to compute RGB (spatial) and flow (temporal) features are provided by [5] which is also fine-tuned on UCF-101, while the adopted AlexNet is pre-trained on ImageNet [17]. All these features are computed with Caffe [13] framework.

We compare the TPRNN features with another two baselines. One is using frame features to represent video clips by averaging across all frames. Another is feeding frame features to LSTM to learn long-term frame dependencies, then average the outputs across all time steps to get video representations. We feed these features into a linear classifier and produce the softmax score for each class. The weights of linear classifier can be learned along with TPRNN by minimizing the cross-entropy loss.

Although TPRNN can be trained along with the CNN in an end-to-end fashion, we fix the CNN weights to compute spatial features and only update weights of LSTM and TPRNN to focus our experiments on the recurrent module. This makes sure we evaluate the impact of encoding temporal order information without changing the spatial inputs. We follow the different frame sampling protocols specified by [5] and [19] on UCF-101 and Charades respectively. During the training phase, the first frame of each video is randomly cropped with an input size of the CNN networks then the same spatial crop is applied to all frames. Specifically, for Charades dataset, we follow the setup in [19] that only train the models with untrimmed intervals which don't include action localization information for multi-labeled video clips. Unless otherwise specified, we employ the central crops to do the testing and the number of hidden states and batch size are always set to 200 and 16 videos per batch, respectively.

At last, we adopt prediction accuracy as evaluation metric for the single label cases of UCF-101 while we adopt mean average precision to handle multi-label cases of Charades.

UCF-101	LSTM	TPRNN
VGG-16-fc6	0.7766	**0.7861**
VGG-16-fc7	0.7769	**0.7938**
VGG-16-flow-fc6	0.8039	**0.8118**
VGG-16-flow-fc7	0.8007	**0.8107**

Table 2: Recognition Accuracy on UCF-101 dataset

Charades	LSTM	TPRNN
AlexNet-fc6	0.1027	0.1061
AlexNet-fc7	0.0996	0.1119

Table 3: Mean Average Precision on Charades dataset

4.3. LSTM vs. TPRNN

We compare TPRNN with the conventional LSTM by generating video representations with various frame level spatial features. Table 2 and 3 demonstrate the comparison results on both datasets. From tables we can see the proposed TPRNN outperforms the conventional LSTM with most input spatial features. On UCF-101, we also test both methods additionally with flow image inputs computed by [5]. Table 2 shows that TPRNN achieves around 1% better performance than LSTM with all input features from different fully connected layers and RGB/flow images. We can also observe that the improvement from LSTM to TPRNN is more obvious when using RGB frames. This is reasonable since RGB features only contain static spatial information, while flow features already have some local motion information. So TPRNN seems producing less benefit to the flow inputs. During the training, we also find LSTM more sensitive to the overfitting since there are much more weights in LSTM as discussed in section 3.3.

Similar results can be observed on Charades dataset, where TPRNN can also outperform LSTM with both fc6 and fc7 features from AlexNet. However, compared to UCF-101, the gap between LSTM and TPRNN is much smaller with features from fc6 than fc7. Note that during the training phases on the Charades dataset, we perform untrimmed training which doesn't utilize any action localization information provided by training set. Thus learning long-term dependencies between input frames may be not as effective as encoding visual concept occurrences along the time domain, since visual concepts occurrences can be served as some boundary points for trimmed interval presenting interested actions.

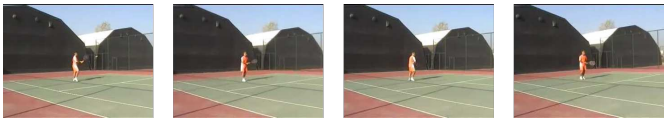
4.4. Analysis on Subsets

Although we demonstrate the TPRNN can achieve better performance on both datasets, it is beneficial to analyze

Class	LSTM	TPRNN	Performance Gain
Cliffdiving	0.6410	0.9359	+0.2949
HighJump	0.5676	0.8378	+0.2702
CleanAndJerk	0.5000	0.8182	+0.3182
BalanceBeam	0.5697	0.7742	+0.2045
PoleVault	0.7125	0.7125	+0.0000
CricketBowling	0.5417	0.6667	+0.1240
LongJump	0.6154	0.6538	+0.0384
BlowingCandles	0.5000	0.5152	+0.0152
TennisSwing	0.5000	0.5102	+0.0102
Rowing	0.5000	0.5000	+0.0000
Overall	0.5697	0.6877	+0.1180

Table 4: Accuracy of predicting reverse/unreverse on 10 classes of UCF-101

(a) CliffDiving



(b) TennisSwing

(c) Rowing

Figure 2: Examples of classes with different foreground and background variations

Group 1– Opening/Closing *sth.*

	Spatial	LSTM	TPRNN
Door	0.2671	0.2662	0.3261
Box	0.0410	0.0446	0.0357
Laptop	0.0186	0.0252	0.0440
Closet/Cabinet	0.1612	0.1704	0.1660
Refrigerator	0.1673	0.0984	0.1712
MAP	0.1311	0.1209	0.1486

Group 2– Taking/Putting *sth.* somewhere

	Spatial	LSTM	TPRNN
Bag	0.0441	0.0457	0.0530
Shoes	0.0379	0.0421	0.0525
Sandwich	0.0305	0.0304	0.0367
Blanket	0.0857	0.0806	0.0972
Broom	0.0416	0.0381	0.0482
MAP	0.0480	0.0474	0.0575
Overall MAP	0.0896	0.0842	0.1031

Table 5: Average Precision (AP) and Mean Average Precision (MAP) of each class pairs and action type group. *sth.* indicates different visual objects

which video classes can benefit more from temporal features of TPRNN. Based on [5] which achieves over 80% recognition accuracy with only spatial VGG features on UCF-101, we can see most of action types can be discriminated well with only spatial context (e.g. background). In order to evaluate the discriminability based on temporal differences without too much interference of spatial context information, we first reverse the frame orders for all video clips to add another 101 classes to the original dataset (total 202 classes after reverse). So the reversed classes can only be distinguished by their temporal order differences. Then we train both LSTM and TPRNN plus using only spatial features as baseline on the dataset with fine-tuned VGG-16 model used in 4.3, and test with fc7 layer features for each of the original classes by predicting whether a sample from that class is reversed or not.

We test and report the original/reversed prediction results for 10 representative UCF-101 classes whose video clips are with different level of foreground, background changes and camera variations. For example, video clips in CliffDiving class begin with a background of cliff but end with a background of water, while video clips of BlowingCandles usually have a static background with relatively small region of interest (candle fire, etc.). As it is a binary classification problem, using only spatial features output =0.5 accuracy for all 10 classes, as the spatial features are exactly symmetric for test samples. This indicates it is impossible to distinguish the original/reversed clips without any long-term or temporal information. Other results are summarized in Table 4, which includes prediction accuracy and performance gain of TPRNN over LSTM for individual and the overall classes. We can see that LSTM performs obviously better than random guess on around half of classes but still produces poor results for another half. In contrast, TPRNN outperforms LSTM with a significant margin on most of classes. However, on some classes it performs close to a random guess. We notice that in the classes where TPRNN and LSTM are significantly better than random, video clips usually contain distinct background or camera variations throughout the frames. For Instance, as shown in Figure 2a, video clips of CliffDiving class always start with a person standing height. After the action begins, the camera will track the person dropping from height until he/she gets into water. So there will be notable background order difference

UCF-101	Spatial	TPRNN	Late Fusion
LRCN[4]-single-fc7	0.6952	0.7112	0.7187
VGG-16-RGB-fc7	0.7893	0.7938	0.8057
VGG-16-flow-fc7	0.8096	0.8107	0.8197
Charades	Spatial	TPRNN	Late Fusion
AlexNet-fc7	0.1034	0.1119	0.1136

Table 6: Late fusion results on UCF101 and Charades

between original and reversed classes, making them much easier to be classified by TPRNN. In such cases, the action region is very small and is not a crucial factor to classify the video. Besides, for those classes which TPRNN performs similarly poor as TennisSwing (Figure 2b), we can see that such actions often take place at some static locations such as tennis ground, etc. Thus background variations contribute very little in temporal order differences and TPRNN will rely on the variations of much smaller action regions (poses), making it less beneficial for these classes. What's more, Rowing class represents classes whose video clips are temporally symmetric. As shown in Figure 2c, the background is almost static and the action region varies periodically. In such cases, encoding the temporal differences may not characterize video clips well and let the TPRNN features perform like random guess. Similar behavior also can be observed on the remaining classes.

Unlike UCF-101, Charades dataset contains many class pairs that naturally with forward and backward orders, e.g. closing/opening a door. Moreover, the classes in such pairs can be further aggregated into some action types like putting something somewhere and take something from somewhere, etc. These action types share similar temporal order patterns only with different visual concepts. We train all three methods (spatial classifier, LSTM, TPRNN) on entire dataset and then calculate the Average Precision (AP) for each class. We report testing results on two different pair groups. Each group contains 5 class pairs in forms of the same action types but with different visual objects. For example, action type of group 1 is opening/closing something while the one of group 2 is putting/taking something somewhere. Table 5 demonstrates the AP for each pair as well as their Mean Average Precision (MAP) for each group, where the AP of each pair is the average of two classes. As we expected, TPRNN works better in most of class pairs and clearly boost the overall performance.

4.5. Fuse with Spatial Features

The TPRNN representation is designed to capture more temporal information and therefore is expected to be a good complement for spatial feature. To verify this argument, we combine it with spatial features and test the performance boost on the same recognition tasks. We compare the spatial, TPRNN features and their late fusion results in Table 6 with different frame feature inputs. Note that for spatial results, we average all frame features of each video clip to generate a fixed-size video feature and then perform the classification, while for late fusion, we follow the fusion setting of [20] by averaging the prediction score of Spatial features and TPRNN features to get the final prediction score. For each CNN architecture, we experiment with features from both fc6 and fc7 layer but only report fc7 results since fc6 results are quite similar. We can see that on UCF-101, late fusion with spatial features and TPRNN features achieves different level of boosting. In cases that TPRNN achieves same level performance as spatial features (VGG-16-RGB and flow), late fusion improves about 1% recognition accuracy than single feature, while when TPRNN performs better than spatial ones such as using LRCN-single frame inputs, fusion results are less boosted because of the discriminative gap between two types of features. Similar results can be observed on Charades dataset. Such results coincide with the expectation that the fusion with two features can achieve better performance.

5. Conclusion

This paper presents a novel Temporal Preserving Recurrent Network (TPRNN) that aims to learn video representation directly from the temporal domain. The proposed network architecture models the temporal order relationships between visual concepts by leveraging their occurrences from spatial feature inputs. The resultant video features provide a new way to characterize video clips with temporal information only. Compared to other RNN structure such as LSTM [9] and GRU [3], TPRNN has simpler inner structure with less parameters which makes it more invunerable to overfitting. The structure design also let the TPRNN overcome the shortcomings that First-Take-All [10] hashing suffers and be able to leverage much more temporal order information in the video representation. We evaluate the proposed TPRNN model on UCF-101 and Charades dataset for action recognition with extensive experiments. The results indicate the proposed TPRNN model outperforms the conventional LSTM and can further improve by combining the spatial features. In particular, significant performance boost is achieved for action classes only are distinguishable by temporal orders.

References

[1] G. Chéron, I. Laptev, and C. Schmid. P-cnn: Pose-based cnn features for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3218–3226, 2015. 2

[2] X. Chu, W. Ouyang, H. Li, and X. Wang. Structured feature learning for pose estimation. *arXiv preprint arXiv:1603.09065*, 2016. 1, 2

[3] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 4, 7

[4] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 1, 2, 5, 7

[5] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. *arXiv preprint arXiv:1604.06573*, 2016. 1, 2, 5, 6

[6] G. Gkioxari and J. Malik. Finding action tubes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 759–768, 2015. 2

[7] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014. 3

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 1, 2

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1, 2, 4, 7

[10] H. Hu, J. Velez-Ginorio, and G.-J. Qi. Temporal order-based first-take-all hashing for fast attention-deficit-hyperactive-disorder detection. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 905–914, New York, NY, USA, 2016. ACM. 1, 2, 7

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 1

[12] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013. 2

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 5

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 1, 2

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 1, 2, 5

[16] H. Qin, J. Yan, X. Li, and X. Hu. Joint training of cascaded cnn for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3456–3465, 2016. 1, 2

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 5

[18] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015. 1, 2

[19] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, 2016. 1, 4, 5

[20] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014. 1, 2, 7

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1, 2, 5

[22] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 1, 4

[23] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR, abs/1502.04681*, 2, 2015. 1, 2

[24] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4597–4605, 2015. 2

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 1, 2

[26] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *European conference on computer vision*, pages 140–153. Springer, 2010. 1

[27] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. 5

[28] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497. IEEE, 2015. 1, 2

[29] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 461–470. ACM, 2015. 2

[30] W. Yang, W. Ouyang, H. Li, and X. Wang. End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation. CVPR, 2016. 1, 2

[31] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4694–4702, 2015. 1, 2