# RhyRNN: Rhythmic RNN for Recognizing Events in Long and Complex Videos

Tianshu Yu†, Yikang Li†, and Baoxin Li

Arizona State University, USA
{tianshuy,yikangli,baoxin.li}@asu.edu

**Abstract.** Though many successful approaches have been proposed for recognizing events in short and homogeneous videos, doing so with long and complex videos remains a challenge. One particular reason is that events in long and complex videos can consist of multiple heterogeneous sub-activities (in terms of rhythms, activity variants, composition order, etc.) within quite a long period. This fact brings about two main difficulties: excessive/varying length and complex video dynamic/rhythm. To address this, we propose Rhythmic RNN (RhyRNN) which is capable of handling long video sequences (up to 3,000 frames) as well as capturing rhythms at different scales. We also propose two novel modules: diversity-driven pooling (DivPool) and bilinear reweighting (BR), which consistently and hierarchically abstract higher-level information. We study the behavior of RhyRNN and empirically show that our method works well even when *only event-level labels are available* in the training stage (compared to algorithms requiring sub-activity labels for recognition), and thus is more practical when the sub-activity labels are missing or difficult to obtain. Extensive experiments on several public datasets demonstrate that, even *without fine-tuning the feature backbones*, our method can achieve promising performance for long and complex videos that contain multiple sub-activities.

**Keywords:** Video understanding; Complex event recognition; RNN

## 1 Introduction

In recent years, video-based event/activity recognition has brought about enormous and important challenges to computer vision. The research community has devoted considerable effort and made progresses in many related tasks (e.g., action recognition [14, 46, 52, 54, 10, 55, 4, 33, 11], temporal localization [43, 7], video question answering [48, 1], video summarization [19, 34, 66], to name a few). By learning more representative features and capturing stronger sequential context, deep-learning-based methods have delivered the state-of-the-art results on several datasets of short videos (e.g. UCF101 [47], KTH [41], HMDB51

† indicates equal contribution. This work was supported in part by a grant from ONR. Any opinions expressed in this material are those of the authors and do not necessarily reflect the views of ONR.

[30]). Recently, more challenging datasets (e.g. VIRAT [37], Charades [45] and Breakfast [31]), which typically contain video clips with complex and/or multiple sub-activities in a much longer time period, have brought about new challenges to video recognition. To address these, some event recognition algorithms were proposed [12, 51, 25, 57, 64, 23, 59, 39, 13, 63, 16], taking into account either long-time dependency or the activity variation to some extent. In this paper, we investigate a specific RNN structure to understand long and complex videos.

For clarity of discussion, we make a distinction between activity and event. Consider one example for each. "Jogging", which belongs to activity in our context, exhibits relatively fixed or homogeneous visual pattern and temporal dynamic (repetitive motion in this case). In contrast, "Cooking spaghetti", which is categorized as an event, is composed of multiple sub-activities (e.g., "bringing out condiment", "boiling spaghetti", etc.) that can occur in different rhythm, order or visual appearance, resulting in much more complex scene dynamics for an algorithm to capture. Furthermore, some events can occur over a significantly longer time period than activities. In general, events in long videos brings about two challenges to video-based recognition: complexity in content and excessive/varying length, making it challenging to adapt a traditional activity recognition model designed for much simpler videos.

Another important yet barely investigated issue in video-based recognition is, how to identify video events when *only event-level labels are available* for training a model. This arises often due to lack of detailed labeling information that is difficult and/or costly to obtain for long videos. Though some previous methods incorporate sub-activity labels to enhance event-level recognition [31, 32, 23, 24], such fine-grained labels are not always available in practice due to the aforementioned reason. In general, the event label describing a long video is highly abstract in nature, and it may imply a lot of latent contexts.

In this paper, we seek to make a progress towards long and complex video event recognition (with or without sub-activity labels). We further study a way to perform video-based recognition when only event-level labels are available. To this end, we propose Rhythmic RNN (**RhyRNN**) which dynamically captures the multi-level contexts, as well as a diversity-driven sequential pooling (**DivPool**) and a Bilinear Re-weighting (**BR**) mechanism. The work has the following contributions: 1) We introduce RhyRNN which can ease the gradient back-propagation for long and complex sequences. RhyRNN also allows to capture latent video context at different levels; 2) We develop DivPool and BR strategies, which further enable multi-level feature aggregation (analogous to pooling in CNNs) with varying sequence length; 3) We study the property and behavior of all the proposed modules analytically and empirically; 4) Our method delivered superior or competitive performance in long video datasets compared to the state-of-the-art algorithms even without fine-tuning feature backbones.

## 2 Related Work

**Short activity recognition** Some early video datasets (e.g., KTH [41] and UCF101 [47]) typically contain activity/action-level video clips, which are homogeneous in content without too complex temporal dynamics. A conventional trial for activity recognition employed 2D CNN features to perform recognition [27], while some variants incorporate complementary frame-level motion features [46, 4, 3]. The main drawback of such a line of works is that the temporal patterns cannot be well learned since neither short nor long range dependencies are explicitly taken into account. 3D CNNs are natural extension from 2D by introducing one additional kernel dimension on the time axis [49, 6, 56], but with excessive parameters. To alleviate this, several works were proposed to decouple the 3D kernel into combinations of lower dimension (e.g., [9, 50, 61]). Another line of works in parallel to CNNs employs RNNs [11, 33, 42, 10]. RNNs can handle varying length of videos compared with CNNs, but suffer from gradient vanishing/explosion issue especially when the sequence is too long.

**Complex event recognition** Datasets consisting of long and complex videos bring about new challenges [37, 31, 45]. Extending CNNs for long-range video recognition has become an aroused research interest recently. To capture more complex temporal patterns in long videos, [44] stacks a CRF on top of CNN output. Under some specific sampling procedure, TSN [55] and TRN [67] model the video-level representation by considering inter and intra video relations, respectively. Non-local networks [56] built upon 3D CNN can range up to 128 time steps, hence is capable of handling more complex dynamics. Timeception [23] can further capture the dependencies up to 1024 frames by designing multi-scale convolutional kernels. In parallel to CNNs, RNNs are also investigated to tackle long and varying video length with complex context. [65] considers dense labeling in complex videos, where the expensive part is to densely label the training data. [58] proposed a hierarchical RNN to capture temporal visual attention. Both [33] and [11] devise hierarchical RNN structures to obtain multi-level representation, which proved effective in understanding video content. In [42], soft attention is computed spatially and temporally via deep RNNs, which helps the model to focus selectively on more meaningful parts of a video.

**RNNs** LSTM [22] and GRU [8] are successively proposed to address the gradient vanishing/exploding issue by introducing the gating mechanism against standard RNNs. There is a series of further developments following this strategy [17, 5, 26]. Skip-RNN [5] learns to keep the hidden state intact at some steps once "Skip" is emitted. H-detach [26] detaches the gradient flow at an arbitrary time step under a Bernoulli distribution. Some other efforts focused on the variants of standard RNNs without using gating. Multiplicative Integration [60] couples the operations on inputs and hidden states. In this fashion, the vanishing gradient is likely to be correlated by the input sequence. Unitary-RNN [2] allows smoother gradient flow by constraining RNNs to have a unitary transition matrix. Very recently, IndRNN [35] was proposed, which enforces the neurons in each RNN unit to be independent. By doing so, IndRNN can handle long sequences and achieved state-of-the-art performance on multiple benchmarks.

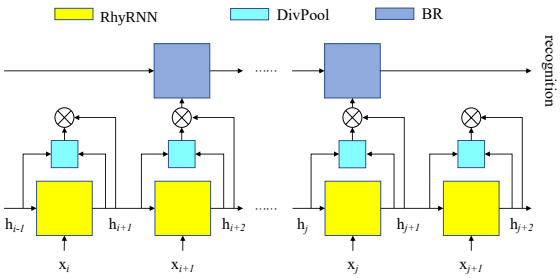


Fig. 1: Overview of our framework which mainly consists of 3 parts: RhyRNN, DivPool and BR (a recognition module). BR refers to the GRU equipped with bilinear re-weighting in our setting.

## 3 Methodology

### 3.1 Algorithm overview

The overview of our framework consisting of three modules is illustrated in Fig. 1. The model takes visual features as input and feeds them sequentially to RhyRNN. RhyRNN outputs embedded features with the same length as the sequence. Using a diversity score, DivPool is then applied to select the most informative features as inputs to the following recognition module. For the final recognition stage, we employ a GRU equipped with BR module. The output of GRU at the final timestamp will be fed to a two-layer fully connected network at last. We detail each part in the following sections. Our approach is motivated by the following considerations. First, our approach should be capable of handling long sequences. To this end, we need to design a specific RNN structure which eases the gradient flow under this setting. Second, since complex events contain latent contexts in different scales, our approach needs to capture such multi-level dynamics. A hierarchical model, in this case, can be a good choice as done in a large body of relevant literature.

### 3.2 RhyRNNs

One essential part of our algorithm is to deploy an architecture that is capable of handling a long and complex sequence. In this section, we propose the RhyRNN structure, which is inspired in part by IndRNN [35] and Skip-RNN [5], and is much more powerful than both (see Sec. 4). IndRNN enforces each neuron operating on the hidden state to be independent, and the update rule of IndRNN reads:

$$\mathbf{h}_t = \sigma\left(\mathbf{W}\mathbf{x}_t + \mathbf{u} \odot \mathbf{h}_{t-1} + \mathbf{b}\right) \tag{1}$$

where $\odot$ is the element-wise product and $\sigma(\cdot)$ is the activation function (ReLU function in [35]). $\mathbf{h}_t$ corresponds to the hidden state at time $t$. It has been shown

that, by enforcing the neuron independence (no matrix multiplication), back-propagation upon Eq. (1) becomes more stable and manageable. IndRNN has delivered good performance for very long sequences.

While IndRNN alleviates gradient vanishing by replacing matrix multiplication with scalar multiplication, we propose RhyRNN to further *shorten the longest path of the computational graph* of IndRNN independently for each neuron, through introducing a *skip* operator. This idea is similar to [5] that implements skip operation on conventional RNN, which can be viewed as a Bernoulli distribution sampler on *UPDATE* or *COPY* operations at each timestamp $t$ (an analogous idea appeared in h-detach [26] which is applied on LSTM). Our RhyRNN differs from Skip-RNN in such a way that, unlike [5] where *UPDATE* and *COPY* operations are computed on a whole hidden state $\mathbf{h}_t$ by a matrix multiplication, our RhyRNN structure decides the choice of *UPDATE* or *COPY* operation by using Hardmard's product, which further makes the decision *independent of each neuron*. The mathematical formula of RhyRNN can be written as follows:

$$\mathbf{s}_t = f_{binarize}(\mathbf{o}_t) \tag{2}$$

$$\mathbf{h}_t = \mathbf{s}_t \odot \tilde{\mathbf{h}}_t + (\mathbf{1} - \mathbf{s}_t) \odot \mathbf{h}_{t-1} \tag{3}$$

$$\Delta\mathbf{o}_t = \zeta(\mathbf{w}_p \odot \mathbf{h}_t + \mathbf{b}_p) \tag{4}$$

$$\mathbf{o}_{t+1} = \mathbf{s}_t \odot \Delta\mathbf{o}_t + (\mathbf{1} - \mathbf{s}_t) \odot (\mathbf{o}_t + \min(\Delta\mathbf{o}_t, \mathbf{1} - \mathbf{o}_t)) \tag{5}$$

where $\zeta(\cdot)$ is the sigmoid activation function and $f_{binarize}$ is the step function: $f_{binarize} : [0, 1]^n \to \{0, 1\}^n$, which binarizes each input element. $\mathbf{w}_p$ is the weight vector that can be learned to obtain the incremental value $\Delta\mathbf{o}_t$. $\tilde{\mathbf{h}}_t$ is obtained by Eq (1) (replacing $\mathbf{h}_t$ with $\tilde{\mathbf{h}}_t$) and $\mathbf{h}_{t-1}$ is the hidden state from the previous timestamp.

**Remark**. There are two advantages of utilizing Hardmard's product in computing the gate value $\mathbf{s}_t$. Firstly, it keeps the independence of each neuron in IndRNN intact, which allows each neuron to have a distinct strategy of choosing *UPDATE*/*COPY* operations and thus being capable of capturing the varying context in different scales. We will demonstrate this advantage in Section 4. Secondly, the computation of the gradient of the RhyRNN is easier and more stable compared to either IndRNN or Skip-RNN, since the lengths of gradient path for different neurons can be shortened due to the skip operator, and the absence of matrix multiplication will yield more tractable gradient flow.

To enable the intra-neuron interaction, we stack multiple layers of RhyRNNs and apply a matrix multiplication $\mathbf{W}_l$ between layers to aggregate the global information. Specifically, assuming $\mathbf{h}_{t,l}$ to be the input to RhyRNN ($\mathbf{h}_{t,0} = \mathbf{x}_t$) at layer $l$ and time $t$, we have:

$$\mathbf{h}_{t,l} = \sigma(\mathbf{W}_l \mathbf{h}_{t,l-1} + \mathbf{u}_l \odot \mathbf{h}_{t-1,l} + \mathbf{b}_l) \tag{6}$$

**Skip regularization** To limit the computational budget, we introduce a regularization term that controls the frequency of *UPDATE*s similar to [5]. This

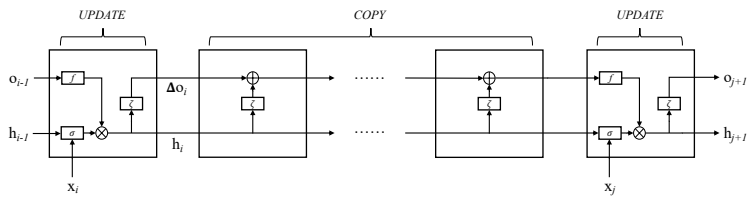


Fig. 2: A basic unit in RhyRNN, where *UPDATE* is emitted at time $i$ and $j$ and all the resting operations in between are *COPY*s. $\otimes$ and $\oplus$ correspond to element-wise product and plus, respectively. A sequence can be divided into several consecutive basic units. Then the gradient back-propagated can be written as the product of multiple gradients of such units. Zoom in for better view.

term is written as:

$$\mathcal{L} = \lambda \sum_{t,k} \mathbf{s}_{t,k} \tag{7}$$

where $\mathbf{s}_{t,k}$ refers to the $k$th binary neuron decision (on *COPY* or *UPDATE*, see Eq. (2)). In general, this term sums up the number of *UPDATE*s on all neurons at every time step.

**Gradient analysis** We employ a strategy to approximate the gradient of the step function $f_{binarize}$ as in [5]:

$$\partial f_{binaries}(x)/\partial x = 1 \tag{8}$$

In other words, Eq. (2) and Eq. (8) are implemented in forward-pass and backward-pass for the network, respectively. Following such a setting, the gradient during the backward pass by taking an example is shown in Fig. 2. In all the following analyses, we discard the bias $\mathbf{b}$ and $\mathbf{b}_p$ for simplicity. In Fig. 2, we analyze the gradient behavior of a sequence segment where only at time stamps $i$ and $j$ are *COPY*s and all the resting time stamps in between are *UPDATE*s. This segment can be viewed as a basic unit since any forward pass of RhyRNN can be separated into such segments (with varying numbers of *COPY*s). Since all operations between $i$ and $j$ are *COPY*s, the hidden state $\mathbf{h}_i$ will directly pass until $j$, thus in the forward pass we have:

$$\mathbf{h}_{j+1} = \sigma(\mathbf{W}\mathbf{x}_j + \mathbf{u} \odot \mathbf{h}_i) \tag{9}$$

In Eq. (9) we omit a term $f_{binarize}(\mathbf{o}_j)$ since it equals 1 (see Fig. 2 at time $j$). However, this term will participate in the backward pass according to the gradient defined in Eq. (8). We expand Eq. (9) as follows:

$$\mathbf{h}_{j+1} = \underbrace{f_{binarize}\left(\sum^{j-i} \zeta\left(\mathbf{w}_p \odot \mathbf{h}_i\right)\right)}_{=\mathbf{1}, \text{for the basic unit}} \odot \sigma(\mathbf{W}\mathbf{x}_j + \mathbf{u} \odot \mathbf{h}_i) \tag{10}$$

Given Eq. (10) and after a series of mathematical manipulations, we can obtain the gradient at time $i$ by taking into account Eq. (8):

$$\nabla J_i = \frac{\partial \mathbf{h}_{j+1}}{\partial \mathbf{h}_i} = \underbrace{\mathbf{u} \odot \sigma' + \sigma \odot \sum^{j-i} \mathbf{w}_p \odot \mathbf{w}_p \odot (\mathbf{1} - \mathbf{w}_p \odot \mathbf{h}_i) \odot \mathbf{h}_i}_{} \tag{11}$$

where the term within the underbrace is the basic unit for any such segment and $\sigma'$ is the gradient of function $\sigma$. Thus, one can calculate the gradient at any time $k$ (where at $k$ there is an *UPDATE* emitted) by calculating the element-wise product:

$$\left.\frac{\partial J}{\partial \mathbf{h}_k}\right|_{k=UPDATE} = \underline{\prod}_{l=UPDATE, l>k} \nabla J_l \tag{12}$$

where $\underline{\prod}$ is the element-wise product. We note two facts involved in this gradient chain rule: 1) there is only scalar multiplication involved in the unit (and element-wise product of multiple such units) which is more tractable than matrix multiplication; 2) hidden states $\mathbf{h}_i$s directly participate in the back-propagation, which can correlate and thus stabilize the gradient from vanishing/exploding as discussed in Multiplicative Integration [60]. In this sense, RhyRNN has a gradient behavior benefiting from both IndRNN and Multiplicative Integration. Readers are referred to [35] and [60] for more details on related analysis.

### 3.3 DivPool

Though the proposed RhyRNN can capture the context at different scales to some extent, it still cannot fully utilize the intrinsically hierarchical context of long videos. In this section, we propose a temporal pooling strategy that explicitly selects most contributing hidden states within a sequence.

The pooling stage is essential in CNNs, which aggregates low-level representations into high-level ones. A series of works also focused on temporal pooling where the objective is to hierarchically shorten and abstract the temporal representations [18, 15, 62, 36]. In this paper, we propose a simple yet efficient method termed as diversity-driven sequential pooling (**DivPool**) by mostly diversifying the capacity of the pooled representations. Our method is based on the observation that, since a video is always highly redundant, an effective pooled representation should ignore the slight difference across frames and concentrate on the most significant changes. Thus, our pooling method performs selection to maximally diversify the hidden states (features). To this end, we first calculate the dissimilarity by cosine distance between $\mathbf{h}_t$ and its previous state $\mathbf{h}_{t-1}$:

$$a_t = 1 - \frac{\mathbf{h}_t \mathbf{h}_{t-1}^T}{\|\mathbf{h}_t\| \|\mathbf{h}_{t-1}\| + \epsilon} \tag{13}$$

where $\epsilon > 0$. Then we sort all $a_t$s in descending order and select the $\alpha\%$ most dissimilar states as the pooled features. Note that this procedure works in an incremental fashion and thus a pairwise distance calculation on all states is not necessary, yielding high efficiency in implementation.

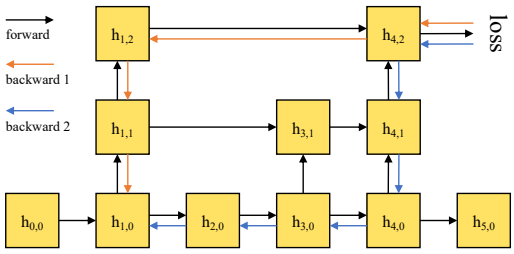


Fig. 3: Schematic diagram of hierarchical architecture with DivPool. The black arrow indicates the flow in forward pass. Orange and blue arrows correspond to the shortest and the longest backward path from state $\mathbf{h}_{1,0}$ to the loss during backward pass, respectively. We note "backward 1" path is shorter than "backward 2" path due to this hierarchy.

The DivPool layer has no learning parameters and thus is similar to max-pooling or average-pooling in CNNs. Yet it differs from max/average-pooling since it performs pooling globally on all features. Besides, it generates the pooling cue in an incremental fashion which is adopted in some effective sequential pooling strategies [18, 15]. The only overhead of performing DivPool is on sorting $a_t$, which is typically $\mathcal{O}(n \log n)$ and can be efficient in practice.

**Back-propagation** DivPool dynamically generates links across RhyRNN layers in the computational graph. The generated links will be effective during a forward-backward round for the network computation. The back-propagated gradients will only follow the general RNN's update path together with the current effective links. Fig. 3 schematically shows an example. We see from Fig. 3 that DivPool can greatly shorten the shortest computational path, which is dominant (compared to other longer paths) in propagating gradients to avoid the vanishing issue. For example, assuming the pooled ratio is 0.5 with $q$ RhyRNN layers and the sequence length is $n$, the length of the shortest path becomes $\mathcal{O}(q + \log(n))$.

### 3.4 Bilinear reweighting for recognition

With DivPool, the redundancy and the complexity of the input video sequence has been reduced. In the following stage, to better incorporate the dependency of the long-range selected hidden states, we design the bilinear reweighting (BR) mechanism to capture the temporal relation among the pooled hidden states.

In our model, we employ a simplified bilinear reweighting (BR) strategy to learn and enhance the temporal correlation of patterns within the pooled hidden states (features). The BR module is applied to the output sequence of DivPool and to embed the hidden states to a new feature re-weighted by the pairwise affinity scores. BR module is inspired by bilinear attention [29] but follows the metric properties. Assuming that the selected features from DivPool form a

feature matrix $\mathbf{V}$, BR rule can be written as:

$$\begin{aligned} \mathbf{S} &= \mathbf{V}^\top \text{norm}(\mathbf{M})\mathbf{V} \\ \hat{\mathbf{V}} &= \mathbf{V} \circ \text{softmax}\left(\text{proj}\left(\text{norm}(\mathbf{S})\right)\right) \end{aligned} \tag{14}$$

where $\mathbf{M} = \mathbf{P}\mathbf{P}^\top$ is a symmetric semi-definite matrix and $\mathbf{P}$ is the parameter to be learnt. This decomposition is to reduce the number of weights to be learnt. The output $\hat{\mathbf{V}}$ is the reweighted feature matrix. norm$(\cdot)$ performs column-wise $L^2$-normalization and proj$(\cdot)$ projects a square matrix into a vector by summarizing the elements per-column. norm$(\cdot)$ performs twice to avoid the magnitude of the final affinity being too large (which may result in almost a one-hot reweighting vector). Note BR (Eq. (14)) differs from Bilinear Pooling in [29] by introducing a column-wise *projection* operator. In this sense, only the magnitude of the input is adjusted, rather than replacing the input by a sum of all other inputs. The intuition is that, since the input $\mathbf{V}$ carries temporal information, a summing schema may violate or mix up this intrinsic (e.g. ordering information).

The output sequence $\hat{\mathbf{V}}$ of BR module is then fed to a standard GRU [8]. We utilize the output of the GRU module at the final time step as the video-level feature and append two fully connected layers following GRU to conduct recognition for different datasets.

## 4 Experiment

### 4.1 Datasets and Reference Methods

We conducted experiments on Breakfast [31], VIRAT 2.0 surveillance video [37] and Charades [45]. All the experiments were done on a computer equipped with a *single* GTX Titan Xp GPU with 12GB memory.

**Breakfast dataset** [31] comprises of 10 breakfast preparation related events that are performed by 52 different individuals in 18 different kitchen scenes. The total number of video clips is 1989. The overall video duration is about 77 hours and the average length of each video is about 140 seconds. Events in the Breakfast dataset are very complicated since each event contains several sub-activities, indicating much higher intra-class variations. We split the dataset into training and testing by following the "s1" split [31].

**VIRAT 2.0 surveillance video dataset** [37] includes about 8 hours of high-resolution surveillance videos (i.e. 1080p or 720p) with 12 kinds of events from 11 different scenes. In our experiment, we only focus on 6 types of person-vehicle interaction events that occur on the parking lot scene. The input video sequence only contains the event area that is cropped based on the ground truth bounding box. The training and testing video samples are randomly selected by following the ratio of 7:3. As such, we conduct the training multiple rounds and report the average performance.

**Charades dataset** [45] is a multi-label action video benchmark with 157 classes in total.Each video is around 30 seconds and contains 6 singleton actions

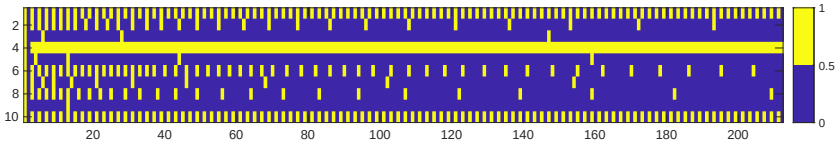


Fig. 4: Visualization of "Skip" operation on the first 10 channels/neurons (out of 256) at the second layer of RhyRNN on a video with 212 frames from Breakfast dataset. Yellow and Blue bars correspond to "*UPDATE*" and "*COPY*" operations, respectively. Vertical and horizontal axes refer to channel and frame, respectively. 10 neurons perform Skip with almost different rhythm to each other.

on average. We follow the same training/testing split in [23] which contains 7.8k and 1.8k videos in each. We report the mean average precision (mAP) on two challenging tasks: multi-label action recognition and temporal localization.

**Reference methods** We employed several existing algorithms for comparison. **C3D** [49], **TSN** [55] and **TRN** [67] are implemented in a simple version with only spatial (RGB) features (without optical-flow). **Two-stream** [46] and **Temporal Fields** [44] utilize both RGB and optical flow features. **IDT** [53] alters to employ action trajectories. For **C3D**, we train it from scratch on all datasets and preprocess frames by following the [49]. For TSN, the frame feature is extracted from a pre-trained VGG16 which won't be fine-tuned in the training stage. Only the segmental consensus part of TSN are trained. We also compare plain **IndRNN** [35] for the Breakfast dataset by stacking 6 layers of IndRNN cells and setting the dropout ratio to 0.5 for each other layer. **3D-ResNet** [20] is employed as both a peer method and a backbone. Timeception [23] (**TC**) is compared since the authors claimed that Timeception is a strong baseline for complex video and can capture long range dependency. (Supervised) **SuperEvents** [39] and (weakly-supervised) **ActGraph** [40] are selected for comparison on Charades. For the tests of the methods with CNN backbone on Breakfast and Charades, we employ the same frame sampling procedure as in [23].

### 4.2 Implementation Details

We employ ResNet101 [21] pre-trained on ImageNet and I3D [6] pre-trained on Kinetics400 to extract features for all frame-wise based algorithms. For event recognition and multi-label action recognition/localization tasks, we employ *Cross-Entropy* and *Binary Cross-Entropy* as loss functions, respectively. For the proposed method, either ResNet101 or I3D backbone is NOT fine-tuned on three selected datasets during the training stage due to GPU resource limitation, different from some prior works [23, 56, 13, 59] which update the CNN backbones. The features are obtained from the last pooling layer of ResNet101 and I3D, yielding 2048-d and 1024-d, respectively. The output (as well as all hidden state) dimension of RhyRNN is 256 and the dimension of the output of BR is 128. Furthermore, there are two fully connected layers with 100 and the number of classes (e.g. 10 for event recognition on Breakfast) neurons following BR.

For 3D-ResNet50 and Timeception [23] models, we extract the 3D video segments with size of $1 \times 7 \times 7$ ($time \times height \times width$) from a 3D ResNet-50 model which is pre-trained on Kinetics-400 [28]. We follow the settings in [23] and collect 64 uniformly sampled video segments, while each segment contains 8 successive frames.

All the proposed and baseline algorithms are implemented with PyTorch [38] toolbox. We train our model with 100 epochs and use Adam optimizer with the learning rate $1e-5$. The pooling ratio for DivPool is set to 25% and the control parameter $\lambda$ for the skip regularization is set to $1e-7$ empirically. For the Breakfast dataset, the training samples are subsampled every 5 frames. The first and last 10 frames are removed from the training samples since those frames are mostly redundant.

### 4.3 Breakfast dataset

Experimental results (of event-level recognition and multi-label activity recognition) on this dataset are summarized in Tab. 1a[1]. In general, our method (full setting) outperformed all the other peer methods in event-level recognition, with competitive performance against state-of-the-art on multi-label recognition. We also see the proposed RhyRNN (2-layer) has better performance compared to IndRNN (6-layer) or SkipRNN without introducing any other modules.

We further evaluate the behavior of our model with multiple RhyRNN + DivPool settings (without BR) on Breakfast, as shown in Tab. 1b. Specifically, "4 RhyRNN + 1 DivPool" and "4 RhyRNN + 2 DivPool" settings are added, referring to the structure {4×RhyRNN+DivPool} and {2×RhyRNN+DivPool+ 2 × RhyRNN + DivPool}, respectively. We can conclude that 2-layer RhyRNN (standard setting in all tests) has slightly better performance than other two.

**Independent Skip strategy** To investigate the effectiveness of "Skip" operations in RhyRNN, we visualize the Skip operations (in Fig. 4) of the first 10 neurons of the weights in the second RhyRNN layer on a breakfast video clip (with length 212) in the testing stage. It is seen that almost every neuron (each row) indeed holds a distinct and independent Skip rhythm. While some neurons emit *UPDATE*s with high frequency to capture the context in high temporal resolution (e.g., neuron 1, 4 and 10), other neurons learn lazier strategies.

### 4.4 VIRAT 2.0 dataset

The performance on this dataset is shown in Tab. 2. Specifically, we test the capacity of algorithms under *varying sampling rhythm* compared to training rhythm. As shown in Tab. 2, "original" indicates sampling each frame (and feature) with the same sampling rhythm at the training stage. The other three

[1] We re-implemented the method TC [23] following the same setting but did not obtain the performance reported in their original paper. Since there is a large gap between our implementation and their results, we report the best performance of [23] in our implementation.

Table 1: Results on Breakfast dataset on (a) event recognition (in Acc) and multi-label classification (in mAP), (b) different settings of RhyRNNs. "RhyRNN(2-layer)" is a model stacked with 2 layers of RhyRNNs concatenated with 2 fully connected layers. Blue color corresponds to singleton RNNs.

(a) Event recognition

| Method | Feature | Acc (%) | mAP (%) |
|---|---|---|---|
| TSN [55] | 2D | 14.3 | - |
| LRCN [10] | 2D | 13.3 | - |
| C3D [49] | 3D | 14.6 | - |
| IndRNN(6-layer) [35] | 2D | 19.4 | 14.1 |
| SkipRNN [5] | 2D | 31.9 | 28.7 |
| IndRNN(+DivPool+BR) | 2D | 42.7 | 40.8 |
| SkipRNN(+DivPool+BR) | 2D | 40.2 | - |
| 3D-Res50 [20] | 3D | 23.7 | - |
| 3D-Res50+TC [23] | 3D | 40.3 | 41.2 |
| RhyRNN(2-layer) | 2D | 35.8 | 30.5 |
| RhyRNN(+DivPool+BR) | 2D | **44.3** | **41.9** |

(b) Different settings

| Setting | 2 RhyRNN + 1 DivPool | 4 RhyRNN + 1 DivPool | 4 RhyRNN + 2 DivPool |
|---|---|---|---|
| Acc | 43.7 | 43.1 | 41.4 |

Table 2: Results on VIRAT 2.0 dataset. The performance of our method is under full setting (RhyRNN+DivPool+BR).

| Method | original | S1 | S2 | S3 |
|---|---|---|---|---|
| C3D [49] | 42.9 | 40.2 | 37.7 | 41.1 |
| TSN [55] | 52.4 | 52.1 | 51.6 | 51.9 |
| IndRNN (6-layer) [35] | 77.6 | 78.3 | 78.2 | 78.0 |
| IndRNN (+DivPool+BR) | 79.0 | 77.4 | 78.2 | 78.2 |
| Ours (full setting) | **81.9** | **81.5** | **81.7** | **80.4** |

scenarios are designed with different combinations of sampling rates. To make the problem more challenging, we first equally divide each testing video sequence into three intervals and apply different sampling rates to each interval to form a new testing sequence. For scenario one (S1), we subsample the first and the third intervals with every 2 and 5 frames respectively, while keeping the rhythm intact for the middle interval. In scenario two (S2), we subsample the first and third intervals every 5 and 2 frames, respectively (reverse of S1). For the last scenario (S3), we randomly sample out a half length of the testing frames. Since the randomness of the last scenario brings uncertainty, we test the well-trained model 5 times and report the average performance of this scenario. We see that our model is quite stable under varying sampling rhythm.

### 4.5 Charades dataset

For the Charades dataset, we employ I3D [6] with 1024-D output feature pre-trained on Kinetics-400 **without** inheriting any frame-level knowledge from or

Table 3: Result on Charades of multi-label activity (MLA) (a) recognition and (b) localization. For (a), "w/o BR" and "w/ BR" refer to the settings removing and keeping BR, respectively. For (b), "S" and "W" refer to "supervised" and "weakly supervised", respectively. *IndRNN here indicates IndRNN+DivPool+BR. †This result is quoted from original Skip-RNN paper [5] where mAP is calculated per 100 frames instead of 25 frames. (TS: two-stream; TF: temporal fields)

(a) MLA recognition

| Method | Modality | mAP(%) |
|---|---|---|
| C3D[49] | RGB | 10.9 |
| TS[46] | RGB+Flow | 18.6 |
| TS+LSTM[46] | RGB+Flow | 17.8 |
| IndRNN*[35] | RGB | 21.1 |
| IDT[53] | RGB+Flow | 17.2 |
| TF[44] | RGB+Flow | 22.4 |
| TRN[67] | RGB | 25.2 |
| Ours(w/o BR) | RGB | 24.6 |
| Ours(w/ BR) | RGB | **25.4** |

(b) MLA localization

| Model | Training | mAP(%) |
|---|---|---|
| LSTM[39] | S | 10.4 |
| Skip-RNN[5]† | S | 8.94 |
| TS+LSTM[39] | S | 18.2 |
| SuperEvents[39] | S | **19.4** |
| TF[44] | S | 12.8 |
| I3D[6] | S | 17.2 |
| ActGraph[40] | W | 15.8 |
| Ours | W | **17.6** |

fine-tuning on the Charades dataset [6, 56, 23]. The frame stride is set to 8 for the I3D model and the size of the feature matrix for each video clip equals to Timelength $\times$ 1024 where the TimeLength = FrameLength/8. We test two challenging tasks: multi-label (MLA) recognition and temporal localization.

Tab. 3a shows the MLA recognition performance of different algorithms on the Charades dataset. And the results demonstrate that our algorithm has a competitive capacity compared to the state-of-the-art on capturing the temporal information of sub-actions in the complex videos.

Tab. 3b summarizes the results of temporal localization. To this end, we inherit the model parameters pre-trained on MLA recognition task but removing the DivPool module[2]. We then fine-tune the last 2 fully-connected layers with BCE loss on MLA recognition task (only for the last time stamp) for 5 epochs. In the testing stage we pass the output of the RhyRNN at *each* time stamp through the last 2 fully-connected layers to produce score of action classes for each frame. Action class with the highest score is regarded as the predicted label for the current frame. Since during the training stage no frame-level label is provided, our model is trained in a **weakly supervised** fashion, which is more challenging than fully supervised localization task (e.g. [39, 44]). Surprisingly, we see that our model outperforms several fully supervised counterparts and a very recent weakly-supervised method ActGraph [40]. This observation supports our claim that RhyRNN is capable of capturing temporal context at multiple levels.

[2] DivPool cannot work frame-wise since it selects only a portion of time stamps. Therefore we remove DivPool from the full setting.

Table 4: (a) Ablation study on Breakfast dataset. In the full setting, we stack 2 layers of RhyRNNs followed by DivPool and BR. (b) Model size comparison. Note: there are two layers of RhyRNNs concatenated together with both hidden state 256-D.

(a) Ablation

| RhyRNN | DivPool | BR | Acc | mAP |
|---|---|---|---|---|
| √ | √ | √ | 44.3 | 41.9 |
| √ | √ | × | 43.7 | 40.5 |
| √ | × | √ | 40.7 | 35.2 |
| √ | × | × | 35.8 | - |

(b) Model size

| Method | Model size |
|---|---|
| TC [23] | 15.8MB |
| Ours (Full setting) | 23.2MB |
| Ours (RhyRNN+DivPool) | 20.4MB |

### 4.6 Ablation analysis and model size

Ablation analysis was conducted on the Breakfast dataset following the settings in Sec. 4.3. Results are summarized in Tab. 4a. By turning off DivPool, we simply fed all the output states of RhyRNN to BR. On the other hand, we removed BR and fed the pooled states to a naive GRU once turning off BR. We also tested the capacity of the RhyRNN module in capturing complex temporal information by turning off both DivPool and BR modules. We see that the full setting (RhyRNN+DivPool+BR) delivers the best performance among all. And the DivPool module plays a more important role in understanding the long and complex videos, compared with the BR module. Moreover, the RhyRNN module itself is able to acquire information from long and complex videos compared with some conventional algorithms presented in the Tab 1a. In general, the proposed modules in our framework consistently enhance the performance.

Tab. 4b summarizes the models sizes on different setting under event-level recognition on the Breakfast dataset. We note the size of our model with or without BR is around 20MB, comparative to the size of state-of-the-art method TC [23], which claimed to be capable of reducing the model size significantly.

## 5 Conclusion

In this paper, we study the task of recognizing events in long and complex videos. Since there is critical distinction between traditional action recognition based on short clips and event recognition using long videos, simply adapting the methods for the former to the latter is ineffective. To address this, we designed an end-to-end RNN framework taking into account the latent context at multiple levels. Especially, three novel and essential parts were proposed: RhyRNN, DivPool and BR. By taking advantage of each, our model can capture video context at different scales in an adaptive and hierarchical fashion. We investigated the property of the proposed model and demonstrated its superiority through extensive experiments even without the need of fine-tuning the feature extraction backbones.

## References


1. Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., Parikh, D.: Vqa: Visual question answering. In: ICCV (2015)
2. Arjovsky, M., Shah, A., Bengio, Y.: Unitary evolution recurrent neural networks. In: ICML (2016)
3. Bilen, H., Fernando, B., Gavves, E., Vedaldi, A.: Action recognition with dynamic image networks. PAMI **40**(12), 2799–2813 (2017)
4. Bilen, H., Fernando, B., Gavves, E., Vedaldi, A., Gould, S.: Dynamic image networks for action recognition. In: CVPR (2016)
5. Campos, V., Jou, B., Giró-i Nieto, X., Torres, J., Chang, S.F.: Skip rnn: Learning to skip state updates in recurrent neural networks. arXiv preprint arXiv:1708.06834 (2017)
6. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: CVPR (2017)
7. Chao, Y.W., Vijayanarasimhan, S., Seybold, B., Ross, D.A., Deng, J., Sukthankar, R.: Rethinking the faster r-cnn architecture for temporal action localization. In: CVPR (2018)
8. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
9. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: CVPR (2017)
10. Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: CVPR (2015)
11. Du, Y., Wang, W., Wang, L.: Hierarchical recurrent neural network for skeleton based action recognition. In: CVPR (2015)
12. Duan, L., Xu, D., Tsang, I.W.H., Luo, J.: Visual event recognition in videos by learning from web data. PAMI **34**(9), 1667–1680 (2012)
13. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: ICCV (2019)
14. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: CVPR (2016)
15. Fernando, B., Gavves, E., Oramas, J., Ghodrati, A., Tuytelaars, T.: Rank pooling for action recognition. PAMI **39**(4), 773–787 (2016)
16. Fernando, B., Tan, C., Bilen, H.: Weakly supervised gaussian networks for action detection. In: WACV (2020)
17. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with lstm (1999)
18. Girdhar, R., Ramanan, D.: Attentional pooling for action recognition. In: NIPS (2017)
19. Gong, B., Chao, W.L., Grauman, K., Sha, F.: Diverse sequential subset selection for supervised video summarization. In: NIPS (2014)
20. Hara, K., Kataoka, H., Satoh, Y.: Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In: CVPR (2018)
21. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
22. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

23. Hussein, N., Gavves, E., Smeulders, A.W.: Timeception for complex action recognition. In: CVPR (2019)
24. Hussein, N., Gavves, E., Smeulders, A.W.: Videograph: Recognizing minutes-long human activities in videos. In: ICCVW (2019)
25. Jiang, Y.G., Bhattacharya, S., Chang, S.F., Shah, M.: High-level event recognition in unconstrained videos. International journal of multimedia information retrieval **2**(2), 73–101 (2013)
26. Kanuparthi, B., Arpit, D., Kerg, G., Ke, N.R., Mitliagkas, I., Bengio, Y.: h-detach: Modifying the LSTM gradient towards better optimization. In: ICLR (2019)
27. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: CVPR (2014)
28. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, A., Suleyman, M., Zisserman, A.: The kinetics human action video dataset. ArXiv **abs/1705.06950** (2017)
29. Kim, J.H., Jun, J., Zhang, B.T.: Bilinear attention networks. In: NIPS (2018)
30. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: a large video database for human motion recognition. In: ICCV (2011)
31. Kuehne, H., Arslan, A., Serre, T.: The language of actions: Recovering the syntax and semantics of goal-directed human activities. In: CVPR (2014)
32. Kuehne, H., Gall, J., Serre, T.: An end-to-end generative framework for video segmentation and recognition. In: WACV (2016)
33. Lan, T., Zhu, Y., Roshan Zamir, A., Savarese, S.: Action recognition by hierarchical mid-level action elements. In: ICCV (2015)
34. Lee, Y.J., Ghosh, J., Grauman, K.: Discovering important people and objects for egocentric video summarization. In: CVPR (2012)
35. Li, S., Li, W., Cook, C., Zhu, C., Gao, Y.: Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In: CVPR (2018)
36. Nguyen, P., Liu, T., Prasad, G., Han, B.: Weakly supervised action localization by sparse temporal pooling network. In: CVPR (2018)
37. Oh, S., Hoogs, A., Perera, A., Cuntoor, N., Chen, C., Lee, J.T., Mukherjee, S., Aggarwal, J.K., Lee, H., Davis, L., Swears, E., Wang, X., Ji, Q., Reddy, K., Shah, M., Vondrick, C., Pirsiavash, H., Ramanan, D., Yuen, J., Torralba, A., Song, B., Fong, A., Roy-Chowdhury, A., Desai, M.: A large-scale benchmark dataset for event recognition in surveillance video. In: CVPR (2011)
38. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
39. Piergiovanni, A., Ryoo, M.S.: Learning latent super-events to detect multiple activities in videos. In: CVPR (2018)
40. Rashid, M., Kjellström, H., Lee, Y.J.: Action graphs: Weakly-supervised action localization with graph convolution networks. arXiv preprint arXiv:2002.01449 (2020)
41. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local svm approach. In: ICPR (2004)
42. Sharma, S., Kiros, R., Salakhutdinov, R.: Action recognition using visual attention. In: NIPS Time Series Workshop (2015)
43. Shou, Z., Wang, D., Chang, S.F.: Temporal action localization in untrimmed videos via multi-stage cnns. In: CVPR (2016)
44. Sigurdsson, G.A., Divvala, S., Farhadi, A., Gupta, A.: Asynchronous temporal fields for action recognition. In: CVPR (2017)

45. Sigurdsson, G.A., Varol, G., Wang, X., Farhadi, A., Laptev, I., Gupta, A.: Hollywood in homes: Crowdsourcing data collection for activity understanding. In: ECCV (2016)
46. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: NIPS (2014)
47. Soomro, K., Zamir, A.R., Shah, M., Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. CoRR (2012)
48. Tapaswi, M., Zhu, Y., Stiefelhagen, R., Torralba, A., Urtasun, R., Fidler, S.: Movieqa: Understanding stories in movies through question-answering. In: CVPR (2016)
49. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: ICCV (2015)
50. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: CVPR (2018)
51. Tran, S.D., Davis, L.S.: Event modeling and recognition using markov logic networks. In: ECCV (2008)
52. Veeriah, V., Zhuang, N., Qi, G.J.: Differential recurrent neural networks for action recognition. In: ICCV (2015)
53. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: ICCV (2013)
54. Wang, L., Qiao, Y., Tang, X.: Action recognition with trajectory-pooled deep-convolutional descriptors. In: CVPR (2015)
55. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: ECCV (2016)
56. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: CVPR (2018)
57. Wang, X., Ji, Q.: Hierarchical context modeling for video event recognition. PAMI **39**(9), 1770–1782 (2017)
58. Wang, Y., Wang, S., Tang, J., O'Hare, N., Chang, Y., Li, B.: Hierarchical attention network for action recognition in videos. arXiv preprint arXiv:1607.06416 (2016)
59. Wu, C.Y., Feichtenhofer, C., Fan, H., He, K., Krahenbuhl, P., Girshick, R.: Long-term feature banks for detailed video understanding. In: CVPR (2019)
60. Wu, Y., Zhang, S., Zhang, Y., Bengio, Y., Salakhutdinov, R.R.: On multiplicative integration with recurrent neural networks. In: NIPS (2016)
61. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In: ECCV (2018)
62. Xu, S., Cheng, Y., Gu, K., Yang, Y., Chang, S., Zhou, P.: Jointly attentive spatial-temporal pooling networks for video-based person re-identification. In: ICCV (2017)
63. Xu, Y., Zhang, C., Cheng, Z., Xie, J., Niu, Y., Pu, S., Wu, F.: Segregated temporal assembly recurrent networks for weakly supervised multiple action detection. In: AAAI (2019)
64. Xu, Z., Yang, Y., Hauptmann, A.G.: A discriminative cnn video representation for event detection. In: CVPR (2015)
65. Yeung, S., Russakovsky, O., Jin, N., Andriluka, M., Mori, G., Fei-Fei, L.: Every moment counts: Dense detailed labeling of actions in complex videos. IJCV **126**(2-4), 375–389 (2018)
66. Zhang, K., Chao, W.L., Sha, F., Grauman, K.: Video summarization with long short-term memory. In: ECCV (2016)

67. Zhou, B., Andonian, A., Oliva, A., Torralba, A.: Temporal relational reasoning in videos. In: ECCV (2018)