Relational Long Short-Term Memory for Video Action Recognition

Zexi Chen
zchen22@ncsu.edu

Bharathkumar Ramachandra
bramach2@ncsu.edu

Tianfu Wu
tianfu_wu@ncsu.edu

Ranga Raju Vatsavai
rrvatsav@ncsu.edu

Abstract

Spatial and temporal relationships, both short-range and long-range, between objects in videos are key cues for recognizing actions. It is a challenging problem to model them jointly. In this paper, we first present a new variant of Long Short-Term Memory, namely Relational LSTM to address the challenge for relation reasoning across space and time between objects. In our Relational LSTM module, we utilize a non-local operation similar in spirit to the recently proposed non-local network [35] to substitute the fully connected operation in the vanilla LSTM. By doing this, our Relational LSTM is capable of capturing long and short-range spatio-temporal relations between objects in videos in a principled way. Then, we propose a two-branch neural architecture consisting of the Relational LSTM module as the non-local branch and a spatio-temporal pooling based local branch. The local branch is introduced for capturing local spatial appearance and/or short-term motion features. The two-branch modules are concatenated to learn video-level features from snippet-level ones which are then used for classification. Experimental results on UCF-101 and HMDB-51 datasets show that our model achieves state-of-the-art results among LSTM-based methods, while obtaining comparable performance with other state-of-the-art methods (which use not directly comparable schema). Our code will be released.

1. Introduction

Action recognition (or video classification) is the task of assigning one of many class labels to a short video clip containing an action. Action recognition in videos plays a crucial role in many applications, e.g. visual surveillance [19], sport video analysis [24], human-machine interaction [30], etc. It has piqued the interest of the computer vision and deep learning communities, owing to the fact that the performances of state-of-the-art approaches are still well below human-level performance. Action recognition is a more complicated task when compared to still image classification because the temporal domain introduces variations in motion and viewpoints which have to be accounted for. Additionally, the use of a moving camera rather than a static camera introduces variations that could make optical flow based features less reliable. Besides, the interactions of multiple objects in actions make the classification task even more challenging.

Intuitively, action recognition requires models capable of learning key features such as:

Appearance features: Some actions are defined by certain special objects. For the "Blowing Candles" class in the UCF-101 dataset [25], the presence of a candle in any one frame is sufficient to correctly classify the action.

Short-term motion features: Some actions are characterized by particular short-term motion with large variations of appearance. For the "Boxing Speed Bag" class in the UCF-101 dataset, a short optical flow snippet of the video clip is sufficient to correctly classify the action.

Long-term trajectory features: Similarly, some actions depend on long-term object trajectories which are defined by appearance and motion jointly. For the "Golf Swing" class in the UCF-101 dataset, a longer optical flow snippet of the video clip composed of the long-term motion of the arm and golf club is required to correctly classify the action.

Object interaction features: Multi-object based actions are frequently observed and entails modeling of object interactions. For the "Frisbee Catch" class in the UCF-101 dataset, the interaction of the Frisbee moving between 2 persons is required to correctly classify the action.

Clearly, key features stated above do not present themselves in a mutually exclusive form in popular benchmark datasets; appearance features would be useful regardless of whether the action possessed interactions between objects. Nevertheless, one popular approach that has brought recent success to modeling these features is that of signal decomposition. Inspired by the discovery that the Human Visual System has separate processing pathways for different types of signals such as fast and slow motion [15], researchers

have employed multi-stream architectures to process each of these features separately. This paper is also motivated by the idea of signal decomposition.

Most significantly, the two-stream architecture of Simoyan and Zisserman [23] pioneered research in this area for action recognition from videos. They design two parallel 2D ConvNet streams to learn appearance features from RGB images and motion features from optical flow fields. An alternative to the optical flow stream also presented itself in the form of 3D convolutions [14], although they have been used together too since then [4]. Since 2D convolution on an optical flow stream and 3D convolution were deemed to not be able to capture sequential information required to model long-term trajectory features well, researchers turned to explore ways to address this next. The use of LSTMs following 2D convolutions [20, 18] and various temporal fusion strategies [5, 6, 16, 2] emerged as competitive candidates. However, object interaction features were left unexplored for the most part until recently where researchers used a self-attention mechanism to develop non-local neural networks [35]. Their use of a "non-local operation" is able to model long-range interactions between objects in space and time, but their network is applied on short snippets cropped from the original videos. As such, they fail to explore truly modeling long-term trajectory features from information across the full lengths of videos.

In this paper, we explore the novel idea of introducing the non-local operation from [35] into an LSTM module to create a Relational LSTM. We hypothesize that introduction of our "relational LSTM" block into a two-stream architecture would aid in the modeling of features that capture object interactions, while retaining the property of LSTM to model long-term trajectory information. Our main contributions can be summarized as follows:

- We develop a novel Relational-LSTM module that models object interaction features and can be inserted into existing diverse architectures as a plug-in module.
- We incorporate the Relational-LSTM module into a two-stream, two-branch architecture to perform video action recognition.
- We show experimentally through ablation studies that the introduction of our module leads to clear and unquestionable gains in performance.
- Our architecture should be considered the new state-of-the-art for video action recognition among LSTM-based architectures, beating the current best architecture by 1.2% on UCF-101 and 5.2% on HMDB-51.
- Our architecture performs comparably to the top-tier of state-of-the-art architectures overall on UCF-101 and HBDM-51 datasets.

2. Related Work

2.1. Deep learning for appearance and short-term motion features

The success of 2D ConvNets did not immediately follow for video tasks, where hand-crafted features of Improved Dense Trajectories dominated. It was not until the work of [23] that deep learning approaches started to show comparable performance.

Two-stream ConvNets: In [23], the authors employ a two-stream architecture with 2D ConvNets to learn appearance and motion features to aid classification. They show that 2D ConvNets are by themselves capable of capturing short-term motion features with densely stacked optical flow fields as inputs. They average the predictions from a single RGB image and a stack of 10 consecutive optical flow fields after feeding them through two separate 2D ConvNets with identical structure. Based on this two-stream architecture, Wang *et al*. [34] propose the averaging pooling operator to aggregate multiple frame-level predictions into a video-level prediction to model long-range temporal structure over the entire video.

3D ConvNets: When viewing a video as a sequential stack of RGB images, it is natural to think of extending 2D convolution to the temporal dimension to model spatio-temporal features in videos. In early stages, Ji *et al*. [14] have attempted to replace pre-computed complex hand-crafted features with 3D ConvNets, but their network is still quite shallow with only three convolutional layers. Following their work, Tran *et al*. [29] further exploit 3D ConvNets' properties under various video datasets and experimentally show that 3D ConvNets are competent for learning appearance and short-motion features. Inflated 3D ConvNet (I3D) proposed by Carreira and Zisserman [4] makes full use of successful pre-trained image classification architectures by inflating all the filters and pooling kernels from 2D to 3D and achieves state of the art performance on UCF-101 [25] and HMDB-51 [17] datasets. Their competitive performance drove research in this direction. The authors in [39] show that factorizing 3D convolution of I3D into a 2D spatial convolution and a 1D temporal convolution, analogous to spatial factorization in Inception-v2 [28], yields slightly better accuracy.

2.2. Sequence modeling for long-term trajectory features

The aforementioned methods successfully model appearance and short-term motion features. However, with regard to long-term trajectory information, they either used unsuitable inputs (frames not spanning long temporal range), or they used the 3D convolution operation or optical flow inputs, which capture only *local* temporal properties. Alternatively, diverse methods have attempted to encode

long-term trajectory information. Some authors [32, 33, 21] make use of dense point trajectories by tracking densely sampled points using optical flow fields. Subsequently, this hand-crafted shallow video representation was replaced by deep representations learned from neural networks. The basic idea is to sequentially aggregate frame-level feature representations extracted from either 2D ConvNets or 3D ConvNets so that long-term trajectory information is encoded into the deep video-level representations. This could be done using either recurrent neural networks (RNNs) such as LSTMs or temporal feature fusion methods such as temporal pooling.

RNN-based architectures: The sequential modeling ability of LSTMs makes them appealing to use for capturing long-range temporal dynamics in videos. In [1], the authors propose applying an LSTM to high-level feature vectors extracted from 3D ConvNets, but they only apply it on short video snippets of 9 frames. Ng $et\ al.$ [20] add five stacked LSTM layers before the last fully connected layer of the two-stream ConvNets [23] and slightly improve the performance. Wang $et\ al.$ [37] design a hierarchical attention network, which is implemented by skipping time steps in higher layers of the stacked LSTM layers. Additionally, the authors in [41] employ an attention mechanism widely used in image captioning tasks to automatically focus on salient regions from high-level appearance or short-motion feature maps.

Temporal feature fusion architectures: Temporal feature pooling [20, 2] is the most popular temporal feature fusion method, which usually uses either max-pooling or average-pooling over the temporal dimension to aggregate frame-level features. Furthermore, the authors [16, 2] propose adaptive temporal feature pooling by simultaneously learning an importance score for each frame and use it as weight in the pooling process. Cherian $et\ al.$ [5] propose generalized ranking pooling, which projects all frame-level features together into a low-dimensional subspace and use an SVM classifier on the subspace representation. The subspace is parameterized by several orthonormal hyperplanes and is designed to have a property of preserving the temporal order of video frames. Besides temporal pooling, Diba $et\ al.$ [6] introduce a temporal linear encoding method, where they first aggregate frame-level features using element-wise multiplication and then project it to a lower dimensional feature space using bilinear model.

2.3. Non-local operation for object interaction features

Both the convolution and recurrent operations compute spatial and temporal features respectively in a primarily *local* fashion. Long-range dependencies are then modeled through applying these local operations repeatedly, often accompanied by downsampling, to propagate signals across space and time domains. The non-local operation [35] is one hypothesized solution to handle the remaining problem of object interaction modeling.

Relation reasoning Exploring object interactions is equivalent to reason the relations of objects. Recently, the authors in [22] propose Relation Network (RN), which is a neural network module primarily for relation reasoning. It uses one MLP layer on top of a batch of feature vector pairs to learn pairwise relations, where each instance in the batch is a pair of feature vectors at two particular positions in the input feature maps. They use this RN module on the visual question-and-answer problem and achieve superhuman level performance. Following their work, Zhou $et\ al.$[42] explore its usage on temporal relation reasoning in videos, which is implemented by sparsely sampling frames from videos and employing RN module to learn the causal relations among frames. A similar work for exploring object relations is proposed by [12], where they use 'Scaled Dot-Product Attention' [31] to compute object relations.

Non-local operation The non-local operation can be considered as a general form of 'Scaled Dot-Product Attention', as mentioned in [35]. The key idea of non-local operation is that the output features of a position are computed as a weighted sum of the features from all positions in the input feature maps, which allows contributions from features in distant positions. The non-local idea originates from [3] for image denoising, where the estimated value in pixel i is computed as the weighted average of all the pixels in the image. In [35], they leverage it to design a non-local block for a neural network, which can be used as a plug-in module inserted into diverse neural network architectures.

3. Relational LSTM Module

Considering the importance of object interaction features in videos, we propose a Relational LSTM module, which not only inherits the sequential modeling ability from LSTM but also incorporates spatial relation reasoning and temporal relation reasoning through a non-local operation. More specifically, we generalize the non-local operation in [35] to compute spatial relations among input features at a single snippet, and to compute temporal relations between input features and past learned features at previous time steps. Meanwhile, because of the use of LSTM, we create video-level feature representations by using selected snippets from the whole video, which inherently encodes long-term trajectory information in our representations.

Non-local operation: We first review the non-local operation defined in [35]. Given input feature maps $\boldsymbol{X} \in \mathbb{R}^{N\times C}$, where N represents the number of positions in $\boldsymbol{X}$, and C represents the number of dimensions of feature vector at each position. If we represent $\boldsymbol{X}$ as $\{\boldsymbol{x}_i\}_{i=1}^{N}$, the output $\boldsymbol{z}_i$ at i-th position of response feature maps $\boldsymbol{Z} \in \mathbb{R}^{N\times C}$ is a weighted sum over all input feature vectors, as shown

in Equation 1.

$$\begin{aligned} \boldsymbol{z}_i &= \sum_{j=1}^{N} \omega_{ij} g(\boldsymbol{x}_j) \\ \omega_{ij} &= \frac{f(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\mathcal{C}(\boldsymbol{X})} \end{aligned} \tag{1}$$

where $i, j, k \in \mathbb{R}^N$ are position indices, $f(\boldsymbol{x}_i, \boldsymbol{x}_k)$ represents compatibility function computing the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and $g(\boldsymbol{x}_j)$ is a unary function computing a representation of the input feature vector at j-th position. $\mathcal{C}(\boldsymbol{X})$ is the normalization factor, usually denoted as $\mathcal{C}(\boldsymbol{X}) = \sum_{k=1}^{N} f(\boldsymbol{x}_i, \boldsymbol{x}_k)$ or $\mathcal{C}(\boldsymbol{X}) = N$.

The non-local operation in Equation 1 considers all positions in $\boldsymbol{X}$ regardless of positional distance, which is in sharp contrast to a convolutional operation of considering absolutely local neighborhood. This non-local operation explores the relations of features globally, which can be adopted to learn object interaction features far from each other in a spatio-temporal layout.

Generalized non-local operation: First of all, we generalize the non-local operation to compute relations between any two feature maps $\boldsymbol{X} \in \mathbb{R}^{N \times C_X}, \boldsymbol{Y} \in \mathbb{R}^{N \times C_Y}$. Given feature maps $\boldsymbol{Y}$, we compute the output $\boldsymbol{Z} \in \mathbb{R}^{N \times C_Z}$ with respect to $\boldsymbol{X}$ as shown in Eq. 2

$$\begin{aligned} \boldsymbol{z}_i &= \sum_{j=1}^{N} \omega_{ij} g(\boldsymbol{y}_j) \\ \omega_{ij} &= \frac{f(\boldsymbol{x}_i, \boldsymbol{y}_j)}{\sum_{k=1}^{N} f(\boldsymbol{x}_i, \boldsymbol{y}_k)} \end{aligned} \tag{2}$$

This general form of the non-local operation can also be interpreted using the attention mechanism in [31]. Given a query feature vector $\boldsymbol{x}_i$ and a set of input feature vectors $\{\boldsymbol{y}_i\}_{i=1}^{N}$, the output $\boldsymbol{z}_i$ is computed as an attentional weighted sum of all input feature vectors, where the attentional weights are computed by a compatibility function of the query feature vector and input feature vectors. In the rest of the paper, we abbreviate this general form as $\boldsymbol{Z} = r(\boldsymbol{X}, \boldsymbol{Y})$.

Relational LSTM: Given a video V, we first divide it into T segments $\{S_1, S_2, ..., S_T\}$ of equal duration, and randomly sample one short snippet K_t from its corresponding segment S_t. Suppose $\boldsymbol{X}_t \in \mathbb{R}^{H \times W \times C}$ for $t = 1, ..., T$ represents the high-level feature maps obtained after feeding K_t through some Convolution layers of a pre-trained CNN model, where C is the number of feature maps, and H and W are the spatial height and width of each feature map. After extracting $\{X_1, ..., X_T\}$ from a convolution layer, a temporal aggregation function is required to encode these snippet-level feature maps into video-level feature maps. Inspired by the deep learning architecture LSTMs, which not only inherit the sequential modeling ability from vanilla RNNs but can also capture long-term dependencies through the memory cell mechanism, we employ LSTM-based architectures to this end.

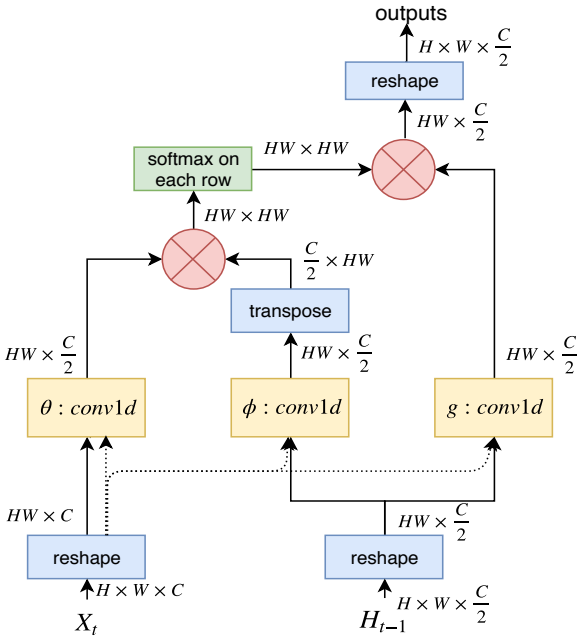

Figure 1. **Generalized non-local operations $r(\boldsymbol{X}_t, \boldsymbol{X}_t)$ and $r(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})$ in Relational LSTM.** "$conv1d$" denotes 1D convolutional operation, "$\otimes$" denotes matrix multiplication. Because the only difference between $r(\boldsymbol{X}_t, \boldsymbol{X}_t)$ and $r(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})$ is the inputs, we use dashed arrow for inputs of $r(\boldsymbol{X}_t, \boldsymbol{X}_t)$, and use solid arrow for inputs of $r(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})$.

Usually, given feature maps X_t which preserve spatial layout, ConvLSTM [40] is the natural choice as the aggregation function because it encodes spatial information through its convolutional operations. However, the experimental results in [26] have shown that ConvLSTM did not perform well on this recognition task, and it could not capture crucial object interaction features because of its convolution operations, which are applied to a *local* receptive field. We introduce the generalized non-local operation into LSTM architecture and present a new module called Relational LSTM. It differs from regular LSTM and ConvLSTM [40] in the aspect that the general form of non-local operation is used in both input-to-state transitions and state-to-state transitions. The key equations of Relational LSTM are shown in Equation 3.

$$
\begin{aligned}
\boldsymbol{i}_t &= \sigma[r_{ix}(\boldsymbol{X}_t, \boldsymbol{X}_t) + r_{ih}(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})] \\
\boldsymbol{f}_t &= \sigma[r_{fx}(\boldsymbol{X}_t, \boldsymbol{X}_t) + r_{fh}(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})] \\
\boldsymbol{o}_t &= \sigma[r_{ox}(\boldsymbol{X}_t, \boldsymbol{X}_t) + r_{oh}(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})] \\
\boldsymbol{g}_t &= \tanh[r_{gx}(\boldsymbol{X}_t, \boldsymbol{X}_t) + r_{gh}(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})] \\
\boldsymbol{C}_t &= \boldsymbol{f}_t \circ \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \circ \boldsymbol{g}_t \\
\boldsymbol{H}_t &= \boldsymbol{o}_t \circ \tanh(\boldsymbol{C}_t)
\end{aligned}
\tag{3}
$$

Here inputs $\boldsymbol{X}_t$, memory cell $\boldsymbol{C}_t$, hidden state $\boldsymbol{H}_t$, and gates $\boldsymbol{i}_t, \boldsymbol{f}_t, \boldsymbol{o}_t, \boldsymbol{g}_t$ have same functionalities as traditional LSTM. σ represents the logistic sigmoid non-linear activation function and $\tanh$ represents the hyperbolic tangent non-linear activation function.

We disentangle the mixed spatial-temporal relational reasoning. To ensure spatial relational reasoning is used, we adopt $r(\boldsymbol{X}_t, \boldsymbol{X}_t)$ in input-to-state transitions to model the feature interactions in same feature maps regardless of their relative positional distance. Regarding temporal relational reasoning, to model feature interactions of X_t with H_{t-1} which stores all important information from preceding feature maps, we adopt $r(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})$ in state-to-state transitions. The detailed implementations of $r(\boldsymbol{X}_t, \boldsymbol{X}_t)$ and $r(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})$ are shown in Fig. 1. In our implementations, we adopt the shape of $\boldsymbol{X}_t$ as $H \times W \times C$, and set $\boldsymbol{H}_{t-1}$ as $H \times W \times \frac{C}{2}$ to reduce memory cost. It is worth mentioning that even though we flatten the spatial layout when feeding X_t to the Relational LSTM block, we still preserve their relative positional information through Relational LSTM block and obtain output hidden feature maps H_t of shape as $(HW) \times \frac{C}{2}$, so that we can reshape H_t back to a shape of $H \times W \times \frac{C}{2}$.

In the generalized non-local operation $r(\boldsymbol{X}, \boldsymbol{Y})$ used in the Relational LSTM block, we adopt Embedded Gaussian function as $f(\boldsymbol{x}_i, \boldsymbol{y}_j)$ shown in Equation 4

$$
f(\boldsymbol{x}_i, \boldsymbol{y}_j) = e^{\theta(\boldsymbol{x}_i)^T \phi(\boldsymbol{y}_j)} \tag{4}
$$

where $\theta(\boldsymbol{x}_i) = W_\theta x_i$ and $\phi(\boldsymbol{y}_j) = W_\phi y_j$ are two linear embedding function. $g(\boldsymbol{y}_j)$ is also considered in the form of a linear embedding function expressed as $g(\boldsymbol{y}_j) = W_g \boldsymbol{y}_j$.

4. Network Architecture

We build our architecture on top of "two-stream ConvNets" [23], where a spatial stream operates on RGB images and a temporal stream operates on sequences of 10 optical flow fields, and their prediction scores are fused by weighted averaging. Our network architecture is shown in Fig. 2. The ResNet-101-v2 [11] has been employed as the backbone of our architecture. The detailed building blocks of ResNet-101-v2 could be found from paper [10]. We extract the feature maps $\{X_1, ..., X_T\}$ after ResNet-101-v2 conv5_2 layer, and split our network into two branches.

Local branch: We design this branch for learning information that can be learned via local operations. The information can either be appearance features with RGB images as inputs, or short-motion features with optical flow fields as inputs. Specifically, we continue adopting ResNet-101-v2 architecture (conv5_3, global spatial average pooling) on every individual short snippet to obtain snippet-level feature representations, then integrating them to a video-level representation by temporal average pooling.

Non-local branch: We design this branch for learning object interactions in space and time, taking into account long-range temporal dependencies. The Relational LSTM module in this branch can not only perform spatial relation reasoning and temporal relation reasoning among those snippets, but also obtain a video-level feature representation natively. As the spatial layout is preserved through the Relational LSTM module, and we further explore local features by adding one residual block (ResNet-101-v2 conv5_3 layer) after Relational LSTM module. Specifically, in our design of Relational LSTM module, we add one batch normalization (BN) layer just before the first reshaping operator, and one 1×1 convolutional layer after the outputs to increase the number of feature maps from $H \times W \times \frac{C}{2}$ to $H \times W \times C$. We initialize H_0 of the Relational LSTM module with zeros, assuming that no information has been observed when the video starts.

Finally, we concatenate the video-level representations generated by the two branches as a complement to each other, and add one fully connected layer to perform classification by taking the argmax.

5. Experiments

In this section, we first introduce the action recognition datasets we conduct experiments on and implementation details of our architecture including training and inferencing. Then, we study different aspects of our network on split 1 of UCF-101 dataset. Finally, we compare our architecture with state-of-the-art methods.

5.1. Datasets

We conduct a series of experiments on two challenging video action recognition benchmark datasets, UCF-101 [25] and HMDB-51 [17]. The UCF-101 dataset consists of 13,320 short video clips with 101 action classes. The HMDB-51 dataset consists of 6,766 short video clips with 51 action classes. There are more than 100 video clips in every action class in both datasets. For both datasets, we use the provided evaluation schema and report the mean average accuracy over 3 training/testing splits.

5.2. Implementation details

Training: We separately train the two streams of our architecture. The backbone ResNet-101-v2 is pre-trained

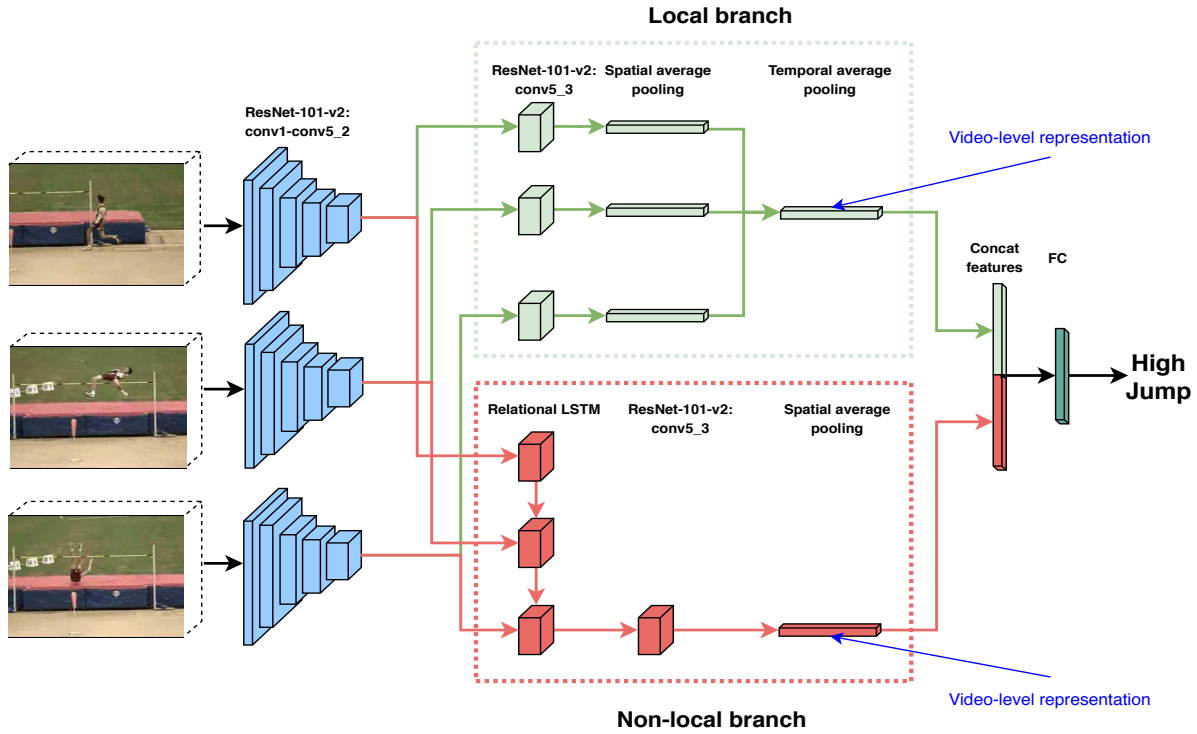

Figure 2. **Network architecture.** Only spatial stream is shown; the temporal stream is identical in structure. Given an input video, we divide it into T segments (T=3 in this case for succinct illustration) of equal duration, and randomly sample one short snippet from its corresponding segment. The short snippet is either a single RGB image (spatial stream) or a sequence of 10 optical flow fields (temporal stream). After feeding the selected short snippets through ResNet-101-v2 conv1 layer to ResNet-101-v2 conv5_2 layer, we have two separate branches. The local branch (green dotted box) captures local appearance and short-term motion features from snippets, and generates a video-level feature representation using temporal average pooling. The non-local branch (red dotted box) is for relation reasoning using Relation LSTM module. Eventually, we obtain an overall video-level feature representation by concatenating feature vectors generated by the two branches, and add one Fully Connected layer and optimize using a standard softmax with cross entropy classification loss.

on ImageNet. We employ the same data augmentations used in [34] e.g. horizontal flipping and multi-scale cropping. We adopt mini-batch stochastic gradient descent with a momentum of 0.9 and weight decay of 0.0005 as our optimizer. And we add a dropout layer right before the final fully connected layer. We train our model with Batch Normalization [13]. Traditionally, batch mean and variance are adopted for training in Batch Normalization, and moving mean and moving variance are adopted for inferencing. However, because of the limited number of training instances and batch size, the estimates of mean and variance from each batch is highly variant from moving mean and moving average. It leads to severe over-fitting due to divergent distributions of data in training and inferencing stages after BN layer, as mentioned in [34]. Although completely freezing mean and variance parameters of all BN layers work well in practice, we found that adopting moving mean and moving variance in the training stage with a very small momentum (e.g. 0.001, 0.0005) to gradually update them makes more sense as they are progressively learning population mean and variance of the training dataset.

For the spatial stream, we initialize parameters for conv1-conv5_2 layers and conv5_3 layer in the local branch from pre-trained ResNet-101-v2 model, and initialize Relational LSTM module and conv5_3 layer in the non-local branch using Xavier-Glorot initialization [9]. We set dropout rate to 0.8 and batch size to 24 (when T = 8). We train the model for 50 epochs. The learning rate starts at 0.0005 and is reduced by a factor of 10 after 35-th epoch and 45-th epoch.

For the temporal stream, we employ the cross modality initialization strategy used by [34], where the parameters are initialized from our trained spatial stream model. We set dropout rate to 0.7 and batch size to 24 (when T=8). We train the model for 60 epochs. The learning rate starts at 0.0005 and is reduced by a factor of 10 after 45-th epoch and 55-th epoch.

Inference: For testing our architecture, we sample 1 RGB image or a sequence of 10 optical flow fields from the same position of each segment ($T = 8$ segments by default)

to form one testing group. Meanwhile, we generate 5 crops (4 corners and 1 center) of 224×224 from the sampled images in the group. And we generate 4 groups by sampling from 4 positions with equal temporal spacing. So the final prediction scores for each stream are obtained by averaging over the 20 testing examples. We fuse the prediction scores of the two streams by weight averaging and the fusion is conducted before softmax normalization. Our empirical experiments suggest that the weight should be chosen around 0.5 for each stream.

5.3. Experimental evaluation on Relational LSTM network

In this experiment, we investigate how well standalone Relational LSTM module can perform on this task. We exclude local branch from our architecture and name the rest Relational LSTM network, and conduct experiments on it using split 1 of UCF-101 dataset. The spatial stream is initialized from pre-trained ResNet-101-v2 on ImageNet, and the temporal stream is initialized from the temporal stream of Temporal Segment Networks (TSN) proposed in [34]. We compare its performance with our implementation of TSN using ResNet-101-v2 as backbone in Table 1. From the table, we observe that Relational LSTM network itself achieves a similar overall performance as TSN. Combining the two models yields obvious increases in both spatial stream and temporal streams, indicating that TSN and our Relational LSTM network are complementary to some extent. This complementary behavior is to be expected as the Relational LSTM network is designed to focus more on object interaction and long-term motion features, whereas TSN emphasizes learning appearance and short-motion features.

Methods	Spatial	Temporal	Two-stream
TSN	87.0%	88.5%	93.8%
R-LSTM	86.9%	87.5%	93.8%
TSN + R-LSTM	88.6%	89.1%	94.2%

Table 1. Performance of Relational LSTM network, TSN and the ensemble of Relational LSTM network and TSN on UCF-101 (split 1). Relational LSTM network uses 8 segments ($T = 8$) in this experiment. We choose the best fusion weights in late fusion, 0.5 for spatial stream in Relation LSTM network, and 0.35 for spatial stream in TSN, and average weight for each stream in the ensemble.

5.4. Ablation studies

In this section, we explore different aspects of our architecture. The experiments are all conducted on split 1 of UCF-101 dataset.

The number of input video segments: The most important parameter influencing our model performance is the number of input video segments. We mutate our architecture with a different number of input video segments from $T = 6$ to $T = 12$. The results are shown in Table 2. From the table, we see that there is a large increase (1.5%) in the spatial stream from $T = 6$ to $T = 8$, which implies that more segments in a video can provide richer information serving as a better video representation. However, on continuing to increase the number of segments, the performance takes a hit. This could partly owe to the naïve temporal pooling operation in the local branch causing a damping of the correct action signal. Therefore, we set $T = 8$ for the rest of our experiments.

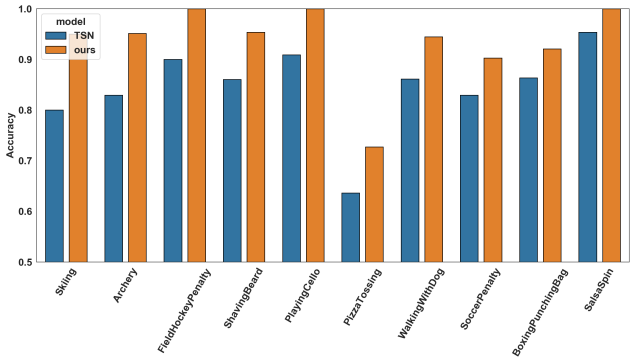

Figure 3. 10 classes of UCF-101 (split 1) with largest improvements from TSN to our two-branch architecture.

Number of segments	Spatial	Temporal	Two-stream
6	87.2%	87.9%	94.2%
8	**88.7%**	**88.1%**	**94.4%**
10	88.1%	87.5%	94.1%
12	87.4%	87.4%	93.5%

Table 2. Performance of our architecture with different number of input video segments on UCF-101 (split 1).

Effect of introducing non-local branch: Since there are two branches in our architecture, we were compelled to quantify the benefit of introducing the non-local branch into our architecture. In this experiment, we compare the performance of including vs. excluding non-local branch in our architecture. The results are shown in Table 3, and we find that there are improvements in each of the individual streams as well as overall. These small improvements on a large-scale dataset such as UCF-101 indicate that the contributions of the non-local branch in our architecture are significant and crucial.

5.5. Comparison with related work

Comparison with Temporal Segment Networks: First, we compare our architecture with TSN on split 1 of UCF-101 dataset. The reason we compare with TSN is that our

Methods	Spatial	Temporal	Two-stream
Local branch only	87.4%	87.3%	94.0%
Two-branch	**88.7%**	**88.1%**	**94.4%**

Table 3. Performance comparison of our architecture with only local branch vs. two-branch on UCF-101 (split 1).

local branch is most similar to TSN, the only difference being that our local branch aggregates snippet-level features before FC layer, whereas TSN aggregates snippet-level prediction scores after FC layer. The performance of our implementation of TSN is shown in Table 1, and the performance of our architecture is shown in Table 3. We observe that there is a 0.6% increase in overall performance. In Figure 3, we show the 10 classes of UCF-101 with the largest improvements in our architecture over TSN. Most of those classes involve object interaction features, e.g. PizzaTossing, Archery, implying that the introduction of non-local branch reaps benefits towards relation reasoning.

Comparison with LSTM-based state-of-the-art methods: As our method belongs to the family of LSTM-based methods, we compare our method with other LSTM-based methods over all three splits of UCF-101 and HMDB-51, as shown in Table 4. From the table, we observe that our method outperforms other LSTM-based methods by a large margin. Even with the standalone Relational LSTM network (non-local branch), over three splits of UCF-101, we achieve 94.2% accuracy, which convincingly outperforms other LSTM-based methods. To the best of our knowledge, we achieve the best performance among all LSTM-based methods.

Methods	UCF-101	HMDB-51
Two-Stream+LSTM [20]	88.6%	-
VideoLSTM [18]	89.2%	56.4%
HAN [37]	92.7%	64.3%
L^2STM [26]	93.6%	66.2%
Ours	**94.8%**	**71.4%**

Table 4. Comparison with LSTM-based state-of-the-art architectures on UCF-101 and HMDB-51 datasets. The performance accuracy is reported over all three splits.

Comparison with non-LSTM-based state-of-the-art methods: To demonstrate the overall performance of our model, we compare our architecture with current non-LSTM-based state-of-the-art methods over all three splits of UCF-101 and HMDB-51 datasets. We report these results in table 5. From the table, we can observe that we obtain performance comparable to the top tier of existing state-of-the-art methods. It is worth noting that Optical Flow Guided ConvNets [27] solve this task from a completely different angle, where they design an Optical Flow guided Feature (OFF) for learning short-motion representations, which makes it a good candidate for combining with our LSTM-based method as mentioned by the authors [27]. Moreover, our method is simple and the resulting networks very easy to train.

Methods	UCF-101	HMDB-51
iDT [33]	86.4%	61.7%
Two-Stream [23]	88.0%	59.4%
KVMDF [43]	93.1%	63.3%
ST-ResNet [7]	93.4%	66.4%
Two-Stream I3D [4]	93.4%	66.4%
TSN [34]	94.0%	68.5%
ST-Multiplier [8]	94.2%	68.9%
ST-Pyramid Network [36]	94.6%	68.9%
CoViAR [38]	94.9%	70.2%
Optical Flow Guided ConvNets [27]	96.0%	74.2%
Ours	94.8%	71.4%

Table 5. Comparison with non-LSTM-based state-of-the-art methods on UCF-101 and HMDB-51 datasets. The performance accuracy is reported over all three splits. For a fair comparison, we only consider models that are pre-trained on ImageNet. We consistently choose 0.45 for spatial stream and 0.55 for temporal stream in late fusion over the three splits of UCF-101 dataset. And we consistently choose 0.33 for spatial stream and 0.67 for temporal stream in late fusion over the three splits of HMDB-51 dataset.

6. Conclusions and Future Work

In this paper, we present a novel Relational LSTM module which we embed into a two-branch architecture for relation reasoning across space and time between objects in videos. It complements most existing action recognition methods for their lack of relation reasoning and learns video-level representations implicitly for modeling long-term trajectory features. In our experiments, we validate the contributions of introducing Relational LSTM module, and demonstrate the performance of our architecture on two challenging action recognition datasets. Before our work, LSTM-based methods have lagged behind non-LSTM-based methods in performance, especially those that use I3D convolutions [4]. Our method achieves state-of-the-art results among LSTM-based competitors and even enjoys performance comparable to non-LSTM-based counterparts. Even though we focus on the action recognition task in our experiments, much like the non-local block of Wang *et al.* [35], our Relational LSTM module can be inserted into various network architectures for other tasks. In this work, we only explore the application of our Relational LSTM module in two-stream ConvNets. In the future, we will explore the possibility of applying it on 3D ConvNets.

References

[1] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011. 3

[2] H. Bilen, B. Fernando, E. Gavves, and A. Vedaldi. Action recognition with dynamic image networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2, 3

[3] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005. 3

[4] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733. IEEE, 2017. 2, 8

[5] A. Cherian, B. Fernando, M. Harandi, and S. Gould. Generalized rank pooling for activity recognition. In *CVPR*, 2017. 2, 3

[6] A. Diba, V. Sharma, and L. Van Gool. Deep temporal linear encoding networks. In *Computer Vision and Pattern Recognition*, 2017. 2, 3

[7] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in neural information processing systems*, pages 3468–3476, 2016. 8

[8] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7445–7454. IEEE, 2017. 8

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. 6

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[11] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 5

[12] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2018. 3

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015. 6

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

[15] P. K. Kaiser and R. M. Boynton. Human color vision. 1996. 1

[16] A. Kar, N. Rai, K. Sikka, and G. Sharma. Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3376–3385, 2017. 2, 3

[17] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2556–2563. IEEE, 2011. 2, 5

[18] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. Snoek. Videolstm convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 166:41–50, 2018. 2, 8

[19] W. Lin, M.-T. Sun, R. Poovandran, and Z. Zhang. Human activity recognition for video surveillance. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 2737–2740. IEEE, 2008. 1

[20] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 4694–4702. IEEE, 2015. 2, 3, 8

[21] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *European Conference on Computer Vision*, pages 581–595. Springer, 2014. 3

[22] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4974–4983, 2017. 3

[23] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 2, 3, 5, 8

[24] K. Soomro and A. R. Zamir. Action recognition in realistic sports videos. In *Computer vision in sports*, pages 181–208. Springer, 2014. 1

[25] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 1, 2, 5

[26] L. Sun, K. Jia, K. Chen, D.-Y. Yeung, B. E. Shi, and S. Savarese. Lattice long short-term memory for human action recognition. In *ICCV*, pages 2166–2175, 2017. 4, 8

[27] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang. Optical flow guided feature: a fast and robust motion representation for video action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 8

[28] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016. 2

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

[30] S. Vantigodi and R. V. Babu. Real-time human action recognition from motion capture data. In *Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG),*

2013 Fourth National Conference on, pages 1–4. IEEE, 2013. 1

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017. 3, 4

[32] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011. 3

[33] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3551–3558. IEEE, 2013. 3, 8

[34] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016. 2, 6, 7, 8

[35] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. *arXiv preprint arXiv:1711.07971*, 2017. 1, 2, 3, 8

[36] Y. Wang, M. Long, J. Wang, and P. S. Yu. Spatiotemporal pyramid network for video action recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2097–2106. IEEE, 2017. 8

[37] Y. Wang, S. Wang, J. Tang, N. O'Hare, Y. Chang, and B. Li. Hierarchical attention network for action recognition in videos. *arXiv preprint arXiv:1607.06416*, 2016. 3, 8

[38] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Compressed video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6026–6035, 2018. 8

[39] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning for video understanding. *arXiv preprint arXiv:1712.04851*, 2017. 2

[40] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015. 4

[41] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015. 3

[42] B. Zhou, A. Andonian, and A. Torralba. Temporal relational reasoning in videos. *arXiv preprint arXiv:1711.08496*, 2017. 3

[43] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao. A key volume mining deep framework for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1991–1999, 2016. 8