Bidirectional Multirate Reconstruction for Temporal Modeling in Videos

Linchao Zhu Zhongwen Xu Yi Yang
CAI, University of Technology Sydney
{zhulinchao7, zhongwen.s.xu, yee.i.yang}@gmail.com

Abstract

Despite the recent success of neural networks in image feature learning, a major problem in the video domain is the lack of sufficient labeled data for learning to model temporal information. In this paper, we propose an unsupervised temporal modeling method that learns from untrimmed videos. The speed of motion varies constantly, e.g.*, a man may run quickly or slowly. We therefore train a Multirate Visual Recurrent Model (MVRM) by encoding frames of a clip with different intervals. This learning process makes the learned model more capable of dealing with motion speed variance. Given a clip sampled from a video, we use its past and future neighboring clips as the temporal context, and reconstruct the two temporal transitions,* i.e.*, present→past transition and present→future transition, reflecting the temporal information in different views. The proposed method exploits the two transitions simultaneously by incorporating a bidirectional reconstruction which consists of a backward reconstruction and a forward reconstruction. We apply the proposed method to two challenging video tasks,* i.e.*, complex event detection and video captioning, in which it achieves state-of-the-art performance. Notably, our method generates the best single feature for event detection with a relative improvement of 10.4% on the MEDTest-13 dataset and achieves the best performance in video captioning across all evaluation metrics on the YouTube2Text dataset.*

1. Introduction

Temporal information plays a key role in video representation modeling. In earlier years, hand-crafted features, *e.g.*, Dense Trajectories (DT) and improved Dense Trajectories (iDT) [46, 47], use local descriptors along trajectories to model video motion structures. Despite achieving promising performance, DT and iDT are very expensive to extract, due to the heavy computational cost of optical flows and it takes about a week to extract iDT features for 8,000 hours of web videos using 1,000 CPU cores [49]. Deep visual features have recently achieved significantly

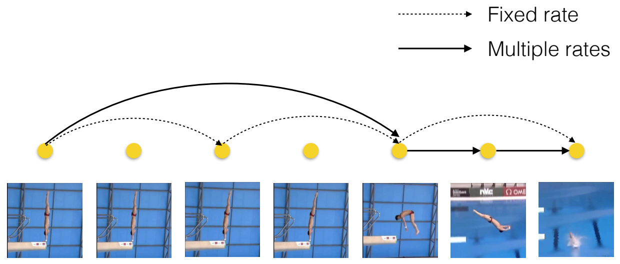

Figure 1. Frame sampling rate should vary in accordance with different motion speed. In this example, only the last three frames have fast motion. The dashed arrow corresponds to a fixed sampling rate, while the solid arrow corresponds to multiple rates.

better performance in image classification and detection tasks than hand-crafted features at an efficient processing speed [23, 14, 12]. However, learning a video representation on top of deep Convolutional Neural Networks (ConveNets) remains a challenging problem. Two-stream ConvNet [36] is groundbreaking in learning video motion structures over short video clips. Although it achieves comparable performance to iDT for temporally trimmed videos, two-stream ConvNet still needs to extract optical flows. The heavy cost severely limits the utility of methods based on optical flows, especially in the case of large scale video data.

Extending 2D ConvNet to 3D, C3D ConvNet has been demonstrated to be effective for spatio-temporal modeling and it avoids extracting optical flows. However, it can only model temporal information in short videos, usually of 16 frames [42]. Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) [16, 27] and a modified Hierarchical Recurrent Neural Encoder (HRNE) [28], have been used to model temporal information in videos. A major limitation of [27] and [28] is that the input frames are encoded with a fixed sampling rate when training the RNNs. On the other hand, the motion speed of videos varies even in the same video. As shown in the Figure 1, there is almost no apparent motion in the first four frames, but fast motion is observed in the last three frames. The encoding rate should be correspondingly low for the first four frames, but high for the last three, as indicated by the solid arrow. The fixed rate strategy, however, is redundant for the first four frames, while important information for the last three frames is lost. The gap between the fixed encoding rate and motion speed variance in real world videos may degrade performance, especially when the variance is extensive.

Notwithstanding the appealing ability of end-to-end approaches for learning a discriminative feature, such approaches require a large amount of labeled data to achieve good performance with plausible generalization capabilities. Compared to images, a large number of videos are very expensive to label by humans. For example, the largest public human-labeled video dataset (ActivityNet) [11] only has 20,000 labeled videos while the ImageNet dataset has over one million labeled instances [34]. Temporal ConvNet trained on the UCF-101 dataset [37] with about 10,000 temporally trimmed videos did not generalize well on temporally a untrimmed dataset [50]. Targeting short video clips, Srivastava *et al.* [39] proposed training a composite autoencoder in an unsupervised manner to learn video temporal structures, essentially by predicting future frames and reconstructing present frames. Inspired by a recent study on neuroscience which shows that a common brain network underlies the capacity both to remember the past and imagine the future [35], we consider reconstructing two temporal transitions, *i.e.*, present→past transition and present→future transition. Importantly, video motion speed changes constantly in untrimmed videos and Srivastava *et al.* directly used an LSTM with a single fixed sampling rate, making it vulnerable to motion speed variance.

In this paper, we propose an unsupervised method to learn from untrimmed videos for temporal information modeling without the heavy cost of computing optical flows. It makes the following two major contributions. First, our Multirate Visual Recurrent Model adopts multiple encoding rates, and together with the reading gate and the updating gate in the Gated Recurrent Unit, it enables communication between different encoding rates and collaboratively learns a multirate representation which is robust to motion speed variance in videos. Second, we leverage the mutual benefit of two learning processes by reconstructing the temporal context in two directions. The two learning directions regularize each other, thereby reducing the overfitting problem. The two contributions yield a new video representation, which achieves the best performance in two different tasks. Note that the method proposed in [49] has been demonstrated to be the best single feature for event detection, and our method outperforms this method with a relative improvement of 10.4% and 4.5% on two challenging datasets, *i.e.*, MEDTest-13 and MEDTest-14 respectively. In the video captioning task, our single feature outperforms other state-of-the-art methods across all evaluation metrics, most of which use multiple features. It is worthwhile mentioning that in very rare cases, one method can outperform all others for video captioning over all evaluation metrics. These results demonstrate the effectiveness of the proposed method.

2. Related Work

Research efforts to improve visual representations for videos have been ongoing. Local features such as HOF [25] and MBH [8] extracted along spatio-temporal tracklets have been used as motion descriptors in the Dense Trajectories feature [46] and its variants [47]. However, it is notoriously inefficient to extract hand-crafted features like improved Dense Trajectories (iDT) [47, 49], mostly due to the dense sampling nature of local descriptors and the time-consuming extraction of optical flows. On the other hand, the classification performance of state-of-the-art hand-crafted features has been surpassed by many methods based on neural networks in web video classification and action recognition tasks [49, 48].

Convolutional Networks for video classification. One way to use ConvNets for video classification is to perform temporal pooling over convolutional activations. Ng *et al.* [27] proposed learning a global video representation by using max pooling over the last convolutional layer across video frames. Wang *et al.* [48] aggregated ConvNet features along the tracklets obtained from iDT. Xu *et al.* [49] applied VLAD encoding [18] over ConvNet activations and found that the encoding methods are superior to mean pooling. The other common solution is to feed multiple frames as input to ConvNets. Karpathy *et al.* [19] proposed a convolutional temporal fusion network, but it is only marginally better than the single frame baseline. Tran *et al.* [42] avoided the extraction of optical flows by utilizing 3D ConvNets to model motion information. Simonyan and Zisserman [36] took optical flows as the flow image input to a ConvNet, and this two-stream network has much better performance than the previous networks on action recognition.

Recurrent Networks for video classification. Ng *et al.* [27] and Donahue *et al.* [10] investigated the modeling of temporal structures in videos with Long Short-Term Memory (LSTM) [16]. However, even with five-layer LSTMs, trained on millions of videos, they do not show promising performance compared to ConvNets [27]. Patraucean *et al.* [31] used a spatio-temporal autoencoder to model video sequences through optical flow prediction and reconstruction of the next frame. Ballas *et al.* [4] used a Convolutional Gated Recurrent Unit (ConvGRU) which leverage information from different spatial levels of the activations. Srivastava *et al.* [39] used LSTM to model video sequences in an unsupervised way. In this work, we utilize the RNNs on video representation learning, improving the representation by being aware of the multirate nature of video content. Moreover, the temporal consistency between frames in the neighborhood is incorporated into the networks in an unsupervised way, providing richer training information and creating opportunities to learn from abundant untrimmed videos.

Video captioning. Video captioning has emerged as a popular task in recent years, since it bridges visual understanding and natural language description. Conditioned on the visual context, RNNs produce one word per step to generate captions for videos. Venugopalan *et al.* [44] used a stacked sequence to sequence (seq2seq) [40] model, in which an LSTM is used as a video sequence encoder and the other LSTM serves as a caption decoder. Yao *et al.* [51] incorporated the temporal attention mechanism in the description decoding stage. Pan *et al.* [28] proposed using a hierarchical LSTM to model videos sequences, while Yu *et al.* [52] used a hierarchical GRU network to model the structure of captions. In this work, we demonstrate that the strong video representation learned in our model improves the video captioning task, confirming the generalization ability of our features.

3. Multirate Visual Recurrent Models

In this section, we introduce our approach for video sequence modeling. We first review the structure of Gated Recurrent Unit (GRU) and extend the GRU to a multirate version. The model architecture for unsupervised representation learning is then introduced, which is followed by task specific models for event detection and video captioning. In the model description, we omit all bias terms in order to increase readability.

3.1. Multirate Gated Recurrent Unit

Gated Recurrent Unit. At each step t, a GRU cell takes a frame representation $\mathbf{x}_t$ and previous state $\mathbf{h}_{t-1}$ as inputs and generates a hidden state $\mathbf{h}_t$ and an output $\mathbf{o}_t$ which are calculated by,

$$\begin{aligned}
\mathbf{r}_t &= \sigma(\mathbf{U}_r\mathbf{x}_t + \mathbf{V}_r\mathbf{h}_{t-1}),\\
\mathbf{z}_t &= \sigma(\mathbf{U}_z\mathbf{x}_t + \mathbf{V}_z\mathbf{h}_{t-1}),\\
\bar{\mathbf{h}}_t &= \tanh(\mathbf{U}_{\bar{h}}\mathbf{x}_t + \mathbf{V}_{\bar{h}}(\mathbf{r}_t \odot \mathbf{h}_{t-1})),\\
\mathbf{h}_t &= (1-\mathbf{z}_t)\odot\mathbf{h}_{t-1} + \mathbf{z}_t \odot \bar{\mathbf{h}}_t,\\
\mathbf{o}_t &= \mathbf{W}_o\mathbf{h}_t,
\end{aligned} \tag{1}$$

where $\mathbf{x}_t$ is the input, $\mathbf{r}_t$ is the reset gate, $\mathbf{z}_t$ is the update gate, $\mathbf{h}_t$ is the proposed state, $\bar{\mathbf{h}}_t$ is the internal state, σ is the sigmoid activation function, $\mathbf{U}_*$ and $\mathbf{V}_*$ are weight matrices, and $\odot$ is element-wise multiplication. The output $\mathbf{o}_t$ is calculated by a linear transformation from the state $\mathbf{h}_t$. We denote the whole process as:

$$\mathbf{h}_t, \mathbf{o}_t = \mathrm{GRU}(\mathbf{x}_t, \mathbf{h}_{t-1}), \tag{2}$$

and when it has iterated S steps, we can obtain the state of the last step $\mathbf{h}_S$.

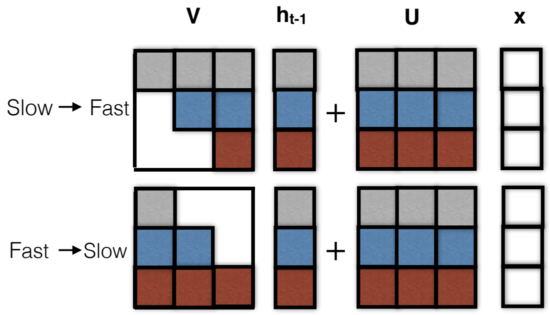

Figure 2. We illustrate the two modes in the mGRU. In the slow to fast mode, the state matrices $\mathbf{V}_*$ are block upper-triangular matrices and in the fast to slow mode, they are block lower-triangular matrices.

Multirate Gated Recurrent Unit (mGRU). Inspired by clockwork RNN [22], we extend the GRU cell to a multirate version. The clockwork RNN uses delayed connections for inputs and inter-connections between steps to capture longer dependencies. Unlike traditional RNNs where all units in the states follow the protocol in Eq. 1, states and weights in the clockwork RNN are divided into groups to model information at different rates. We divide state $\mathbf{h}_t$ into k groups, and each group g_i has a clock period T_i, where $i \in \{1,\ldots,k\}$. T_i can be arbitrary numbers, and we empirically use $k = 3$ and set $T_1, T_2, T_3 = 1, 3, 6$. Faster groups (with smaller T_i) take inputs more frequently than slower groups, and the slower module skips more inputs. Formally, at each step t, matrices of the group satisfying $(t \text{ MOD } T_i) = 0$ are activated and are used to calculate the next state, which is

$$\begin{aligned}
\mathbf{r}^i_t &= \sigma(\mathbf{U}^i_r\mathbf{x}_t + \textstyle\sum_{j=1}^k \mathbf{V}^{i,j}_r\mathbf{h}^j_{t-1}),\\
\mathbf{z}^i_t &= \sigma(\mathbf{U}^i_z\mathbf{x}_t + \textstyle\sum_{j=1}^k \mathbf{V}^{i,j}_z\mathbf{h}^j_{t-1}),\\
\bar{\mathbf{h}}^i_t &= \tanh(\mathbf{U}^i_{\bar{h}}\mathbf{x}_t + \textstyle\sum_{j=1}^k \mathbf{V}^{i,j}_{\bar{h}}(\mathbf{r}^i_t \odot \mathbf{h}^j_{t-1})),\\
\mathbf{h}^i_t &= (1-\mathbf{z}^i_t)\odot\mathbf{h}^i_{t-1} + \mathbf{z}^i_t\odot\bar{\mathbf{h}}^i_t,
\end{aligned} \tag{3}$$

where the state weight matrices $\mathbf{V}_*$ are divided into k block-rows and each block-row is partitioned into k block-columns. $\mathbf{V}^{i,j}_*$ denotes the sub-matrix in block-row i and block-column j. The input weight matrices $\mathbf{U}_*$ are divided k block-rows and $\mathbf{U}^i_*$ denotes the weights in block-row i and

$$\textstyle\sum_{j=1}^k \mathbf{V}^{i,j}_*\mathbf{h}^j_{t-1} = \begin{cases} \sum_{j=1}^i \mathbf{V}^{i,j}_*\mathbf{h}^j_{t-1}, & \text{Fast} \rightarrow \text{slow mode}\\ \sum_{j=i}^k \mathbf{V}^{i,j}_*\mathbf{h}^j_{t-1}, & \text{Slow} \rightarrow \text{fast mode}\end{cases} \tag{4}$$

Two modes can be used for state transition. In the slow to fast mode, states of faster groups consider previous slower states, thus the faster states incorporate information not only at the current speed but also information that is slower and more coarse. The intuition for the fast to slow mode is that when the slow mode is activated, it can take advantage of the information already encoded in the faster states. The two modes are illustrated in Figure 2. Empirically, we use the fast to slow mode in our model as it performed better in the preliminary experiments.

If $(t \texttt{ MOD } T_i) \neq 0$, the previous state is directly passed over to the next state,

$$\mathbf{h}_t^i = \mathbf{h}_{t-1}^i. \tag{5}$$

Figure 3 illustrates the state iteration process. Note that not all previous modules are considered to calculate the next state at each step, thus fewer parameters will be used and the training will be more efficient.

3.2. Unsupervised Video Sequence Reconstruction

Video sequences are highly correlated to their neighboring context clips. We use the idea of context reconstruction for video sequence modeling. The similar methods have been successfully applied for language modeling and other language tasks [26, 21]. In the unsupervised training process, we follow the classic sequence-to-sequence (seq2seq) model [40] where an encoder encodes a sequence of inputs and passes the last state to the decoder for target sequence generation. In our scenario, the mGRU encoder takes frame-level features extracted from the pre-trained convolutional models as inputs and generates the state at each step which will be attended by the decoders. The state of the last step of the encoder is passed to the decoder, *i.e.*, $\mathbf{h}_0^{\text{dec}} = \mathbf{h}_S^{\text{enc}}$. Two decoders are used to predict the context sequences of the inputs, *i.e.*, reconstructing the frame-level representations of the previous sequence and next sequence.

Decoder. We use the seq2seq model with attention mechanism to model video temporal structures via context reconstruction. We denote that $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$ is the previous sequence of input sequence $\mathbf{X}$, and $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)$ is the next sequence. The decoder is a GRU conditioned on the encoder outputs $\mathbf{o}_{1,\ldots,S}^{\text{enc}}$ and the last step state $\mathbf{h}_S^{\text{enc}}$ of the encoder. We use the attention mechanism at each step to help the decoder to decide which frames in the input sequence might be related to the next frame reconstruction. At step t, the decoder ϕ generates the prediction $\mathbf{o}_t^{\text{dec}}$ by calculating,

$$\begin{aligned}
\mathbf{y}_t^{\text{attn}} &= \text{Linear}(\mathbf{y}_t, \mathbf{a}_{t-1}),\\
\mathbf{h}_t^{\text{dec}}, \mathbf{o}_t^{\text{attn}} &= \text{GRU}(\mathbf{y}_t^{\text{attn}}, \mathbf{h}_{t-1}^{\text{dec}}),\\
e_t^i &= \mathbf{v}^{\text{T}}\tanh(\mathbf{W}_{he}\mathbf{h}_t^{\text{dec}} + \mathbf{W}_{oe}\mathbf{o}_i^{\text{enc}}),\\
a_t^i &= \exp(e_t^i) \,/\, \textstyle\sum_{j=1}^{S} \exp(e_t^j),\\
\mathbf{a}_t &= \textstyle\sum_{i=1}^{S} a_t^i \mathbf{o}_i^{\text{enc}},\\
\mathbf{o}_t^{\text{dec}} &= \text{Linear}(\mathbf{o}_t^{\text{attn}}, \mathbf{a}_t),
\end{aligned} \tag{6}$$

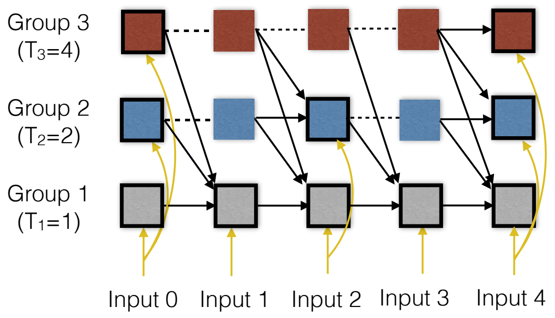

Figure 3. Unrolled mGRU. In the example, the state is divided into three groups and the slow to fast mode is shown. At each step t, groups satisfying $(t \texttt{ MOD } T_i) = 0$ are activated (cells with black border). For example, at step 2, group 1 and group 2 are activated. The activated groups take the frame input and previous states to calculate the next states. For those that are inactivated, we simply pass the previous states to the next step. Group 1 is the fastest and group 3 is the slowest with larger T_i. The slow to fast mode is the mode by which the slower groups pass the states to the faster groups.

where $\text{Linear}(\mathbf{a}, \mathbf{b}) = \mathbf{W}_a\mathbf{a} + \mathbf{W}_b\mathbf{b}$, a_t^i is the normalized attention weight for encoder output $\mathbf{o}_i^{\text{enc}}$ and $\mathbf{a}_t$ is the weighted average of the encoder outputs. We use two decoders that do not share parameters: one for the past sequence reconstruction and the other for the future sequence reconstruction (Figure 4). The decoders are trained to minimize the reconstruction loss of two sequences, which is

$$\begin{aligned}
&\sum_t \ell(\phi(\mathbf{y}_{<t}, \mathbf{o}_{1,\ldots,S}^{\text{enc}}, \mathbf{h}_S^{\text{enc}}; \theta), \mathbf{y}_t) + \\
&\sum_{t'} \ell(\phi(\mathbf{z}_{<t'}, \mathbf{o}_{1,\ldots,S}^{\text{enc}}, \mathbf{h}_S^{\text{enc}}; \theta'), \mathbf{z}_{t'}).
\end{aligned} \tag{7}$$

We choose the Huber loss for regression due to its robustness following Girshick [12],

$$\ell(y, \bar{y}) = \begin{cases} \frac{1}{2}(y - \bar{y})^2 & \text{for } |y - \bar{y}| \leq \delta, \\ \delta\,|y - \bar{y}| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases} \tag{8}$$

We set $\delta = 0.5$ in all experiments.

For the past reconstruction, we reverse the input order as well as the target order to minimize information lag [40]. The two decoders are trained with the encoder via back-propagation, and we regularize the network by randomly dropping one decoder for each batch. As we have two decoders in our model, each decoder will have the probability of being chosen for training of 0.5 (Figure. 4).

During unsupervised training, we uniformly sample video frames and extract frame-level features from convolutional models. We set the sequence length to K, *i.e.*, the encoder takes K frames as inputs, while the decoders reconstruct previous K frames and next K frames. We randomly sample a temporal window of consecutive $3K$ frames (3 segments) during training. If the video length is less than $3K$, we pad zeros for each segment.

3.3. Complex Event Detection

We validate the unsupervised learned features on the task of complex event detection. We choose the TRECVID Multimedia Event Detection (MED) task as it is more dynamic and complex compared to the action recognition task, in which the target action duration is short and usually lasts only seconds. As the features from the unsupervised training are not discriminative, *i.e.*, label information has not been applied during training, we further train the encoder for video classification. We use the mGRU encoder to encode the video frames and take the last hidden state in the encoder for classification. We do not apply losses at each step, *e.g.*, the LSTM model in [27], as the video data in our task is untrimmed, which is more noisy and redundant. We use the network structure of FC(1024)-ReLU-Dropout(0.5)-FC(1024)-ReLU-Dropout(0.5)-FC(*class_num*+1)-Softmax. Since there are background videos which do not belong to any target events, we add another class for these videos.

During supervised training, we first initialize the weights of the encoder with the weights pre-trained via unsupervised context reconstruction. For each batch, instead of uniformly sampling videos within the training set, we keep the ratio of the number of positive and background videos to $1:2$. We bias the mini-batch sampling because of the imbalance between the positive and negative examples.

During inference, the encoder generates multirate states at each step, and there are several ways to pool the states to obtain a global video representation. One simple approach is to average the outputs, and the obtained global video representation is then classified with a Linear SVM. The other way is to encode the outputs with an encoding method. Xu *et al.* [49] found that Vector of Locally Aggregated Descriptors (VLAD) [18] encoding outperforms average pooling and Fisher Vectors [32] over ConvNets activations by a large margin on the MED task. We thus apply the VLAD encoding method to encode the RNN representations.

Given inputs $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ and centers $\mathbf{C} = \{\mathbf{c}_1, \dots, \mathbf{c}_K\}$ which are calculated by the k-means algorithm on sampled inputs, for each $k \in \{1, \dots, K\}$, we have,

$$\mathbf{u}_k = \sum_{i:\text{Nearest}(\mathbf{x}_i)=\mathbf{c}_k} \mathbf{x}_i - \mathbf{c}_k, \tag{9}$$

where $\mathbf{x}_i$ is assigned to the center $\mathbf{c}_k$ if it is the nearest center. Concatenating $\mathbf{u}_k$ over all K centers, we obtain the feature vector of size DK where D is the dimension of $\mathbf{x}_i$. Normalization methods are used to improve the encoding performance. Power normalization, often signed square

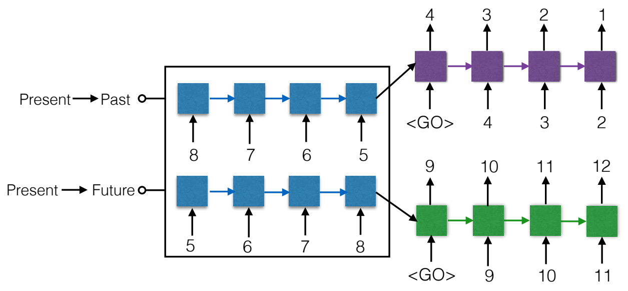

Figure 4. The model architecture of unsupervised video representation learning. In this model, two decoders are used to predict surrounding contexts by reconstructing previous frames and next frame sequences. The "<GO>" input, which is a zero vector, is used at step 0 in the decoder. During training, one of the two decoders is used with a probability of 0.5 for reconstruction.

rooting (SSR), is usually used to convert each element x_i into $\text{sign}(x_i)\sqrt{|x_i|}$. The intra-normalization method normalizes representations for each center, followed by the ℓ_2 normalization for the whole feature vector [32]. The final normalized representation is classified with a Linear SVM.

Note that the states in mGRU are divided into groups, we thus encode the state of the three different scales independently. We combine the three scores by average fusion.

3.4. Video Captioning

We also demonstrate the generalization ability of our proposed video representation on the video captioning task. In video captioning, an encoder is used to encode video representations and a decoder is used to generate video descriptions. We follow the basic captioning decoding process. Given a video sequence $\mathbf{X}$ and a description sequence $Y = \{y_1, \dots, y_N\}$, where each word is represented by a one-hot vector and a one-of-K (K is the vocabulary size) embedding is used in the decoder input to represent a discrete word with a continuous vector, the overall objective is to maximize the log-likelihood of the generated sequence,

$$\max_{\boldsymbol{\theta}} \sum_{t=1}^{N} \log \Pr(y_t | y_{<t}, \mathbf{X}; \boldsymbol{\theta}). \tag{10}$$

Softmax activation is used on the decoder output to obtain the probability of word y_t. The attention mechanism (Eq. 6) is used in both the input and output of the decoder.

4. Experiments

We show the results of our experiments on complex event detection and video captioning tasks. We implement our model using the TensorFlow framework [3].

4.1. Complex Event Detection

4.1.1 Dataset

We collect approximately 220,000 videos without label information from TRECVID MED data, which excludes videos in MEDTest-13 and MEDTest-14, for unsupervised training. The average length of the collected videos is 130 seconds with a total duration of more than 8,000 hours.

We use the challenging MED datasets with labels, namely, TRECVID MEDTest-13 100Ex [1] and TRECVID MEDTest-14 100Ex [2] for video classification[1]. There are 20 events in each dataset, 10 of which overlap. It consists of approximately 100 positive exemplars for each event in the training set, and 5,000 negative exemplars. In the testing set, there are about 23,000 videos and the total duration in each collection is approximately 1,240 hours. The average video length is 120 seconds. These videos are temporally untrimmed YouTube videos of various resolutions and quality. We use the mean Average Precision (mAP) as the performance metric following the NIST standard [1, 2].

4.1.2 Model Specification

For both unsupervised training and classification, we uniformly sample video frames at the rate of 1 FPS and extract features for each frame from GoogLeNet with the Batch Normalization [17] pre-trained on ImageNet. Following standard image preprocessing procedures, the shorter edges of frames are rescaled to 256 and we crop the image to 224×224. We use activations after the last pooling layer and obtain representations with length 1,024. There are 20 classes in the MEDTest-13 and MEDTest-14 datasets, thus with the background class, we have 21 classes in total. In the training stage, we set sequence length K to 30 and pad zeros if the video has fewer than 30 frames. During inference, we take the whole video as input and use 150 steps.

Training details. We use the following settings in all experiments unless otherwise stated. The model is optimized with ADAM [20], and we fix the learning rate at 1×10^{-4} and clip the global gradients at norm 10. We use a single RNN layer for both the encoder and decoder, and the cell size is set to 1,024. We set the attention size to 50 and regularize the network by using Dropout [38] in the input and output layer [33]. We also add Dropout when the decoder copy state from the encoder and all dropout probability is set to 0.5. Weights are initialized with Glorot uniform initialization [13] and weight decay of 1×10^{-4} is applied for regularization. In the supervised training, we initialize the weights of the encoder using the learned weights during unsupervised learning, and the same sequence length is used as in the unsupervised training stage.

[1]Development data is not updated for TRECVID MED 15 and TRECVID MED 16 competition.

Methods	MEDTest-13	MEDTest-14
GoogLeNet	32.0	25.1
mGRU	**39.6**	**32.2**

Table 1. Comparison between GoogLeNet features and our mGRU model. Average pooling is used for both models. The result shows our feature representation significantly outperforms the GoogLeNet feature.

Methods	MEDTest-13	MEDTest-14
GoogLeNet	42.0	33.6
mGRU	**44.5**	**37.3**

Table 2. Comparison between GoogLeNet and mGRU models when VLAD encoding is used to aggregate frame-level features.

4.1.3 Results

Average pooling. For the GoogLeNet baseline, we average frame-level features and use a Linear SVM for classification. For our model, we first train an unsupervised encoder-decoder model with mGRU and fine-tune the encoder with label information. To make a fair comparison with the GoogLeNet baseline, we extract outputs of the mGRU encoder at each step and average them to obtain a global representation for classification. Note that both feature representations have same dimensions and we empirically set $C = 1$ for both of the linear classifiers. The result is shown in Table 1 and shows that our model with temporal structure learning is able to encode valuable temporal information for classification.

VLAD Encoding. We now show that VLAD encoding is useful for aggregating RNN representations. We compare our method with GoogLeNet features using VLAD encoding. Following [49], we set the number of k-means centers to 256 and the dimension of PCA is 256. Three scales are learned at each step for our mGRU model. We divide the state into three segments and each sub-state is individually aggregated by VLAD. Note that each encoded representation has the same feature vector length as the GoogLeNet model, and we use late fusion to combine the scores of the three scales. The results in Table 2 show that our mGRU model outperforms GoogLeNet features when encoded by VLAD. It also shows that VLAD encoding outperforms average pooling for RNN representations. Our model also achieves state-of-the-art performance on the MEDTest-13 and MEDTest-14 100Ex datasets.

4.1.4 Ablation Study

We compare several variants in the unsupervised training, and show the performance of different components. The results are shown in Table 3. We obtain features from the

Methods	MEDTest-13	MEDTest-14
mGRU w/o attention	32.7	27.5
mGRU w/o context	37.1	30.1
mGRU w/o multirate	36.5	29.3
mGRU (full)	37.4	30.6

Table 3. Comparison between mGRU and other variants in the unsupervised training stage. Detailed discussion can be found in text.

Methods	MEDTest-13	MEDTest-14
mGRU (random)	38.3	29.5
mGRU (pre-trained)	39.6	32.2

Table 4. Comparison between models which have the same structure but different initialization. This shows that good initialization enables better features to be learned.

unsupervised model by extracting states from the encoder at each step, which are then averaged to obtain a global video representation. The results show that the representation learning from unsupervised training without discriminative information also achieves good results.

Attention. We compare our model with a model without the attention mechanism, where temporal attention is not used and the decoder is forced to perform reconstruction based only on the last encoder state, *i.e.*, "mGRU w/o attention" in Table 3. The results show that the attention mechanism is important for learning good video representations and also helps the learning process of the encoder.

Context. In a model without context reconstruction, *i.e.*, only one decoder is used (autoencoder), neither past nor future context information is considered, *i.e.*, "mGRU w/o context" in Table 3. The results show that with context prediction, the encoder has to consider temporal information around the video clip, which models the temporal structures in a better way.

Multirate. We also show the benefit of using mGRU by comparing it with the basic GRU, *i.e.*, "mGRU w/o multirate" in Table 3. Note that the mGRU model has fewer parameters but better performance. It shows that an mGRU that encodes multirate video information is capable of learning better representations from long, noisy sequences.

Pre-training. We now show the advantages of the unsupervised pre-training process by comparing an encoder with random initialization with the same encoder whose weights are initialized by the unsupervised model. The result is shown in Table 4 and demonstrates that the unsupervised training process is beneficial to video classification. It incorporates context information in the encoder, which is an important cue for the video classification task.

Models	MEDTest-13	MEDTest-14
IDT + FV [49]	34.0	27.6
IDT + skip + FV [24]	36.3	29.0
VGG + RBF [53]	-	35.0
C3D [42] *	36.9	31.4
VGG16 + VLAD [49]	-	33.2
NI-SVM$_2$ [5]	39.2	34.4
VGG16+LCD+VLAD [49]	40.3	35.7
LSTM autoencoder [39] *	38.2	31.0
GoogLeNet + VLAD *	42.0	33.6
Our method	**44.5**	**37.3**

Table 5. Comparison with other methods. We achieve state-of-the-art performance on both MEDTest-13 and MEDTest-14 100Ex datasets. * denotes that the model is implemented by ourselves.

4.1.5 Comparison with the State-of-the-art

We compare our model with other models and the results are shown in Table 5. Our single model achieves the state-of-the-art performance on both the MEDTest-13 and MEDTest-14 100Ex settings compared with the performances of other single models. We report the C3D result by using the pre-trained model [42] and we set the length of the input short clip to 16. Features are averaged across clips which are classified with a Linear SVM. Our model with VLAD encoding outperforms previous state-of-the-art results with 4.2% on MEDTest-13 100Ex and 1.6% on MEDTest-14 100Ex.

4.2. Video Captioning

We now validate our model on the video captioning task. Our single model outperforms previous state-of-the-art single models across all metrics.

4.2.1 Dataset

We use the YouTube2Text video corpus [6] to evaluate our model on the video captioning task. The dataset has 1,970 video clips with an average duration of 9 seconds. The original dataset contains multi-lingual descriptions covering various domains, *e.g.*, sports, music, animals. Following [45], we use English descriptions only and split the dataset into training, validation and testing sets containing 1,200, 100, 670 video clips respectively. In this setting, there are 80,839 descriptions in total with about 41 sentences per video clip. The vocabulary size we use is 12,596 including `<GO>`, `<PAD>`, `<EOS>`, `<UNK>`.

We evaluate the performance of our method on the test set using the evaluation script provided by [7] and the results are returned by the evaluation server. We report BLEU [30], METEOR [9] and CIDEr [43] scores for comparison with other models. We stick with a single rule dur-

Methods	B@1	B@2	B@3	B@4	M	C
GRU	79.46	67.52	57.98	47.14	32.31	72.46
mGRU	79.42	67.79	58.32	48.12	32.79	73.21
mGRU+ pre-train	**80.76**	**69.49**	**60.03**	**49.45**	**33.39**	**75.45**

Table 6. Comparison between different models on YouTube2Text dataset. GoogLeNet features are used as frame-level representations. **B**, **M**, **C** are short for BLEU, METEOR, CIDEr.

Methods	B@1	B@2	B@3	B@4	M	C
GRU	80.88	70.15	61.08	51.06	33.48	79.16
mGRU	82.03	71.41	62.38	52.49	33.91	78.41
mGRU+ pre-train	**82.49**	**72.16**	**63.30**	**53.82**	**34.45**	**81.20**

Table 7. Comparison between different models on YouTube2Text dataset. ResNet-200 features are used as frame-level representations. **B**, **M**, **C** are short for BLEU, METEOR, CIDEr.

ing model selection, namely we choose the model with the highest METEOR score on the validation set.

4.2.2 Model Specification

The video length in the YouTube2Text dataset is short, thus we uniformly sample frames at a higher frame rate of 15 FPS. The sequence length is set to 50 and we use the default hyper-parameters in the last experiment. We use two different convolutional features for the video captioning task, *i.e.*, GoogLeNet features and ResNet-200 features [15]. We use beam search during decoding by default and set the beam size to 5 following [52] in all experiments. Attention size is set to 100 empirically.

4.2.3 Results

We first use GoogLeNet features and the result is shown in Table 6. We compare our mGRU with GRU which shows that mGRU outperforms GRU on all metrics except BLEU@1. However, the difference is only 0.04%. We initialize the mGRU encoder via unsupervised context learning and the result shows that with good initialization, performance is improved by more than 1.0% on the BLEU and CIDEr scores and 0.6% on the METEOR score compared with random initialization. We also utilize the recent ResNet-200 network as a convolutional model. We use the pre-trained model and follow the same image preprocessing method. The result of using ResNet-200 is shown in Table 7 and demonstrates that our MVRM method not only works better than GRU on different tasks, but also works better on different convolutional models. Additionally, we can improve all the metrics with ResNet-200 network.

Methods	BLEU@4	METEOR	CIDEr
S2VT [44]	-	29.20	-
Temporal attention [51]	41.92	29.60	51.67
GoogLeNet+ Bi-GRU-RCN$_1$ [4]	48.42	31.70	65.38
GoogLeNet+ Bi-GRU-RCN$_2$ [4]	43.26	31.60	68.01
VGG+LSTM-E [29]	40.20	29.50	-
C3D+LSTM-E [29]	41.70	29.90	-
GoogLeNet+HRNE+ Attention [28]	43.80	33.10	-
VGG+p-RNN [52]*	44.30	31.10	62.10
C3D+p-RNN [52]*	47.40	30.30	53.60
GoogLeNet+MVRM	**49.45**	**33.39**	**75.45**

Table 8. Comparison with other models without fusion. * denotes that the model is trained with different settings ([52] used the train+val data for training).

4.2.4 Comparison with the State-of-the-art

We compare our methods with other models on the YouTube2Text dataset. Results are shown in Table 8. "S2VT" [44] is the first model to use a general encoder-decoder model for video captioning. "Temporal Attention" [51] uses the temporal attention mechanism on the video frames to obtain better results. "Bi-GRU-RCN" [51] uses a ConvGRU to encode activations from different convolutional layers. "LSTM-E" [29] uses embedding layers to jointly project visual and text features. Our MVRM method has similar performance to [28], but with the pre-training stage, we outperform [28] in all metrics. Some methods fuse additional motion features like C3D [42] features, *e.g.*, Pan *et al*. [28] obtained 33.9% on METEOR after combing multiple features. With ResNet-200, we can obtain **34.45%** on METEOR.

5. Conclusion

In this paper, we propose a Multirate Visual Recurrent Model to learn multirate representations for videos. We model the video temporal structure via context reconstruction, and show that unsupervised training is important for learning good representations for both video classification and video captioning. The proposed method achieves state-of-the-art performance on two tasks. In the future, we will investigate the generality of the video representation in other challenging tasks, *e.g.*, video temporal localization [11] and video question answering [54, 41]

Acknowledgments. Our work is partially supported by the Data to Decisions Cooperative Research Centre (www.d2dcrc.com.au), and Google Faculty Award, and an Australian Government Research Training Program Scholarship. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the TITAN X (Pascal) GPU used for this research.

References

[1] TRECVID MED 13. http://nist.gov/itl/iad/mig/med13.cfm, 2013. 6

[2] TRECVID MED 14. http://nist.gov/itl/iad/mig/med14.cfm, 2014. 6

[3] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. In *OSDI*, 2016. 5

[4] N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. *ICLR*, 2016. 2, 8

[5] X. Chang, Y. Yang, E. P. Xing, and Y.-L. Yu. Complex event detection using semantic saliency and nearly-isotonic svm. In *ICML*, 2015. 7

[6] D. L. Chen and W. B. Dolan. Collecting highly parallel data for paraphrase evaluation. In *ACL*, 2011. 7

[7] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015. 7

[8] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006. 2

[9] M. Denkowski and A. Lavie. Meteor Universal: Language specific translation evaluation for any target language. In *EACL*, 2014. 7

[10] J. Donahue, L. Anne 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. 2

[11] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles. ActivityNet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015. 2, 8

[12] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 4

[13] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 6

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[15] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 8

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

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 6

[18] H. Jegou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 2, 5

[19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2

[20] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[21] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *NIPS*, 2015. 4

[22] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber. A clockwork RNN. In *ICML*, 2014. 3

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[24] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj. Beyond Gaussian pyramid: Multi-skip feature stacking for action recognition. In *CVPR*, 2015. 7

[25] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 2

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013. 4

[27] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 1, 2, 5

[28] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. In *CVPR*, 2016. 1, 3, 8

[29] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui. Jointly modeling embedding and translation to bridge video and language. In *CVPR*, 2016. 8

[30] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A method for automatic evaluation of machine translation. In *ACL*, 2002. 7

[31] V. Pătrăucean, A. Handa, and R. Cipolla. Spatio-temporal video autoencoder with differentiable memory. In *ICLR Workshop*, 2016. 2

[32] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *ECCV*, 2010. 5

[33] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. Dropout improves recurrent neural networks for handwriting recognition. In *ICFHR*, 2014. 6

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. 2

[35] D. L. Schacter, D. R. Addis, D. Hassabis, V. C. Martin, R. N. Spreng, and K. K. Szpunar. The future of memory: Remembering, imagining, and the brain. *Neuron*, 76(4):677–694, 2012. 2

[36] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1, 2

[37] 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. 2

[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014. 6

[39] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015. 2, 7

[40] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014. 3, 4

[41] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler. MovieQA: Understanding stories in movies through question-answering. In *CVPR*, 2016. 8

[42] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015. 1, 2, 7, 8

[43] R. Vedantam, C. Lawrence Zitnick, and D. Parikh. CIDEr: Consensus-based image description evaluation. In *CVPR*, 2015. 7

[44] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence – video to text. In *ICCV*, 2015. 3, 8

[45] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. In *NAACL HLT*, 2015. 7

[46] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *CVPR*, 2011. 1, 2

[47] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 1, 2

[48] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 2

[49] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative CNN video representation for event detection. In *CVPR*, 2015. 1, 2, 5, 6, 7

[50] Z. Xu, L. Zhu, Y. Yang, and A. G. Hauptmann. UTS-CMU at THUMOS 2015. *THUMOS Challenge*, 2015. 2

[51] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In *ICCV*, 2015. 3, 8

[52] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *CVPR*, 2016. 3, 8

[53] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov. Exploiting image-trained CNN architectures for unconstrained video classification. In *BMVC*, 2015. 7

[54] L. Zhu, Z. Xu, Y. Yang, and A. G. Hauptmann. Uncovering temporal context for video question and answering. *arXiv preprint arXiv:1511.04670*, 2015. 8