Lattice Long Short-Term Memory for Human Action Recognition

Lin Sun[1,2], Kui Jia[3], Kevin Chen[2], Dit Yan Yeung[1], Bertram E. Shi[1], and Silvio Savarese[2]

[1]The Hong Kong University of Science and Technology
[2]Stanford University
[3]South China University of Technology

Abstract

Human actions captured in video sequences are three-dimensional signals characterizing visual appearance and motion dynamics. To learn action patterns, existing methods adopt Convolutional and/or Recurrent Neural Networks (CNNs and RNNs). CNN based methods are effective in learning spatial appearances, but are limited in modeling long-term motion dynamics. RNNs, especially Long Short-Term Memory (LSTM), are able to learn temporal motion dynamics. However, naively applying RNNs to video sequences in a convolutional manner implicitly assumes that motions in videos are stationary across different spatial locations. This assumption is valid for short-term motions but invalid when the duration of the motion is long.

In this work, we propose Lattice-LSTM (L^{2}STM), which extends LSTM by learning independent hidden state transitions of memory cells for individual spatial locations. This method effectively enhances the ability to model dynamics across time and addresses the non-stationary issue of long-term motion dynamics without significantly increasing the model complexity. Additionally, we introduce a novel multi-modal training procedure for training our network. Unlike traditional two-stream architectures which use RGB and optical flow information as input, our two-stream model leverages both modalities to jointly train both input gates and both forget gates in the network rather than treating the two streams as separate entities with no information about the other. We apply this end-to-end system to benchmark datasets (UCF-101 and HMDB-51) of human action recognition. Experiments show that on both datasets, our proposed method outperforms all existing ones that are based on LSTM and/or CNNs of similar model complexities.

1. Introduction

Video based human action recognition is of importance for many applications, including surveillance[20], abnormal activity detection [3], human computer interaction (HCI) [6] and so forth. Unlike recognition in static images, human action recognition relies on motion dynamics in addition to visual appearance. Designs of many human action recognition datasets [18, 28] are based on this observation.

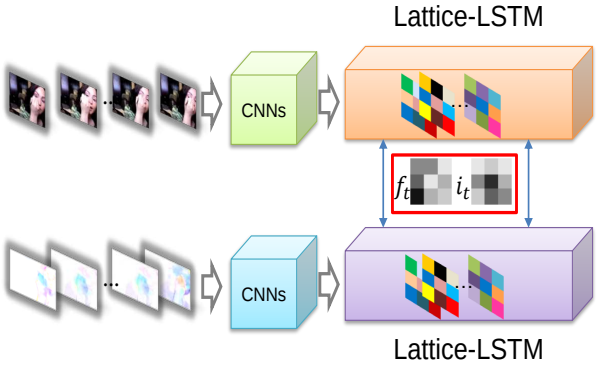

Figure 1. The mechanism of Lattice-LSTM (L^{2}STM). f_t and i_t represent the forget and input gate patterns at time t, respectively. Two technical contributions are illustrated here: Firstly, more motions, particularly complex ones, can be depicted using a spatially varying local superposition within the memory cell. Secondly, input and forget gates are multi-modal trained using RGB and corresponding optical flow. They will serve as the motion attention mask controlling the entering/leaving dynamics from the memory cell. Even more motion dynamics are generated from the memory cell, but only the useful ones are preserved for representation.

Following the success of Convolutional Neural Networks (CNNs) on computer vision tasks such as image classification, researchers have achieved remarkable performance on public human action recognition datasets. Deep learning has been shown to have an extraordinary ability to extract powerful representations. In order to handle videos, CNNs have been applied to videos on a per-frame basis, followed by simple or strategic pooling across the temporal domain. However, CNNs only consider appearance features frame by frame without any consideration of interac-

tions across the temporal domain. Meanwhile, the traditional Long Short-Term Memory (LSTM) with matrix multiplications does not take spatial layout into consideration. ConvLSTM [35] and VideoLSTM [19] address this limitation by applying convolutional operations on sequences of images or feature maps. These models tend to work well for actions with little movement, which we refer to as stationary motions. However, the limited modeling capacity of these architectures causes the network to struggle with long-term and complex motions, as verified by our experiments. An accurate action recognition should (1) have a high capacity for learning and capturing as many motion dynamics as possible, and (2) when an action appears in sequential images, the neurons should properly decide what kind of spatio-temporal dynamics should be encoded into the memory for distinguishing actions.

In this paper, we propose a novel two-stream LSTM architecture named Lattice-LSTM (L^2STM). L^2STM extends LSTM by learning independent memory cell transitions for individual spatial locations with control gates that are shared between RGB and optical flow streams. It can more effectively address the non-stationary issue of long-term motion dynamics without significantly increasing the model complexity. At the same time, the control gates are trained using multiple modalities, RGB images and optical flow, in order to better control the information entering/leaving the memory cell. We summarize the main contributions of this paper as follows:

- Our derivation of the recurrent relation confirms our assumption that the current LSTM architecture, including LSTM, ConvLSTM and VideoLSTM, cannot accurately model the long-term and complex motions within videos. A novel LSTM, called Lattice-LSTM (L^2STM), that can greatly enhance the capacity of the memory cell to learn motion dynamics is proposed. Extensive experiments have been conducted to verify the effectiveness of the proposed L^2STM.

- To leverage both RGB and the corresponding optical flow information, the two input gates and the two forget gates of our two-stream L^2STM are shared and trained by both modalities, which achieves better control of the dynamics passing through the memory cell.

- We propose a new sampling strategy to enable RNNs to learn long- and short-term temporal information from randomly extracted clips. Compared with consecutive frame sampling and sampling with strides, our sampling method provides data augmentation along the temporal domain, which helps the learning of recurrent dynamics.

- We present a novel architecture to enhance, encode and decide motion dynamics for action recognition. Experiments on the benchmark UCF-101 and HMDB-51 datasets show that our proposed methods outperform all existing ones that are based on LSTM and/or CNNs of similar model complexities.

2. Related Work

Following the great success of CNNs for image classification [17], image segmentation [21], image retrieval [1] and other computer vision tasks, deep learning algorithms have been used in video based human action recognition as well. Karpathy et al. [16] directly apply CNNs to multiple frames in each sequence and obtain the temporal relations by pooling using single, late, early and slow fusion; however, the results of this scheme are just marginally better than those of a single frame baseline, indicating that motion features are difficult to obtain by simply and directly pooling spatial features from CNNs.

In light of this, Convolutional 3D (C3D) is proposed in [31] to learn 3D convolution kernels in both space and time based on a straightforward extension of the established 2D CNNs. However, when filtering the video clips using 3D kernels, C3D only covers a short range of the sequence. Simonyan et al. [15] incorporate motion information by training another neural network on optical flow [37]. Taking advantage of the appearance and flow features, the accuracy of action recognition is significantly boosted, even by simply fusing probability scores. Since optical flow contains only short-term motion information, adding it does not enable CNNs to learn long-term motion transitions.

Several attempts have been made to obtain a better combination of appearance and motion in order to improve recognition accuracy. Lin et al. [30] try to extract the spatial-temporal features using a sequential procedure, namely, 2D spatial information extraction followed by 1D temporal information extraction. This end-to-end system considers the short (frame difference) and long (RGB frames with strides) motion patterns and achieves good performance. Feichtenhofer et al. [9] study a number of ways of fusing CNN towers both spatially and temporally in order to take advantage of this spatio-temporal information from the appearance and optical flow networks. They propose a novel architecture [8] generalizing residual networks (ResNets) [12] to the spatio-temporal domain as well. However, a CNN based method cannot accurately model the dynamics by simply averaging the scores across the time domain, even if the appearance features already achieve remarkable performance on other computer vision tasks.

In order to model the dynamics between frames, recurrent neural networks (RNNs), particularly long short-term memory (LSTM), have been considered for video based human action recognition. LSTM units, first proposed in [13], are recurrent modules which have the capability of learning long-term dependencies using a hidden state augmented

with nonlinear mechanisms to allow the state to propagate without modification. LSTMs use multiplicative gates to control access to the error signal propagating through the networks, alleviating the short memory in RNNs [11]. [35] extends LSTM to ConvLSTM, which considers the neighboring pixels' relations in the spatial domain. ConvLSTM can learn spatial patterns along the temporal domain.

LSTMs have achieved remarkable performance on language modeling [36], but their performance on video action recognition still lags. [2], [23] and [5] propose LSTMs that explicitly model short snippets of ConvNet activations. Ng et al. [23] demonstrate that two-stream LSTMs outperform improved dense trajectories (iDT) [32] and two-stream CNNs [15], although they need to pre-train their architecture on one million sports videos. Srivastava et al. [29] propose an interesting LSTM based unsupervised training method using an encoder LSTM to map an input sequence into a fixed-length representation, which is then decoded using single or multiple decoder LSTMs to perform reconstruction of the input sequence or prediction of the future sequence. Then they fine-tune this unsupervised pre-training LSTM to adapt human action recognition tasks. [22] proposes to train an LSTM that is regularized using the output of another encoder LSTM (RLSTM-g3) grounded on 3D human-skeleton training data, while [2] proposes a two-step learning method in which 3D features are extracted using CNNs, and then an RNN is trained to classify each sequence by considering the temporal evolution of the learned features. Finally, [5] proposes a class of end-to-end trainable CNN and/or RNN architectures to handle different kinds of inputs. However, the traditional LSTM with matrix multiplications is adopted without exploiting the spatial correlations in the video frames. Directly applying the LSTM models to videos based action recognition is not satisfactory since the spatial correlations and motion dynamics between the frames are not well presented.

To address this, in VideoLSTM [19] convolutions are hardwired in the soft-Attention LSTM [26]. By stacking another RNN for motion modeling, an enhanced version of attention model is assembled. The input of ConvLSTM, X_t, becomes $A_t \odot X_t$ by element-wise multiplying the attention map A_t. However, this complex architecture does not bring significant performance improvement. In fact, the performance is highly dependent on the iDT features [32]. In other words, VideoLSTM does not characterize the motion dynamics well, even when several attention models are stacked within/on the LSTM.

3. Models and Algorithm

3.1. Revisiting the problem of modeling in RNNs

Given a video sequence of length T, suppose at each time t with $t = 1, \ldots, T$, we can extract at any spatial location $x \in \Omega$ a one-dimensional motion signal $\mathbf{h}_t^x \in \mathbb{R}^L$ of a short length L, where Ω is the 2D spatial domain of the video. To characterize such motion signals, one may use a set of linear filters $\{\Psi_i \in \mathbb{R}^L\}_{i=1}^n$ (e.g., wavelets), and compute their pixel-wise features at time t as $\{\Psi_i \mathbf{h}_t^x\}_{i=1}^n$. This is similar to image processing, where a set of pre-defined 2D filters can be used to produce feature maps from an input image via convolution, due to the stationary property of natural images. We also know from the image processing literature that such 2D filters can be learned from a set of training images, so that statistics specific to these training images can be learned to better process images of similar kind [7]. The learned 2D filters are typically in the format of 5×5 patches. When applied to video processing, this translates to learning a set of filters $\{\Phi_i \in \mathbb{R}^L\}_{i=1}^n$ that are adapted to short-length motion dynamics of training videos.

However, different from image processing, in many video processing applications, including human action recognition, long-term motion dynamics are important for characterizing different motion patterns.

Directly learning filters for long-term motion patterns would make both the size L and the number n of filters $\{\Phi_i \in \mathbb{R}^L\}_{i=1}^n$ prohibitively large. This however creates learning difficulties due to increased model complexity that are similar to the difficulties in image processing when learning 2D filters larger than 5×5 patches.

To resolve this issue, one may still learn short-length motion filters and apply them sequentially across the sequence for $t = 1, \ldots, T$. However, this scheme cannot take full advantage of modern video processing models such as RNNs/LSTMs. In order to analyze how to resolve this issue when using RNNs/LSTMs, we first investigate how they learn to process video sequences. RNNs use either a hidden layer or a memory cell to learn a time varying set of states which models the underlying dynamics of the input sequence. To better derive and present the limits of RNN, we adopt the general RNN definition without nonlinearities instead of LSTM, since the conclusion from the derivation is the same. We write the recurrent formula of RNNs as follows, where the nonlinearities, control gates, and memory operations are ignored for simplicity:

$$H_t = W_H H_{t-1} + W_X X_t, \tag{1}$$

where $\mathbf{X} = \{X_1, X_2, \cdots, X_t\}$ is the input sequence, X_t is an element of the sequence at time t, $\mathbf{H} = \{H_1, H_2, \cdots, H_t\}$ is the hidden states, H_t is the output of one hidden state at time t, and W_H and W_X are the transition weights. Since the weights are shared across different iterations, the output at the next time step $t+1$ is

$$H_{t+1} = W_H^2 H_{t-1} + W_H W_X X_t + W_X X_{t+1}. \tag{2}$$

Iteratively, when the time step becomes $t + \tau$, the recurrent formula becomes

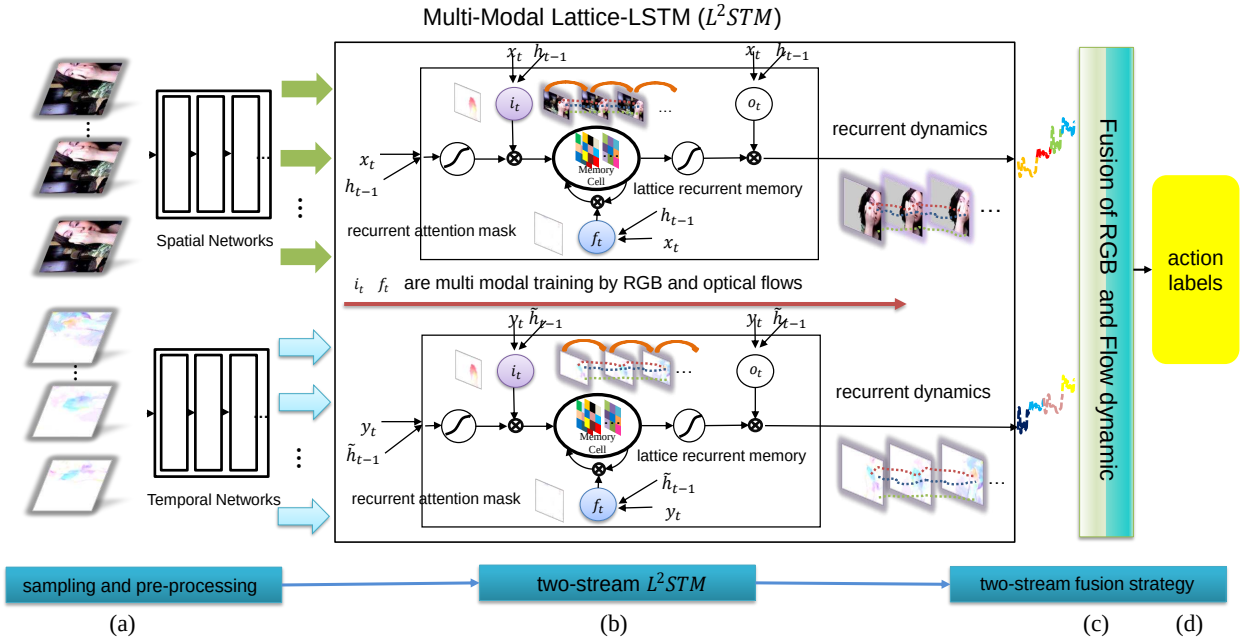

Figure 2. Illustration of the proposed Lattice-LSTM (L^{2}STM). (a) depicts the sampling and pre-processing procedure. A set of video clips (RGB and flow) are passing through CNNs. (b) presents the core functions of the two-stream L^{2}STM. Local space-variant superposition operations are applied on the cell memory, enlarging the LSTM's capability to learn the different motion patterns, and the optical flow are fed into the systems simultaneously in order to learn the input and forget gates using multi-modal training. In order to vividly illustrate the system, instead of feature maps, the raw inputs (RGB and corresponding flow) are presented here. (c) is the vectorized learned motion dynamics from L^{2}STM and weighted average fusion. (d) represents the final classification procedure. The whole system is end-to-end trainable.

$$
\begin{aligned}
H_{t+\tau} &= W_H^{\tau+1} H_{t-1} + W_H^{\tau} W_X X_t + \cdots + W_X X_{t+\tau} \\
H_{t+\tau} &= \sum_{j=0}^{\tau} W_H^{\tau-j} W_X X_{t+j} + W_H^{\tau+1} H_{t-1} \\
&= \begin{bmatrix} W_H^{\tau} & W_H^{\tau-1} & \dots & 1 \end{bmatrix} \begin{bmatrix} W_X X_t \\ W_X X_{t+1} \\ \vdots \\ W_X X_{t+\tau} \end{bmatrix} + W_H^{\tau+1} H_{t-1}.
\end{aligned} \tag{3}
$$

For ease of analysis, we simplify Equ. 3 as

$$H_{t+\tau} = f\left(X_{t:t+\tau}, H_{t-1}\right),$$

where $X_{t:t+\tau}$ denotes $\left[X_t^\top, \dots, X_{t+\tau}^\top\right]^\top$ compactly, $f(\cdot)$ is a linear function to be learned to achieve state transition in LSTM, and $X_{t:t+\tau}$ is a high-dimensional input signal, since it is formed by concatenating all the appearance features from time t to time $t+\tau$. Note in the formula, W_H are the only time step related parameters.

It is generally difficult to learn an accurate mapping function f due to its high dimensionality. When $X_{t:t+\tau}$ represents a sequence of video frames, a natural choice is to learn filters that process $X_{t:t+\tau}$ at the local patch level. Indeed, ConvLSTM [35] learns the same set of patch-level filters for different spatial locations of videos. The form of Equ. 3 (i.e., polynomials of the basic filters W_H) reduces the search space of high-dimensional filters to be learned, and removes some possible patterns that the learned filters should characterize. We would like to increase the model capacity to better characterize complex motion patterns. In machine learning, a traditional way to learn a function with a higher model capacity is to partition the feature space into local cells and learn separate mappings in each local cell. In this work, we follow this idea and propose to partition the high-dimensional feature space into local cells based on their positions in the spatial domain. Thus the difficulty of learning mapping functions in high-dimensional space is reduced to learning mappings for each pixel separately.

3.2. Lattice-LSTM (L^{2}STM)

In this section, we propose Lattice-LSTM (L^{2}STM) to solve the issues presented in the above analysis and derivation. In order to enhance the modeling of motion dynamics, the newly generated cell memory $\widetilde{C}_t$ at time t of L^{2}STM is formulated as

$$\widetilde{C}_t = tanh(W_{xc} * X_t + W_{hc} \textcircled{L} H_{t-1}), \tag{4}$$

where W_{xc} and W_{hc} are the distinct weights for the input and hidden state, $*$ denotes the conventional convolution operation, and $\textcircled{L}$ denotes a local superposition summation..

The local superposition is defined as follows:

$$W_{hc} \textcircled{L} H_{t-1} = \forall_{k,i,j}\Big(\sum_{l,m,n} W_{hc}(l,i,j,k,m,n) H_t(l, i+m-1, j+n-1)\Big), \quad (5)$$

where (i, j) indicates the position, m and n index position in the kernel, and l and k index the input and output feature maps. Intuitively, this is performing a local filtering operation with different filters at different spatial locations.

These local filters, which are initialized independently, are applied on each memory cell pixel in order to learn the different temporal patterns at different pixels. According to Equ. 3, we only apply the local superposition on the hidden transition of cell memory within the recurrence and the other linear combinations are convolutional. Therefore, W_{hc} has a larger capacity than W_{xc}. Since the memory cell stores the information obtained over time, the local superposition will enhance the capability of preserving dynamics in the memory cell and model spatial nonhomogeneity in the long term dependency. Here the term 'lattice' is used to vividly indicate the local superposition. At the same time, the input and forget gates which control the memory cell are

$$\begin{aligned} i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1}) \\ f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1}), \end{aligned} \quad (6)$$

where W_{xi}, W_{hi} and W_{xf}, W_{hf} are distinct weights for the input and forget gates respectively. The updated memory cell C_t can be represented as

$$C_t = f_t \circ C_{t-1} + i_t \circ \widetilde{C}_t, \quad (7)$$

where C_{t-1} is the previous memory cell and '$\circ$' denotes the Hadamard product. The input gates and forget gates together determine the amount of dynamic information entering/leaving the memory cell. Since the input gates and forget gates are important for controlling the memory cell, during the learning of $\{W_{xi}, W_{hi}, W_{xf}, W_{hf}\}$, we use a multi-modal learning procedure. RGB frames of the video and corresponding optical flow are fed into the system simultaneously to learn the shared input gates and the shared forget gates for the two-stream architecture. Therefore the gradient update for the input gate and forget gate should be

$$\begin{aligned} W_{\cdot i} &:= W_{\cdot i} - \eta(\bigtriangledown Q^{RGB}(W_{\cdot i}) + \bigtriangledown Q^{Flow}(W_{\cdot i})) \\ W_{\cdot f} &:= W_{\cdot f} - \eta(\bigtriangledown Q(^{RGB}W_{\cdot f}) + \bigtriangledown Q^{Flow}(W_{\cdot f})), \end{aligned} \quad (8)$$

where $Q^{RGB}(\cdot)$ and $Q^{Flow}(\cdot)$ are the estimated loss functions passing to the memory cell from the RGB inputs and corresponding flow, $\cdot$ denotes the input and hidden state and η is the learning rate. With multi-modal learning, the controlled gates can well adjust the information entering/leaving the cell memory. Finally, the output of one iteration of L^{2}STM is

$$\begin{aligned} o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1}) \\ H_t &= o_t \circ tanh(C_t), \end{aligned} \quad (9)$$

where W_{xo} and W_{ho} are distinct weights for the output gate.

Because we consider the spatial variability in the temporal modeling, the representations of motion dynamics have been greatly enhanced due to the local superposition. Therefore, more motion patterns existing in videos can be depicted and captured. Meanwhile, the multi-modal-trained control gates provide motion masks for the memory cell in order to generate distinguishing representations. Batch normalization [14] is applied after each spatial convolutional operation.

Figure 3. Frames extracted from an ice dancing video. Sampling using different strides makes different episodes. Episode 1 is almost stationary, Episode 2 becomes time varying and Episode 3 has dramatic change.

3.3. Long-Short Term Sampling

Due to computational limitations, we cannot feed all sequential inputs into the system and train them, so sampling is necessary for training the videos end-to-end. As shown in Fig. 3, even from the same video sequence, different sampling methods can generate different episodes. In this section, we implement a new sampling strategy in order to make the recurrent networks learn more reasonable 'long' and 'short' representations.

Inspired by [30], we apply temporal data augmentation for training the network. Suppose we are given a video sequence $\mathbf{V} \in \mathbb{R}^{m_x \times m_y \times m_t}$, where (m_x, m_y) are the spatial dimensions of the video and m_t is the number of frames in the video sequence. Given this video sequence and a patch size of $l_x \times l_y$ (such that $l_x < m_x, l_y < m_y$, $l_t < m_t$), we can sample a set of n video clips $S = \{\mathbf{V}_{clip}^{(1)}, \mathbf{V}_{clip}^{(2)}, \ldots, \mathbf{V}_{clip}^{(n)}\}$ which constitute a single forward pass through the unrolled model (with n steps) as follows. To sample a video clip $\mathbf{V}_{clip} \in \mathbb{R}^{l_x \times l_y \times l_t}$, we sample $l_x \times l_y$ crops of l_t frames with a temporal stride of s_t, and we use a spacing of s_s frames between each video clip in the sequence to form our set S. Note that for each individual sequence, we sample a new crop center, a new stride s_s such that $0 \leq s_s \leq s_{max}$, where s_{max} is the maximum possible stride, and a new stride s_t such that $0 \leq s_t \leq s_s$.

Depending on our sampling of the parameters, we can select a sequence of clips S covering a very short portion of the video or a sequence of clips S covering a long portion of the video. In other words, this kind of sampling can provide

'short'-term clips, which cover consecutive frames ($s_t = 1$ and s_s is the minimum stride) and in which the movement is stationary, and 'long'-term clips, which cover the whole video ($s_t = s_s$ and $s_s = s_{max}$) and in which the movement changes a lot. All LSTM models can benefit from this sampling formulation. We see about a 2% gain compared with the traditional sampling that uses a fixed stride and randomly extracted clips, and we have adopted our temporal data augmentation in all of our experiments.

3.4. Learning Architecture

The whole architecture is presented in Fig. 2. In order to illustrate the concept clearly, instead of using feature maps from a CNN, in the figure, we use raw RGB and optical flow images to illustrate the memory cell, input, forget and output gates. Local superposition operations are applied on the cell memory to form the lattice recurrent memory, enhancing the LSTM's ability to learn complex motion patterns and addressing the long-term dependency issue.

As illustrated in Fig. 2, we adopt the two-stream framework in which the optical flow is fed as an additional modality to further compensate and improve the prediction of RGB inputs. Note that video clips, each composed of several frames sampled from a video, are fed through the pre-trained spatial and temporal networks at each time step to extract the high-level feature maps. Compared to traditional two-stream frameworks which train each stream independently (on the corresponding input modality), combining the scores together at the end, our model feeds in the RGB and flow information simultaneously in order to learn the shared input gates and shared forget gates that form the recurrent attention mask for the memory cell. Note that only the input and forget gates are shared between the RGB and flow, while the others are learned independently.

This multi-modal learning procedure allows the input and forget gates to leverage both appearance and dynamic information. The learned recurrent attention mask of the input and forget gates, which can further regularize enhanced dynamics from the memory cell, is formed to control the entering/leaving dynamics from the memory cell. After regularization of the input and forget gates, the output should be motion features which can capture very complex dynamics. Each memory cell and output gate is learned independently of the other stream in order to optimize the features to each modality. The final prediction is the weighted average of the two outputs from the sequential RGB and flow. The video sequence is recognized as the action category corresponding to the highest probability.

4. Experiments

Experiments are mainly conducted on two action recognition benchmark datasets, namely UCF-101 [28] and HMDB-51 [18], which are currently the largest and most challenging annotated action recognition datasets.

UCF-101 [28] is composed of realistic web videos with various camera motions and illuminations. It has more than 13K videos, with an average length of 180 frames per video categorized into 101 actions, ranging from daily life activities to unusual sports.

HMDB-51 [18] has a total of 6766 videos organized as 51 distinct action categories. This dataset is more challenging than others because it contains scenes with more complex backgrounds. It has similar scenes in different categories, and it has a small number of training videos.

Both UCF-101 and HMDB-51 have three split settings to separate the dataset into training and testing videos. We report the mean classification accuracy over these splits.

4.1. Implementation Details

We choose part of VGG16 [27], which consists of 13 convolutional layers, as our CNN feature extractor for the RGB and optical flow images. The spatial and temporal networks are pre-trained on ImageNet [4] and finetuned on the corresponding human action recognition datasets. L^2STM is trained from scratch. The input of the optical flow network is stacked optical flow images, each of which has two channels (the horizontal and vertical flow fields). The optical flow [37] is computed from every two adjacent frames. We linearly rescale the values of the flow fields to [0, 255]. We feed in a stack of five RGB frames to the spatial network and a stack of five corresponding flow frames to the temporal network. Since the pre-trained VGG16 network uses inputs of different dimensions, we re-train the first layer of both networks. Then the features from the last pooling layer (e.g. pool5) of each VGG16 network are fed into L^2STM.

Our implementation contains 8 unrolled time steps. Therefore, eight video clips, composed of 5 frames each, are regularly extracted from the videos with the random strides (as described in the long-short term sampling) for each L^2STM training step using back-propagation through time (BPTT). Within L^2STM, all convolutional kernels and superposition kernels are 3×3. We use mini-batch stochastic gradient descent (SGD) to train the network with an initial learning rate of 0.001, a momentum of 0.9, and a weight decay 5×10^{-4}. We decay the learning rate when it is saturated and adopt two-step training. That is, we first fix the CNN weights and train the L^2STM only. When the accuracy of the L^2STM on the validation videos stops increasing, we fine-tune the whole network using a smaller learning rate. For data augmentation, we use color jittering, as described in [17], horizontal flipping, and random cropping for the spatial network and horizontal flipping and random cropping for the temporal network. When testing, more video clips are fed, and we accumulate different probabilities from different length sequences with different strides in order to

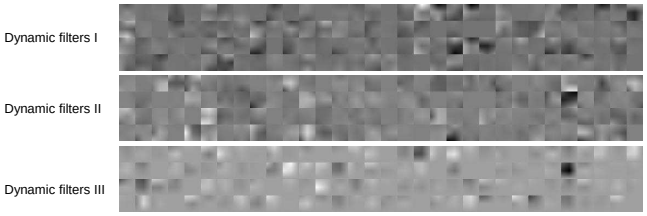

Figure 4. The visualization of several leaned kernels in the local superposition operator. For each filter, we visualize the kernels of 128 channels. Note that we rescale the 3×3 filters to make the visualization more pleasing.

combine 'long' and 'short' predictions.

Video clips sampled from videos are fed into the system and scores are extracted from different time steps of L^{2}STM. The final scores are the combination of these scores $[S_1, S_2, S_3, \cdots S_t]$, where S is the output score and t is the time step with different strides. The final scores should be the average of these outputs with different strides, that is, $S = mean(S_1, S_2, S_3, \cdots, S_t)$. Real-time action recognition can be achieved when the video is streaming sequentially. The video streaming can be fed into the system sequentially, and the prediction accuracy can be accumulated when more clips are fed in.

4.2. Visualization

Several learned local superposition kernels are shown in Fig. 4. Due to the page limit, we present three sets of learned position-varying filters, each of which has 128 channels. We can clearly see that these sets of filters learn different properties, further verifying our previous assumptions; different superposition may represent the different semantic meaning during actions.

To visually verify the effect of L^{2}STM, we use BPTT to visualize saliency regions for specific video sequences, i.e., back-propagating the neuron of that action category in the classifier layer to the input image domain and plotting the corresponding gradient. As shown in Fig. 5, our L^{2}STM can catch the most dynamic regions, while ConvLSTM cannot for the sake of treating all movements the same. For instance, in Sequence I, L^{2}STM can provide accurate motion region detection, even when the movement is very small (scissors moving around a head), while ConvLSTM can not. What is more, in Sequence III, L^{2}STM can detect a leg movement and a lifting movement, while ConvLSTM only focuses on the leg movement. Owing to the local superposition operation on the cell memory, our proposed model can provide more dynamics through recurrence. At the same time, thanks to the multi-modal learning, the input and forget gates can control the dynamic entering/leaving well, so useless dynamics are not present in the salient regions.

4.3. Effect on videos with complex movements

We specifically evaluated our model on videos with complex movements. Besides its 101 categories, UCF-101 has coarse definitions which divide the videos into human and object interaction, human and human interaction and sports. We evaluate the performance on these coarse categories, and as reported in Table. 1. The upper part of the table lists the performance on the coarse category. Compared with ConvLSTM (note here the only difference between ConvLSTM and L^{2}STM is the network architecture), our model performs much better on all coarse categories, particularly on human and object interaction $\uparrow$ 10.7%, human and human interaction $\uparrow$ 6.3% and body motion only $\uparrow$ 0.5%. Since our model can well handle the complex movements while ConvLSTM may not, it performs much better on the human interacting actions. On the other hand, they perform similarly on the body motion only actions since the movements are simple in this category, which verifies our hypothesis. The lower part of the table shows the performance for specific categories. To provide an overview, the sample frames from each coarse category can be found from the dataset website. We can see from the table that the more complex the videos are, the greater the percentage improvement there is. In the pizza tossing category in UCF-101, the movements of the person and pizza are complex and fast, and we see a 51.5% improvement.

Table 1. Performance evaluation on videos of split 1 of UCF-101 with complex movements

Data Types	ConvLSTM	L^{2}STM	Gain
Human-Object Interaction	76.0	86.7	$\uparrow$ **10.7**
Human-Human Interaction	89.1	95.4	$\uparrow$ **6.3**
Body Motion Only	88.1	88.6	$\uparrow$ **0.5**
Pizza Tossing	21.2	72.7	$\uparrow$ **51.5**
Mixing Batter	55.6	86.7	$\uparrow$ **31.1**
Playing Dhol	81.6	100	$\uparrow$ **18.4**
Salsa Spins	86.0	100	$\uparrow$ **4.0**
Ice Dancing	97.8	100	$\uparrow$ **2.2**

4.4. Comparison with LSTM-like architecture

In Table. 2, we list the performance of all LSTM variants and our main component for action recognition of spatial and temporal networks on split 1 of UCF-101. Our proposed method significantly outperforms the variants on each network. From the component analysis, we find that the local superposition has a more significant impact on the temporal networks than spatial networks, since more useful features are extracted from flow using L^{2}STM. Additionally, the improvement of the shared control gates on the spatial networks is higher than on the temporal networks, indicating that spatial networks benefit more from multi-modal training.

In Table. 3, we list all state-of-the-art methods which use an LSTM-like architecture for action recognition tasks.

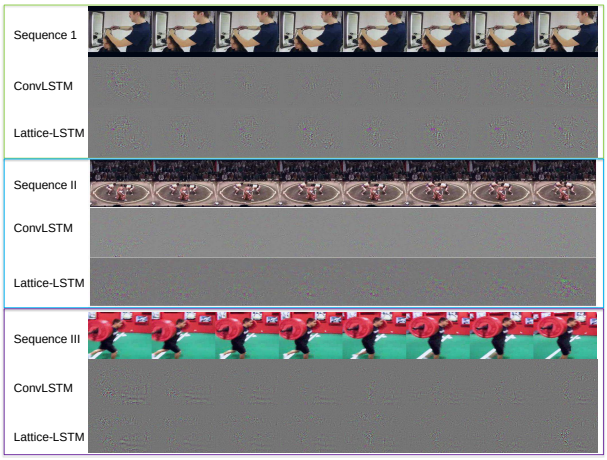

Figure 5. Saliency map comparison. From top to bottom is the original video sequence, ConvLSTM and our L^{2}STM. As shown in the figure, L^{2}STM can accurately detect the interactive and moving points, while ConvLSTM can not. This visualization looks better in color and high resolution.

Table 2. L^{2}STM components analysis on split 1 of UCF-101

Method	Spatial Networks	Temporal Networks
LSTM [19]	77.5	78.3
ALSTM [19]	77.0	79.5
VideoLSTM [19]	79.6	82.1
ConvLSTM	77.6	79.1
+long-short term sampling	80.4	82.3
+locally superpostion	82.2	86.9
+shared controlling gates	83.2	87.4

Table 3. State-of-the-art comparison with LSTM-like architectures

Method	Pre-training		Networks		UCF101	HMDB51
	ImageNet	1M Sports	VGG-M	VGG16		
LRCN[5]	✓	-	✓	-	82.9	-
TwoLSTM[23]	✓	✓	-	✓	88.3	-
ALSTM[26]	✓	-	-	✓	77	41.3
RLSTM-g3[22]	✓	-	-	✓	86.9	55.3
UnsuperLSTM[29]	✓	✓	-	✓	84.3	44.0
VideoLSTM[19]	✓	-	-	✓	89.2	56.4
L^{2}STM	✓	-	-	✓	**93.6**	**66.2**

In order to present the comparison completely and clearly, we elaborate on some factors, such as, pre-training type and network architecture. From the table, our proposed method clearly performs the best amongst the LSTM-like architectures, even though we do not have 1M sports pre-training.

4.5. Comparison with the-state-of-the-art

The temporal information can be well modeled using L^{2}STM. In addition to the LSTM-like comparison in Table 3, another state-of-the-art algorithm comparison is presented in Table 4, where we compare our method with traditional approaches such as improved trajectories (iDTs) [32] and deep learning architectures such as two-stream networks [15], factorized spatio-temporal convolutional networks (F$_{ST}$CN) [30] (which use VGG-M [38] as a deep feature extractor), 3D convolutional networks (C3D) [31], trajectory-pooled deep convolutional descriptors (TDD) [33], and spatio-temporal fusion CNNs [9], which use a VGG16 deep architecture on the UCF-101[28] and HMDB-51[18] datasets. We also include some complex network-based methods: ST-ResNet [8] and TSN [34]. Compared with the traditional hand-crafted methods and deep learning methods, our proposed L^{2}STM achieves better performance.

Table 4. Mean accuracy on the UCF-101 and HMDB-51

Model	Method	UCF-101	HMDB-51
Traditional	iDT + FV [32]	85.9	57.2
	iDT + HSV [25]	87.9	61.1
	VideoDarwin [10]	-	63.7
	MPR [24]	-	65.5
Deep	EMV-CNN [39]	86.4	-
	Two Stream [15]	88.0	59.4
	$F_{ST}CN$ [30]	88.1	59.1
Very Deep	C3D (3 nets) [31]	85.2	-
	VideoLSTM[19]	89.2	56.4
	TDD+FV [33]	90.3	63.2
	Fusion [9]	92.5	65.4
Ours	L^{2}STM	**93.6**	**66.2**
Complex *	ST-ResNet [8]	93.4	66.4
	TSN [34]	94	68.5

* Take advantages of complex deep architecture; their spatial and temporal networks (baseline) are about 1% to 2% better than ours.

5. Conclusion

In order to solve the existing problems in action recognition, L^{2}STM is proposed. This method applies local spatially varying superposition operations for the memory cell and multi-modal trained control gates (input and forget) using the RGB and (corresponding) flow, sampled using long-short term sampling. A huge gain is observed compared with other LSTM-like architectures for action recognition tasks. This model achieves state-of-the-art performance, even compared with more complex networks. High level feature maps from CNNs provide several semantic meanings, which can be modeled using different filters across time. In this paper, we learn filters from space-variant superposition. An extension would be to adaptively group these semantic meanings based on videos, without learning them on each location. We hope this work can shed light on LSTM architectures for video analysis.

Acknowledgements: We gratefully acknowledge the support of the Hong Kong University of Science and Technology, MURI (1186514-1-TBCJE), ONR (1196026-1-TDVWE); NISSAN (1188371-1-UDARQ. This work also used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562.

References

[1] A. Babenko, A. Slesarev, A. Chigorin, and V. S. Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014. 2

[2] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *Proceedings of the Second International Conference on Human Behavior Unterstanding*, 2011. 3

[3] O. Boiman and M. Irani. Detecting irregularities in images and in video. *IJCV*, 74(1):17–31, Aug 2007. 1

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: a large-scale hierarchical image database. In *CVPR*, 2009. 6

[5] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 3, 8

[6] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *ICCV*, 2003. 1

[7] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Transaction on Image Processing*, 15(12):3736–3745, Dec 2006. 3

[8] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal residual networks for video action recognition. *CoRR*, abs/1611.02155, 2016. 2, 8

[9] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 2, 8

[10] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, 2015. 8

[11] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal Machine Learning Research*, 3:115–143, Mar. 2003. 3

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

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

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[15] A. Z. K. Simonyan. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 3, 8

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

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NPIS*, 2012. 2, 6

[18] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 1, 6, 8

[19] Z. Li, E. Gavves, M. Jain, and C. G. M. Snoek. VideoLSTM convolves, attends and flows for action recognition. volume abs/1607.01794, 2016. 2, 3, 8

[20] W. Lin, M.-T. Sun, R. Poovendran, and Z. Zhang. Human activity recognition for video surveillance. In *ISCAS*, 2008. 1

[21] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

[22] B. Mahasseni and S. Todorovic. Regularizing long short term memory with 3D human-skeleton sequences for action recognition. In *CVPR*, 2016. 3, 8

[23] 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. 3, 8

[24] B. Ni, P. Moulin, X. Yang, and S. Yan. Motion part regularization: Improving action recognition via trajectory selection. In *CVPR*, June 2015. 8

[25] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 150:109–125, 2016. 8

[26] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. *CoRR*, 2015. 3, 8

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 6

[28] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. 1, 6, 8

[29] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015. 3, 8

[30] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015. 2, 5, 8

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

[32] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 3, 8

[33] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 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 *ECCV*, 2016. 8

[35] 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 *NIPS*, 2015. 2, 3, 4

[36] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig. Achieving human parity in conversational speech recognition. *CoRR*, abs/1610.05256, 2016. 3

[37] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-l1 optical flow. In *Annunal Symp. German Association Pattern Recognition*, pages 214–223, 2007. 2, 6

[38] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 8

[39] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with enhanced motion vector CNNs. In *CVPR*, 2016. 8