Deep Multi-Kernel Convolutional LSTM Networks and an Attention-Based Mechanism for Videos

Sebastian Agethen, Winston H. Hsu

***Abstract*—Action recognition greatly benefits motion understanding in video analysis. Recurrent networks such as long short-term memory (LSTM) networks are a popular choice for motion-aware sequence learning tasks. Recently, a convolutional extension of LSTM was proposed, in which input-to-hidden and hidden-to-hidden transitions are modeled through convolution with a single kernel. This implies an unavoidable trade-off between effectiveness and efficiency. Herein, we propose a new enhancement to convolutional LSTM networks that supports accommodation of multiple convolutional kernels and layers. This resembles a Network-in-LSTM approach, which improves upon the aforementioned concern. In addition, we propose an attention-based mechanism that is specifically designed for our multi-kernel extension. We evaluated our proposed extensions in a supervised classification setting on the UCF-101 and Sports-1M datasets, with the findings showing that our enhancements improve accuracy. We also undertook qualitative analysis to reveal the characteristics of our system and the convolutional LSTM baseline.**

I. Introduction

ACTION recognition is a challenging-yet-essential task in modern computer vision that is typically performed on video clips. Videos are now frequently encountered in our everyday lives on social media platforms such as Instagram, Facebook, and YouTube. The amount of video data is indeed vast; in 2015, 500 hours of videos were uploaded to YouTube every minute[1]. These quantities mean that automatic processing by machines is necessary, which is why action recognition is too. Many applications can benefit from action recognition; for example, autonomous driving, security and surveillance, and sports analysis.

Unlike static images, videos have an inherently spatiotemporal nature. The motion of subjects, such as persons, animals, or objects, carries significant information on the current action. By observing and exploiting motion, we can improve our understanding of an action over simple classification of static background features.

Particularly successful attempts at action recognition have been made using deep learning [1][2][3][4]. Two approaches are common: deep convolutional networks (DCNs) have achieved impressive results, but they are unaware of temporal dependencies (i.e., reordering frames has little effect), and therefore they perform poorly when processing motion information. A second choice is recurrent networks, among which long short-term memory [5] (LSTM) in particular has been used frequently in recent works. LSTM networks are sensitive to reordering of frames, and they can therefore be used for sequence learning.

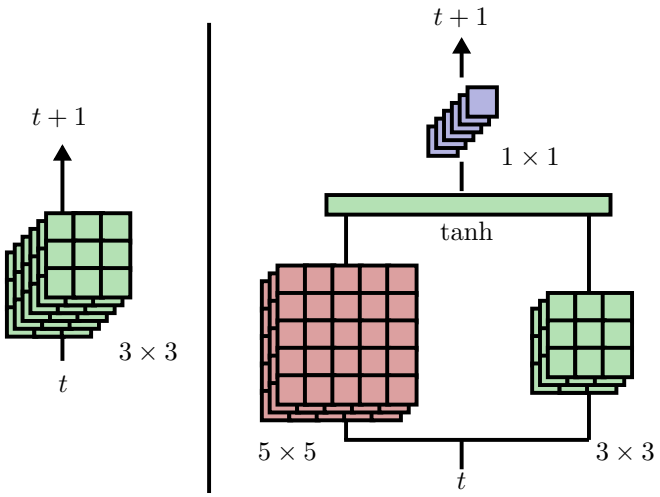


Fig. 1. Employing multiple kernels in ConvLSTM. **Left:** Traditional ConvLSTM with single kernel input-to-hidden and hidden-to-hidden transitions, here 3×3. **Right:** Exemplary multi-kernel configuration. We propose replacing the single kernel with a set of kernels. The total number of channels remains unchanged. In addition, a deep network may be used instead of a single layer.

Several variations of LSTM exist. Most recently, convolutional LSTM cells [6] (ConvLSTM) have been proposed, which add the benefits of convolution to an LSTM unit, namely *spatial invariance*; for example, given a two-dimensional image, a convolutional filter can correctly respond to an object no matter at which location it is placed in the image. This is not the case with *fully connected* units, which were previously used in LSTM. In the following, we follow the notation in [6] and refer to traditional LSTM as FC-LSTM, whereas we refer to convolutional LSTM as ConvLSTM. Our work is greatly based on ConvLSTM, and we discuss it in more detail in Section III.

In this paper, we reason that using a single kernel as in ConvLSTM is not optimal. First, whether a single optimal kernel size exists is unclear. Small kernels may not be able to build a causal connection between a phenomenon and a related previous one, as is the case when observing an actor's motion or with the simple example of a fast-moving object. Larger kernels, however, are very slow to compute and lack spatial invariance. We exhibit this in an example in Section III. Second, we note a lack of depth, if we consider the transition

Sebastian Agethen and Winston H. Hsu are affiliated with National Taiwan University.

Email: d01944015@ntu.edu.tw; whsu@ntu.edu.tw

[1]According to a statement by YouTube CEO Susan Wojcicki at VidCon 2015.

at times $t \rightarrow t+1$: The convolutional operations performed can be viewed as a one-layer DCN. Additional depth and nonlinearities in particular are known to markedly improve learning.

To address these issues, we propose the following:

- **Multi-kernel approach.** Replacement of the single kernel convolution with N output channels by a set of convolutional kernels of different dimensions (such that the number of output channels still sum up to N).
- **Additional depth.** Introduce additional depth for both *input-to-hidden* and *hidden-to-hidden* processing. For example, 1×1 convolutions may be used on the concatenated multi-kernel output, as seen in Fig. 1. An alternative use is 1×1 bottlenecks ahead of large filters to reduce the computational burden.
- **Flow-based masking.** To support the multi-kernel approach, we propose generating kernel-dependent attention masks. One mask is generated for each convolutional kernel and applied pixelwise to the input before convolution.

To the best of our knowledge, such a Network-in-LSTM has not been used before. Attention-based models for different LSTM variants are known, but they differ from our approach, which is specifically tailored to support the multi-kernel approach.

The remainder of our paper is organized as follows: In Section II, we discuss related work to our approach. We then present our method in detail in Section III. In Section IV, we comprehensively evaluate our approach, and we conclude our work in Section V.

II. Related Work

1) Action recognition: Action recognition has been a popular area of research for many years. To capture the information in videos, handcrafted global features such as the histogram of oriented gradients [7] can be used. In the presence of noise, however, such features fail easily [7]. To mitigate this issue, local features have increasingly been adopted [8][9][10]. Such local features focus on salient spatial regions in images; for example, a person, and can be temporally extended to some degree for videos.

Most modern action recognition methods rely on deep features. In a breakthrough work, Karpathy *et al.* [2] successfully investigated pooling strategies to fuse temporal information in DCNs. Further advances were made by Simonyan *et al.* [3], who showed that deep features extracted on optical flow can improve video classification. A notable insight of their work is the fact that convolutional networks based on optical flow features can be fine tuned from DCNs taught on RGB inputs, such as that used for image classification on the ImageNet dataset [11]. The multistream architecture introduced in [3] has since been adopted by many works for action recognition [12][13].

The works by Donahue *et al.* and Ng *et al.* [14][1] successfully employed recurrent networks, specifically LSTM networks, to learn temporal sequences in action recognition videos. Current approaches make extensive use of pretraining on large datasets. Carreira *et al.* recently proposed inflated 3D convolutions [15], and they demonstrated state-of-the-art performance by pretraining on large image and video datasets.

Several widely used datasets for action recognition are available. The most popular datasets have been UCF-101 [16] and HMDB-51 [17], which both contain thousands of videos and are therefore considered small scale. More recently, large-scale datasets such as Sports-1M [2], Kinetics [18], and Moments in Time [19] have been published, each of which encompasses hundreds of thousands to millions of videos.

2) Recurrent architectures and applications: In this work, we investigate improvements to convolutional recurrent networks, in particular ConvLSTM [6]. As traditional LSTM models suffer from a lack of spatial invariance by using fully connected operations, Shi *et al.* addressed the issue by replacing the operations with convolutions and subsequently evaluating their system on a meteorological task. We detail the ConvLSTM equations in Section III.

Although they lack spatial invariance, fully connected LSTM-based models have also been widely used in fields other than action recognition. Sutskever *et al.* [4] introduced the sequence-to-sequence framework to generate fixed-size representations from inputs of unconstrained sizes. The framework allows many applications; for example, the processing of natural language for translation purposes [20][4]. Based on the same approach, several works have proposed solutions to tasks such as video-caption generation [21][22][23].

Encoder–decoder structures, of which the sequence-to-sequence framework is one example, are frequently used for future prediction. Typically, an encoding LSTM reads in a sequence from the past or present, whereas a decoding LSTM produces the corresponding future. This can be used for low-dimensional tasks such as human pose prediction [24]. However, some work has also attempted to solve high-dimensional problems, such as the pixelwise reconstruction of future video frames; Srivastava *et al.* [25] proposed an unsupervised learning method that is evaluated on the synthetic *MovingMNIST* dataset, wherein synthetic video sequences are generated by assigning a handwritten digit (such as those in the MNIST dataset) a speed and an orientation. We use this intriguing dataset in our qualitative evaluation. Future prediction tasks are also possible with the convolutional extension ConvLSTM; for example, future semantic segmentation [26] and weather prediction [27][6].

Attention-based mechanisms in (FC-)LSTM have proven useful for a variety of tasks. Gao *et al.* [28] used such a model for video captioning applications. Wang *et al.* [29] employed attention-based LSTM networks for a tracking task. Fan *et al.* [30] conducted action recognition based on the human skeleton; their system architecture employed LSTM networks to achieve that goal. Finally, Zhao *et al.* [31] used visual attention for fine-grained object classification.

Most recently, an attention system was attempted in connection with ConvLSTM in the work by Li *et al.* [32]. Notably, our use of attention masks differs from their work; although our approach is also motion based, we generate masks that are specialized for use with a specific convolutional kernel in a multi-kernel system.

3) Inception: Finally, our work bears some resemblance to the well-known GoogLeNet [33], in particular the so-called *Inception* module, as well as some resemblance with the network-in-network (NIN) approach [34]. An *Inception* module is an ensemble of convolutional filters of different sizes (typically 1×1, 3×3, and 5×5) in parallel. The module is designed purely with computational complexity (in terms of operations and parameters) in mind and applied in a DCN for *static* image classification. By contrast, our work is designed under different considerations; we investigate the behavior of ConvLSTM for *temporal* processes and argue that multiple kernels are necessary for correct recognition.

III. Methodology

A. Long Short-Term Memory

In the following we briefly recapitulate both classical *long short-term memory* [5] (FC-LSTM) and the convolutional extension (ConvLSTM) presented in [6].

1) Classic model: Consider an input sequence $\mathbf{x}$ of length T, where $\mathbf{x}_t$ represents the t-th element. Such a sequence may for example be the RGB frames of a video clip or features extracted from a deep convolutional stack. An LSTM unit is typically expressed as follows:

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\circ\mathbf{c}_{t-1} + b_i\right) \quad (1)$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\circ\mathbf{c}_{t-1} + b_f\right) \quad (2)$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\circ\mathbf{c}_{t-1} + b_o\right) \quad (3)$$

$$\mathbf{c}_t = \mathbf{f}_t\circ\mathbf{c}_{t-1} + \mathbf{i}_t\circ\tanh\left(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + b_c\right) \quad (4)$$

$$\mathbf{h}_t = \mathbf{o}_t\circ\tanh\left(\mathbf{c}_t\right) \quad (5)$$

The tensor $\mathbf{c}_t$ is the so-called *cell state*, whereas $\mathbf{h}_t$ is named the *hidden state*, which acts as the output of the unit over time. The tensors $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$ are the *control gates*. We term the operations $\mathbf{W}_{x*}\cdot\mathbf{x}_t$ the *input-to-hidden* transition and the operations $\mathbf{W}_{h*}\cdot\mathbf{h}_{t-1}$ the *hidden-to-hidden* transition. We also note that some studies and implementations may ignore the Hadamard terms $\mathbf{W}_{c*}\circ\mathbf{c}_{t-1}$.

Backpropagation over time, which is used to train recurrent networks, suffers from the *vanishing gradient* [5] problem. The issue lies with the choice of activation functions; Sigmoid $\sigma(x)=\frac{1}{1+e^{-x}}$ and hyperbolic tangents $\tanh(x)=2\sigma(2x)-1$ have derivatives with upper bounds at 0.25 and 1, respectively, and repeated application therefore reduces the magnitude of a gradient. Notably, the LSTM formulation above avoids this problem; given any output $\mathbf{h}_{t_i}$, we can find a path to any cell state $\mathbf{c}_{t_j}$ while passing only a single activation function such as $\tanh(\cdot)$ or $\sigma(\cdot)$.

Furthermore, we note the use of the dot product in the formulation. In effect, this resembles a fully connected layer as known from DCNs. Such fully connected layers are not spatially invariant [35].

2) Convolutional extension: To address the problem of spatial invariance, the authors of [6] propose to replace the dot product by *convolutional* operations. We refer to such a system in this work as ConvLSTM. The exact formulation is as follows:

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{xi} * \mathbf{x}_t + \mathbf{W}_{hi} * \mathbf{h}_{t-1} + \mathbf{W}_{ci}\circ\mathbf{c}_{t-1} + b_i\right) \quad (6)$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{xf} * \mathbf{x}_t + \mathbf{W}_{hf} * \mathbf{h}_{t-1} + \mathbf{W}_{cf}\circ\mathbf{c}_{t-1} + b_f\right) \quad (7)$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{xo} * \mathbf{x}_t + \mathbf{W}_{ho} * \mathbf{h}_{t-1} + \mathbf{W}_{co}\circ\mathbf{c}_{t-1} + b_o\right) \quad (8)$$

$$\mathbf{c}_t = \mathbf{f}_t\circ\mathbf{c}_{t-1} + \mathbf{i}_t\circ\tanh\left(\mathbf{W}_{xc} * \mathbf{x}_t + \mathbf{W}_{hc} * \mathbf{h}_{t-1} + b_c\right) \quad (9)$$

$$\mathbf{h}_t = \mathbf{o}_t\circ\tanh\left(\mathbf{c}_t\right) \quad (10)$$

This modification ensures spatial invariance of the unit. The tensors $\mathbf{i}, \mathbf{f}, \mathbf{o}, \mathbf{c}$, and $\mathbf{h}$ are now convolutional maps and therefore higher dimensional.

B. Problem of Kernel Sizes

Each of the operations $\mathbf{W}_{x*} * \mathbf{x}_t$, $\mathbf{W}_{h*} * \mathbf{h}_{t-1}$ is associated with a single convolutional kernel. A problem that arises is the choice of optimal size; choosing a kernel size that is too large results in the system degenerating into the original fully connected formulation. However, if the size is too small, the kernel will not be able to capture all information. In the original work on ConvLSTM, the authors [6] found that a kernel size of 5×5 is optimal for the particular problem evaluated in their work, which was an unsupervised prediction problem such as weather prediction.

In this paper, we argue that the exclusive use of kernels of one particular size is not optimal for ConvLSTM; instead we suggest using an array of kernels of different sizes, reminiscent of the *Inception* module used with DCNs in [33]. Consider a video showing two objects, where the objects are moving at significantly different speeds. A small kernel is unable to link a fast-moving object during the transition from timestep t to timestep $t+1$, simply because it has moved too far. However, always using large kernels is disadvantageous because they require *more* parameters, are significantly *slower* ($\frac{5\cdot5}{3\cdot3}\approx2.78$), and *degenerate* into fully connected layers at very large sizes, which lack spatial invariance. To solve this dilemma, we propose employing multiple kernels.

1) Motion and kernel size: To support our hypothesis, we first visualize the problem of kernel size and velocity with the following simple experiment. In a variation of the *MovingMNIST* dataset (which we discuss in more detail in Section IV-C2), we construct a synthetic dataset of sequences of moving digits. We first pick a single digit from the MNIST dataset and subsequently assign it a velocity and a direction. The moving digit is then animated for T = 20 frames. The appearance of the digit remains unchanged at all times (i.e., we only translate it). On contact with the image boundaries, the moving digits are reflected. An example can be seen in Fig. 2.

On this dataset, we now train a future predictor in the form of an encoder–decoder architecture. The encoding ConvLSTM reads in an input sequence of $T_{in}=10$ frames and produces a fixed-size state $(\mathbf{C},\mathbf{H})$. A second decoding ConvLSTM is initialized with this state and generates $T_{out}=10$ output features. A final 1×1 convolution layer reconstructs the predicted sequence. With each pixel taking on values in $[0,1]$,

Fig. 2. Sequence of a single handwritten digit moving over time (here with $T = 5$). Note that the appearance of the digit remains unchanged at all times.

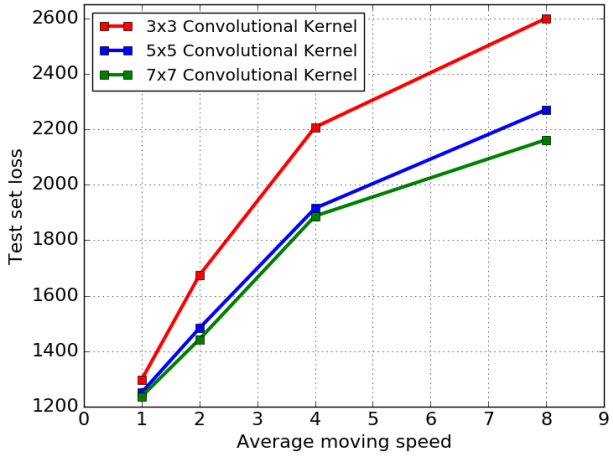


Fig. 3. Test set (cross-entropy) loss for an unsupervised future prediction task with single moving digits (MovingMNIST). We vary the average moving speed of the digits but keep the appearance otherwise intact. Our observation is that the loss improves overproportionally for smaller convolutional kernels on slowly moving digits in comparison with their larger counterparts. Based on this, we propose that there is a correlation between moving speed and optimal kernel size.

we can choose the binary sigmoid cross entropy (SCE) as our prediction loss:

$$L = \frac{-1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} p_n \log \hat{p}_n + (1 - p_n) \log (1 - \hat{p}_n)$$

To implement the encoder–decoder structure, we tried three different (traditional) ConvLSTM layers, each of them with a different kernel size (i.e., 3×3, 5×5, and 7×7 kernels). We varied the average speed at which the digit was moving and repeated the experiment.

Fig. 3 shows our results. The associated loss for smaller kernel sizes grew significantly faster than for larger kernel sizes. Concurrently, the computational costs were much higher for the larger kernels. We stress, however, that the digits in our experiment had static appearances, and in a real-world dataset motion may not necessarily be the only factor that determines the optimal kernel size.

C. Concatenation of multiple kernels

During implementation of a multi-kernel configuration, special care must be taken to correctly concatenate parallel kernels (see Fig. 4). From the ConvLSTM equations (6)–(9), we can see that the term inside the activation function (i.e., $\mathbf{W}_{x*} * \mathbf{x}_t + \mathbf{W}_{h*} * \mathbf{h}_{t-1} + b_*$) differs only in the parameters

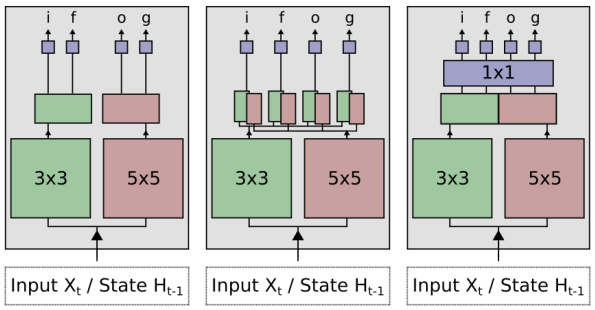


Fig. 4. Multi-kernel concatenation problem. Each gate activation i, f, o, g should take a mixture of both kernels as input. (a) Using naive concatenation, each gate takes only one kernel output as input. (b) Splitting and interleaving before concatenation are one possible remedy for this. (c) An additional 1 x 1 convolution also avoids this problem and integrates both results.

$\mathbf{W}, b$. A common implementation optimization is therefore to use two convolution operations $\mathbf{W}_x * \mathbf{x}_t$ and $\mathbf{W}_h * \mathbf{h}_{t-1}$ with $C' = 4 \cdot C$ channels. The result is then split up into the four corresponding terms.

In our system, when using multiple kernels in parallel, we concatenate the individual results. At this point, the data order must be considered, as visualized in Fig. 4a. A naive concatenation will result in the gate activations i, f, o, g to depend solely on one of the used kernels. However, each gate activation should take both kernel outputs as input. We propose two strategies to avoid this. First, as in Fig. 4b, we can interleave the results by first splitting each kernel's output into four and then concatenating in the correct order. Second, we can use a 1×1 convolution operation, which integrates the individual results (see Fig. 4c). Our results use the interleaving strategy in Fig. 4b if no final 1×1 kernel is present.

Our second strategy has the additional benefit that it introduces additional nonlinearity and depth, which is known to be beneficial [36] to deep learning. Convolutions with a 1×1 kernel can, however, also be used as so-called bottleneck layers: the layer is added to reduce the number of input channels before applying large convolutional kernels (e.g., a 5×5 kernel). We test bottleneck layers in our inception-like configuration in Section IV-A4.

D. Attention-based masking

In our multi-kernel extension, any kernel learns on all regions of an image, even if it is not optimally suited for this image region. Ideally, our system should avoid this. In this work, we aim to enforce large kernels to concentrate on faster objects and smaller kernels to concentrate on slower objects. Note that the background scene and static objects may also contribute to the learning process, and we consequently suggest to apply attention masks only on a subset of convolutional kernels.

To determine the utility of a kernel on a particular image region, we generate attention masks from optical flow features. The magnitude of the optical flow determines the distance a particular pixel has moved on the x- or y-axis, and therefore

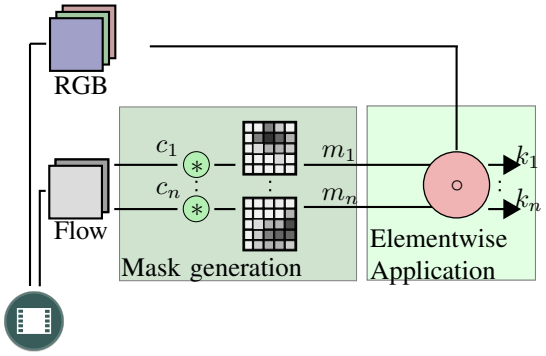


Fig. 5. Multi-kernel attention scheme. For each kernel using attention, we generate a mask m_i by convolution c_i over the flow features. Subsequently, for each ConvLSTM kernel k_i, the corresponding mask is applied to the RGB input features by element-wise multiplication.

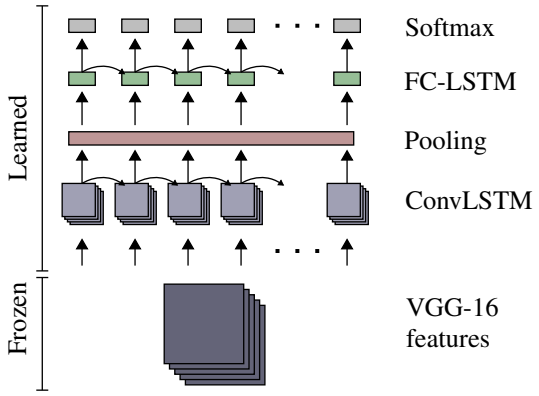


Fig. 6. Model used for all VGG-based experiments on Sports-1M and UCF-101.

represents speed. To generate each mask, we employ a DCN on the optical flow features. A non-linearity, such as the sigmoid, is applied subsequently. This mask can then be applied by element-wise multiplication to all input features $\mathbf{x}_t$. Our goal is that through backpropagation of error, each mask specializes onto the corresponding convolutional kernel. Fig. 5 shows this process.

The above masking process affects input-to-hidden convolutions (i.e., parameters $\mathbf{W}_{x*}$), and we reserve a hidden-to-hidden extension ($\mathbf{W}_{h*}$) for future work.

IV. Evaluation

In the following, we present our experimental results. We consider both quantitative and qualitative aspects. First, we seek a direct comparison with the ConvLSTM baseline, and we show on the example of video classification that our proposed method improves classification accuracy. Following this, we attempt to gain a better understanding of our modifications in the qualitative analysis.

A. Quantitative results

The work in [6] was conducted with weather forecasting in mind. Our proposed changes, however, benefit applications in which visible objects move at different speeds. Hence, we chose action recognition as the application to quantitatively evaluate our system. We begin with a description of the two popular action recognition datasets used in this work: UCF-101 [16] and Sports-1M [2].

1) Datasets: The *UCF-101* action recognition dataset contains 13,320 videos, each showing a single human action. The average length of the videos is 180 frames (i.e., just a few seconds), and there are 101 categories, which implies a small number of videos per class, slightly more than 100 on average. This causes models trained solely on UCF-101 to overfit easily. The dataset offers three different splits into training and test data; we used split one.

The second dataset, the*Sports-1M dataset*, is a collection of over one million videos on YouTube in 487 classes, which all relate to sports. The videos have unconstrained lengths, ranging from a few seconds to hours. The size of the dataset inherently causes difficulties; required storage space is massive even on the lowest resolution (approximately 7 TB), and processing one million videos in each epoch leads to long computation times. For this study, we therefore decided to run our experiments on a subset by randomly sampling 20 classes[2].

2) Experimental setup: Two difficulties arose during our experiments. First, unrolling a convolutional LSTM over time requires large amounts of GPU memory. Second, training on a small-scale dataset such as UCF-101 will lead to severe overfitting when training from scratch. To mitigate these two problems, we therefore decided to initialize the weights of the convolutional stack with pretrained weights and subsequently freeze them. Here, we describe our setup for the two convolutional base network variants.

The first set of experiments used a VGG-16 [36] convolutional stack. We used ImageNet-pretrained weights and extracted features of dimensions $14 \times 14 \times 512$ at layer `conv5_3`. Using the larger spatial extent in this layer exposes the differences between smaller and larger kernels more clearly. A ConvLSTM unit then processed the features. We established both 3×3 and 5×5 kernel baselines with $C = 512$ channels and then compared with a mixture of both kernels ($C_{3\times3} = 256, C_{5\times5} = 256$) while keeping the total number of channels fixed. The ConvLSTM layer was followed by pooling, an FC-LSTM, and a Softmax classifier.

Our second set of experiments employed a recent state-of-the-art action recognition network named I3D [15]. Here, we initialized with Kinetics-pretrained weights and extracted RGB features of dimensions $7\times7\times1024$ at layer `incept5b`. Accuracy was averaged over all frames in both sets of experiments.

3) Results of VGG-based configurations: UCF-101 contains very short videos, and we therefore sampled them at 15 fps. From a set of 140 extracted frames, we chose $T = 50$ (i.e., frames 25 to 75) frames as input. Our results for this dataset can be found in Table I.

Videos in Sports1M-20 are typically longer, and we therefore sampled frames at 1 fps. We sampled a set of 140 frames and picked $T = 30$ frames (i.e., frames 70 to 100) as input

[2]*rafting, skittles, test cricket, shidokan, pitch and putt, dirt track racing, freestyle skiing, street football, sprint, motorcycle speedway, trial, dressage, surf fishing, juggling club, soft tennis, sailing, road racing, jetsprint, gatka,* and *enduro.*

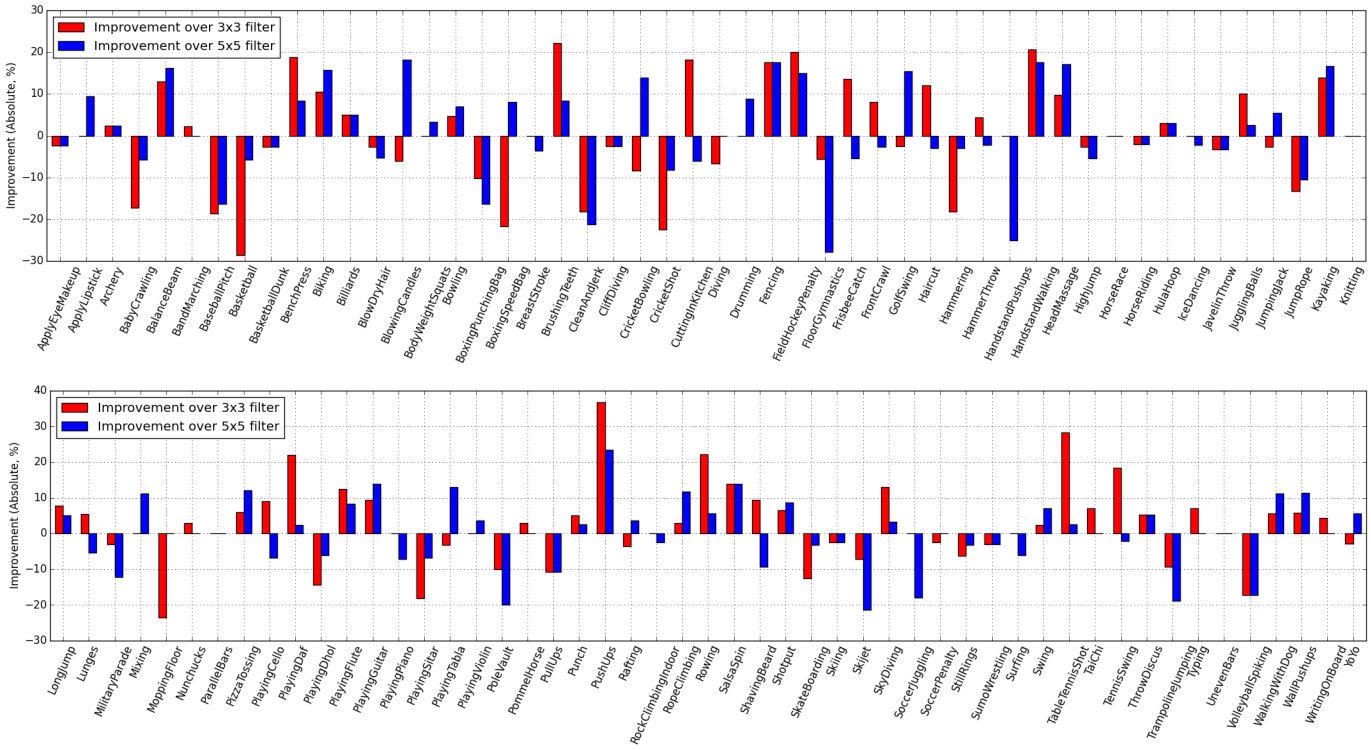


Fig. 7. Classwise comparison of our multi-kernel system with two ConvLSTM baselines on the UCF-101 test set (split 1). Notably, classes 27 and 71 (`Fencing`, `PushUps`) strongly outperform both baselines.

TABLE I
SUPERVISED CLASSIFICATION RESULTS OF UCF-101 ON TOP OF A VGG [36] ARCHITECTURE. OUR MULTI-KERNEL CONFIGURATION CLEARLY OUTPERFORMS BOTH BASELINE VARIANTS.

Configuration	Top-1 Accuracy
Baseline $3 \times 3 \times 512$	71.27%
Baseline $5 \times 5 \times 512$	72.20%
Simple Multi-kernel $C_3 = C_5 = 256$ **(Ours)**	73.18%
Simple Multi-kernel (with stacked 1×1) $C_3 = C_5 = 256$ **(Ours)**	**74.09%**

TABLE II
TIME PER ITERATION OF CONVLSTM LAYER WITH $C_{VGG} = 512$ INPUT CHANNELS, BATCH SIZE OF 10, AND T = 20 FRAMES PER VIDEO. THE NUMBERS ARE AVERAGED OVER 100 ITERATIONS.

Configuration	Training	Testing
$3 \times 3 \times 256$	3.424s	2.821s
$5 \times 5 \times 256$	4.430s	3.290s
Simple Multi-kernel $C_3 = C_5 = 128$ **(Ours)**	4.007s	3.008s
Simple Multi-kernel (with stacked 1×1) $C_3 = C_5 = 128$ **(Ours)**	4.176s	3.072s

TABLE III
NUMBER OF PARAMETERS REQUIRED FOR DIFFERENT CONVLSTM CONFIGURATIONS. WE ASSUME $C_{VGG} = 512$ INPUT CHANNELS.

Configuration	Weights	Biases
$3 \times 3 \times 256$	4.7 M	1024
$5 \times 5 \times 256$	13.1 M	1024
$7 \times 7 \times 256$	25.7 M	1024
Simple Multi-kernel $C_3 = C_5 = 128$ **(Ours)**	8.9 M	1024
Simple Multi-kernel (with stacked 1×1) $C_3 = C_5 = 128$ **(Ours)**	**9.1 M**	**1024**

TABLE IV
SUPERVISED CLASSIFICATION RESULTS OF SPORTS1M-20 ON TOP OF A VGG [36] ARCHITECTURE. BOTH BASELINE AND THE PROPOSED METHOD EACH HAVE A TOTAL OF $C = 512$ OUTPUT FEATURE MAPS.

Configuration	Top-1 Accuracy
Baseline $3 \times 3 \times 512$	80.67%
Baseline $5 \times 5 \times 512$	81.09%
Simple Multi-kernel $C_3 = C_5 = 256$ **(Ours)**	**81.34%**

for our system. In cases where the videos were not sufficiently long, those frames were sampled modulus their length. Our results are presented in Table IV.

Our experiment evaluated the performance of the proposed multi-kernel system and compared it against the ConvLSTM baselines, which use a single convolutional kernel only. Our approach outperformed both the 3×3 and 5×5 ConvLSTM baselines, with improvements of 2.82% and 1.89% on UCF-101, respectively. The improvement was more marginal on Sports-1M, where we gained 0.67% and 0.25%, respectively. In both cases, the larger 5×5 baseline performed better than the 3×3 baseline, as already observed in Fig. 2. Furthermore, we attempted several configurations of 3×3 and 5×5 kernels for our multiple kernel system on UCF-101, and the results are shown in Fig. 8.

Both baseline and our extension performed well below state-of-the-art levels on UCF-101. Modern state-of-the-art net-

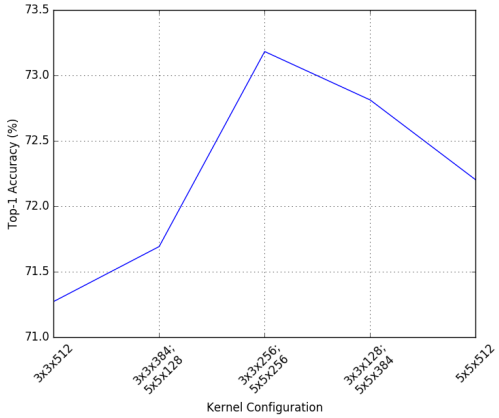
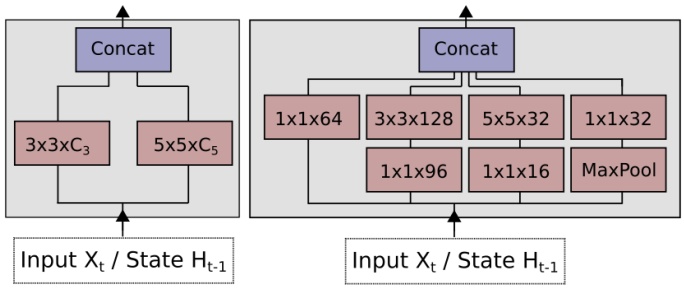


Fig. 8. Top-1 accuracy on the UCF-101 dataset with varying proportions of 3×3 and 5×5 convolutional kernels.



(a) (b)

Fig. 9. Configurations used in our experiments: (a) simple multi-kernel and (b) Inception-like multi-kernel.

works utilize extensive pretraining (e.g., on Kinetics as in [15]) and batch normalization [37] or dropout [38]. Our proposed extension can be combined with such modern networks easily. Here, we report the results of experiments with an I3D-based setup and show that state-of-the-art can be beaten on a subset of UCF-101.

4) Results of I3D-based configurations: We first show that standard I3D can be outperformed on the RGB stream on a subset[3] of classes of UCF-101. According to [15], I3D greatly profits from high temporal resolution, and consequently, we sampled frames at 24 fps, selecting clips of 64 frames. The pooling and FC-LSTM used in the VGG configuration were removed and replaced with a 1×1 convolution with batch normalization. We tested a configuration similar to the Inception layer in [33] (see Fig. 9b). We used $C = 256$ in all cases. Because batch normalization can combat overfitting, we supplied all convolutional kernels with such normalization. Our results for a 17-classes subset of UCF-101 are presented in Table V; both of our configurations outperformed I3D, which is state-of-the-art, by over 0.84%.

We also evaluated our proposed optical flow-based attention. Using an I3D base network and the simple multi-kernel configuration $C_{3\times3} = 128, C_{5\times5} = 128$, we generated attention masks and applied them to the input features of the ConvLSTM. The result in Table VI shows an improvement of 0.18%. We used a single convolutional layer to generate the attention masks. In our future work, we will consider deeper architectures to improve flow masks further.

Finally, we trained an Inception multi-kernel in an end-to-end fashion by simultaneously fine tuning the I3D base network. The Inception multi-kernel was an excellent choice for this experiment because 1×1 bottleneck convolutions help reduce the memory footprint significantly. Table VI shows that training in an end-to-end fashion improved accuracy by a further 1.69%.

[3]ApplyEyeMakeup, BlowDryHair, BoxingPunchingBag, FloorGymnastics, HandstandPushups, HorseRace, JumpRope, Lunges, MilitaryParade, PlayingCello, PlayingPiano, PlayingTabla, PushUps, Rafting, Shotput, Skiing, WalkingWithDog

TABLE V
STATE-OF-THE-ART EXPERIMENT ON A 17-CLASS SUBSET OF UCF-101. OUR INCEPTION-LIKE MULTI-KERNEL OUTPERFORMS STANDARD I3D BY 0.84% AS WELL AS BOTH CONVLSTM BASELINES ON THE RGB STREAM. FURTHER IMPROVEMENT CAN PRINCIPALLY BE ACHIEVED BY PRETRAINING THE CONVLSTM LAYER, WHICH IS OUTSIDE THE SCOPE OF THIS STUDY. FOR NOMENCLATURE, PLEASE REFER TO FIG. 9.

Configuration		Top-1 accuracy
I3D	Baseline	96.62%
ConvLSTM 3×3	Baseline	96.46%
ConvLSTM 5×5	Baseline	96.62%
Inception multi-kernel	**Ours**	**97.46%**

TABLE VI
CLASSIFICATION RESULTS ON UCF-101, CHOOSING I3D AS THE BASE NETWORK: **END-TO-END** TRAINING AND ABLATIVE STUDY FOR OUR **FLOW-BASED ATTENTION** SCHEME.

Configuration		Top-1 accuracy
ConvLSTM 3×3	Baseline	86.33 %
ConvLSTM 5×5	Baseline	86.92 %
Simple multi-kernel	**Ours**	87.21%
Simple multi-kernel (*with* flow-based attention)	**Ours**	**87.39%**
Inception multi-kernel	**Ours**	88.40%
Inception multi-kernel (trained end-to-end)	**Ours**	**90.09%**

B. Analysis

Because our proposed extensions may perform better on certain data, we also provide a breakdown of classification accuracy by class for UCF-101 (with the VGG-based setup). Our system improved classification if both the blue and red bars in Fig. 7 show positive values. Considerable improvements can be seen in certain categories, such as `Fencing` or `Pushups`. In other cases, a class did not benefit from multiple kernels. We suggest that this occurs if motion velocity is fairly constant throughout the instances of a class. Such an example is `MoppingFloor`, where a pure 3×3 ConvLSTM exhibits better performance.

To investigate this more closely, we collected the sets of UCF-101 classes in which our system performed significantly worse or better (i.e., classes with a difference of at least 10% accuracy) and computed the magnitude of the optical flow in these videos. Taking the median over the respective sets of classes, we can investigate the distribution of squared velocities $v = x^2 + y^2$, where x, y are the components of the

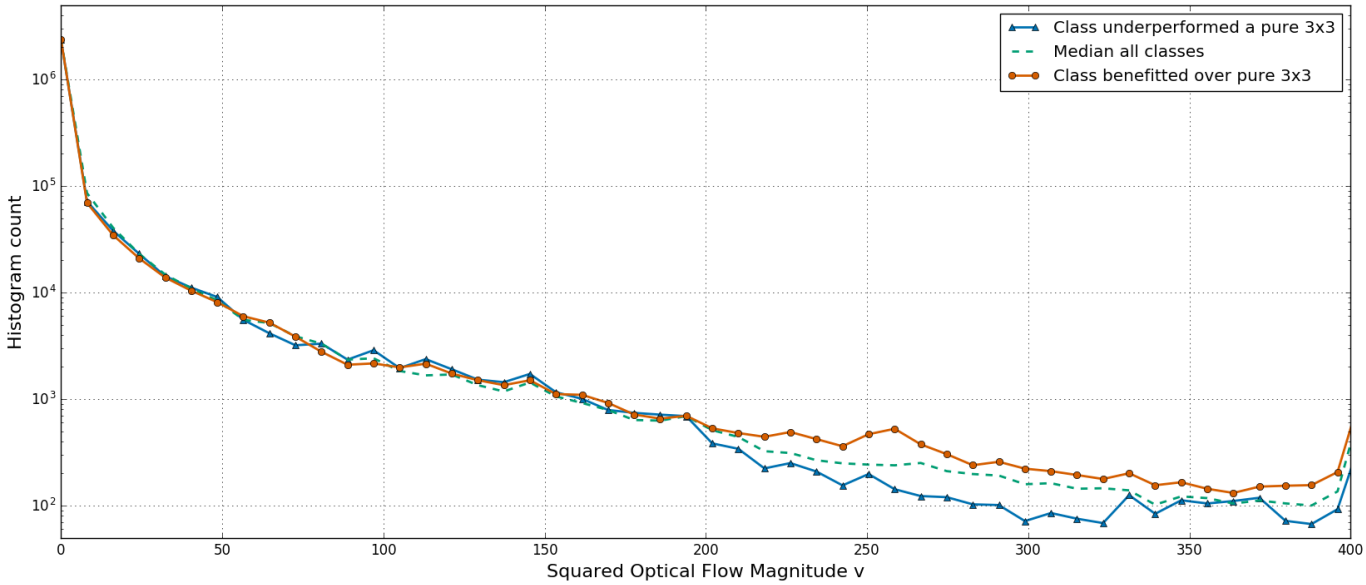


Fig. 10. Histogram of squared optical flow velocities $v = \left(x^2 + y^2\right)$ for two sets of classes. The blue line shows the distribution for classes that performed significantly better in the ConvLSTM 3×3 baseline, whereas the red line shows the counts for classes in which our approach outperformed the 3×3 baseline. For velocities $200 \leq v \leq 400$, the blue line is significantly below the median of all classes, while the red line is above the median. In other words, our extensions work best when many velocities between 200 and 400 are present. Note that the peak near $v = 400$ is because we clipped all x, y values with magnitudes above 20, and we focus on $0 \leq v \leq 400$ here (the maximum value of v is 800). Furthermore. the histogram counts are normalized by number of videos per class.

two-dimensional optical flow. The histogram for comparison with the 3×3 ConvLSTM baseline can be found in Fig. 10, and the histogram for comparison with the 5×5 ConvLSTM baseline is in Fig. 11.

Our improvements in terms of classification accuracy coincide with a *larger proportion* of *higher* velocities in the optical flow, whereas the failure cases coincide with a reduced amount of these speeds. More specifically, the divergence occurs for $200 \leq v \leq 400$ compared with the 3×3 baseline and for $100 \leq v \leq 300$ compared with the 5×5 baseline. This makes sense; we can improve performance over the 3×3 ConvLSTM particularly if more higher velocities than those that a 3×3 convolutional kernel can capture are present. Similarly, to improve performance over the 5×5 ConvLSTM, more medium velocities must be present because only these can be captured by the smaller convolutional kernel.

1) Computational complexity: We outline the computational impact of our multi-kernel approach in Table II. All timings refer exclusively to the ConvLSTM layer (i.e., they do not include the feature extractor). As expected, the processing time for mixing kernels of two sizes is between the required times for the traditional ConvLSTM layers of the respective kernel sizes. The overhead is minimal and in the order of 1%.

We also computed the number of parameters in Table III, again only for the ConvLSTM layer. The results underline the necessity of our proposed changes, because larger filters require tens of millions of parameters. To clarify, consider that the 25 million parameters of a 7×7 ConvLSTM make up approximately 18% of the total parameters of VGG-16. Mixing kernels of different sizes enables reduction of the parameter count. This count can be further reduced by adding 1×1 bottleneck layers ahead of larger convolutional kernels.

TABLE VII
PREDICTION LOSS (SCE) FOR LOCAL MOTION IN HUMAN GESTURES ON THE JHMDB DATASET. SMALL KERNELS PROVIDE SLIGHTLY BETTER PERFORMANCE, HOWEVER THE IMPACT IS SMALLER THAN FOR GLOBAL MOTION. *Normal speed* REFERS TO SAMPLING THE DATASET AT EVERY FRAME, WHEREAS *fast speed* INDICATES SAMPLING ONLY EVERY THIRD FRAME.

Configuration	Normal speed	Fast speed
$3 \times 3 \times 256$	1288	1521
$5 \times 5 \times 256$	1295	1521
$7 \times 7 \times 256$	1331	1553

2) Global motion and local motion: In Section III-B1, we studied how convolutional kernels of different sizes reacted to moving objects. However, our investigation was limited to global motion, in which an object is uniformly moving in one direction. Here, we investigate if the same is true for local motion using human gestures as examples. Therefore, we use an action recognition dataset with pose annotation, JHMDB [39]. Human poses in JHMDB are represented by 15 two-dimensional pose joint coordinates.

Our goal is to analyze local motion. To simplify our experiments, we first eliminated texture information as well as global motion information as follows: we preprocessed the pose data by subtracting the coordinates of the center joint, here "belly." This was followed by centering and scaling. To build an input representation that can be processed by a ConvLSTM, we generated a skeleton for each pose by drawing lines between pose joints on a black background. Fig. 12 shows examples of this dataset (top row), and visualizes prediction results for both local and global motion. To test the influence of the kernel size, we again used a prediction task as in Section III-B1. Images of size 56×56 were max-pooled and processed by a

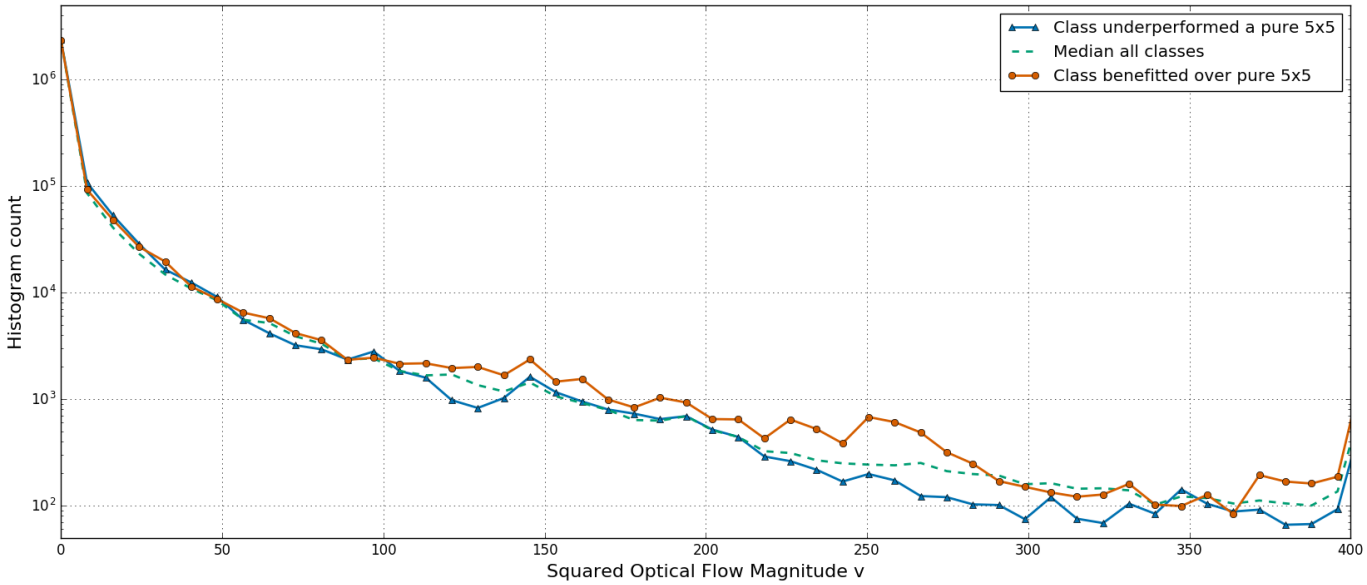
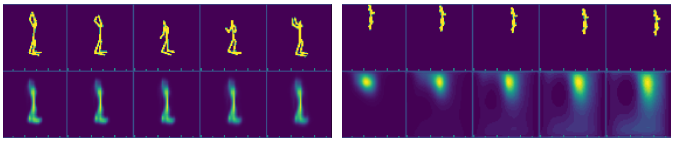


Fig. 11. Histogram of squared optical flow velocities $v = \left(x^2 + y^2\right)$ (also see Fig. 10) for a 5×5 ConvLSTM baseline and our extension. In comparison to Fig. 10, the red and blue lines diverge earlier, approximately between $100 \leq v \leq 300$. This is logical; our system performs better than a pure 5×5 ConvLSTM baseline when there are fewer high velocities and more small velocities.



(a) (b)

Fig. 12. Visualization of (a) Local motion and (b) global motion prediction on JHMDB. Top row: Groundtruth, bottom row: Prediction.

two-layer ConvNet first. The resulting 28×28 features were fed to a ConvLSTM with $C = 256$. The prediction task was implemented in an encoder–decoder fashion, and the used loss was the sigmoid cross entropy.

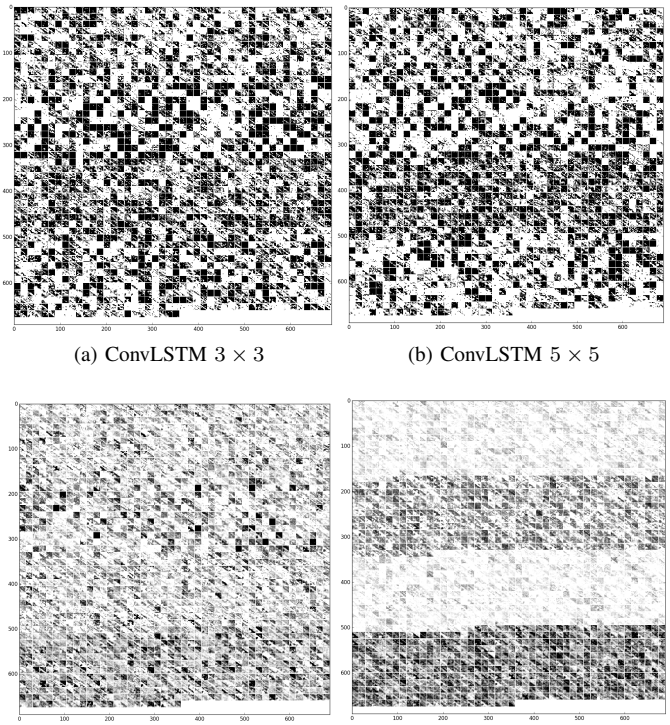
Quantitatively, the results in Table VII show a slight improvement for smaller kernel sizes. Compared to the global motion experiment (see Fig. 3), the effect is significantly smaller. We argue that gesture motion typically occurs in a more confined area. By providing a mixture of small and large kernels, our system is robust to both global and local motion.

C. Qualitative results

Here, we discuss visualizations that provide insight into our proposed extensions. In particular, we discuss saturation problems and our attention scheme.

1) Saturation in ConvLSTM: Gate activations in our baseline ConvLSTM models typically took on extreme values. Using our multi-kernel scheme with additional *depth*, i.e., the 1×1 convolution in our model, this was not the case. We visualize this matter in Fig. 13, where we show the feature maps (after applying σ and $\tanh$, respectively). Fig. 13 a) and 13 b) appear black and white due to the extreme values, whereas our approach in 13 c) and 13 d) produces well-balanced features.

We also note a difference in homogeneity between Fig. 13 c) and 13 d), which differs only in an additional $\tanh$ activation

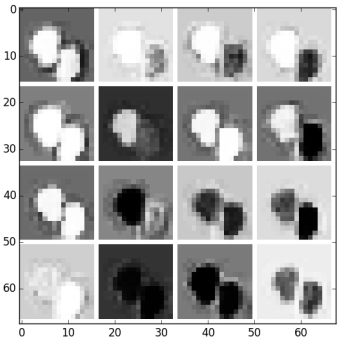


(a) ConvLSTM 3×3 (b) ConvLSTM 5×5

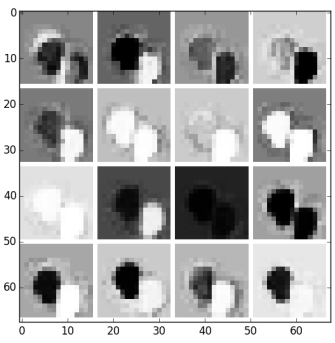
(c) Multi-kernel, no additional activation function. (d) Multi-kernel, $\tanh$ activation.

Fig. 13. Visualization of the three LSTM gates (*input*, *forget*, *output*) and cell activation (*tanh* term in cell-state) features (in that order). Notably, both baselines (a, b) are oversaturated, taking on only black-and-white color values. Adding the nonlinearity improved homogeneity within gates (c, d). Best viewed digitally.

(a) Input sequence.



(b) Masks for 5×5 kernel.



(c) Masks for 7×7 kernel.

Fig. 14. Visualization of the (16-channel) input attention masks on a modified MovingMNIST dataset for the (a) last frame in sequence. Black areas represent 0 and removed information, whereas white areas represent 1 and admit information. Notice there is a preference in (b) to admit the slow moving digit ("9") and in (c) to admit the fast moving digit ("7").

function before the 1×1 convolutional layer. These findings clarify which feature map is associated with which gate; *input gate* activations are generally high and encourage saving of input information, whereas *forget gate* activations have a tendency to be lower. Note that cell activations have a value range of (-1,1) and therefore appear darker and are not directly comparable to the control gates.

2) Multi-kernel attention: In Section III, we discussed a flow-based attention mechanism specifically for multiple kernel systems. To visualize the attention masks, we chose the MovingMNIST dataset, which is a synthetic dataset that is based on the handwritten digits of the well-known MNIST dataset. Sequences of 20 frames were generated by picking a 22×22 digit from MNIST and pasting it into a 64×64 empty bitmap. A speed and an orientation were assigned to the digit. On contact with the image borders, the digit "bounced" back. Several variations of this dataset exist; here, we use the sequence generator supplied[4] with the original ConvLSTM work [6] to generate a sequence with two digits, where one digit moves at an average speed of 1 and the other at an average speed of 8.

Following [6] and [25], we trained a future predictor based on our multi-kernel ConvLSTM. As in the original ConvLSTM [6], we patchified the input. The encoding ConvLSTM was fitted with a 5×5 and a 7×7 kernel, both using the attention mechanism.

3) Discussion: We visualize the learned masks and the corresponding frame sequence in Fig. 14. Due to the patchification, each mask had 16 channels. Visual inspection reveals that the slower digit (the "9") is more frequently associated with the smaller kernel (i.e., it is represented by a white blob; white represents a high activation value) in Fig. 14b a total of 10 out of 16 times. The opposite is true for the larger kernel in Fig. 14c, where the fast-moving digit ("7") is represented 11 out of 16 times by a white blob. This clearly shows the tendency of the larger kernel to associate with faster objects and supports our initial argument on multi-kernel ConvLSTM.

[4]Website: http://home.cse.ust.hk/~xshiab/

V. Conclusion

We analyzed the problem of different speeds in ConvLSTM and proposed replacing the single convolutional kernel with a set of kernels or a small network. To the best of our knowledge, we are the first to propose such an extension. We also presented an attention-based method that is specifically tailored to our system. Our results were tested using the popular UCF-101 and Sports-1M action recognition datasets, and the findings were discussed using qualitative analysis.

1) Future Work: In Section III-D, we described an attention scheme for use with multiple kernels. We limited the investigation to the input-to-hidden operations $\mathbf{W}_{x*}$ and demonstrated the behavior of multiple kernel attention exemplary on MovingMNIST. Future studies may wish to examine if an extension to hidden-to-hidden ($\mathbf{W}_{h*}$) operation is beneficial for multiple kernel systems. An initial idea is to introduce an additional latent state to the ConvLSTM layer, which is manipulated by input and forget gates in a manner similar to the cell-state in equation (9). However, this will come at the expense of higher memory usage.

Acknowledgement

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 108-2634-F-002-004. We also benefit from the NVIDIA grants and the DGX-1 AI Supercomputer.

References

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

[2] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 1725–1732.

[3] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, pp. 568–576. [Online]. Available: http://dl.acm.org/citation.cfm?id=2968826.2968890

[4] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[6] X. Shi, 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 *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 802–810. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969239.2969329

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 886–893 vol. 1.

[8] A. Klaser, M. Marszalek, and C. Schmid, "A Spatio-Temporal Descriptor Based on 3D-Gradients," in *BMVC 2008 - 19th British Machine Vision Conference*, M. Everingham, C. Needham, and R. Fraile, Eds. Leeds, United Kingdom: British Machine Vision Association, Sep. 2008, pp. 275:1–10. [Online]. Available: https://hal.inria.fr/inria-00514853

[9] M. Bregonzio, S. Gong, and T. Xiang, "Recognising action as clouds of space-time interest points," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 1948–1955.

[10] I. Laptev, "On space-time interest points," *Int. J. Comput. Vision*, vol. 64, no. 2-3, pp. 107–123, Sep. 2005. [Online]. Available: http://dx.doi.org/10.1007/s11263-005-1838-7

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[12] X. Wang, L. Gao, P. Wang, X. Sun, and X. Liu, "Two-stream 3-d convnet fusion for action recognition in videos with arbitrary size and length," *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 634–644, March 2018.

[13] Y. Shi, Y. Tian, Y. Wang, and T. Huang, "Sequential deep trajectory descriptor for action recognition with three-stream cnn," *IEEE Transactions on Multimedia*, vol. 19, no. 7, pp. 1510–1520, July 2017.

[14] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677–691, April 2017.

[15] 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)*, July 2017, pp. 4724–4733.

[16] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, vol. abs/1212.0402, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1212.html#abs-1212-0402

[17] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: a large video database for human motion recognition," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[18] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, A. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," *CoRR*, vol. abs/1705.06950, 2017.

[19] M. Monfort, B. Zhou, S. A. Bargal, T. Yan, A. Andonian, K. Ramakrishnan, L. Brown, Q. Fan, D. Gutfruend, C. Vondrick *et al.*, "Moments in time dataset: one million videos for event understanding."

[20] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016. [Online]. Available: http://arxiv.org/abs/1609.08144

[21] S. Venugopalan, M. Rohrbach, J. Donahue, R. J. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence – video to text," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 4534–4542, 2015.

[22] 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.

[23] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. J. Mooney, and K. Saenko, "Translating videos to natural language using deep recurrent neural networks," *CoRR*, vol. abs/1412.4729, 2014. [Online]. Available: http://arxiv.org/abs/1412.4729

[24] J. Walker, K. Marino, A. Gupta, and M. Hebert, "The pose knows: Video forecasting by generating pose futures," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 3352–3361.

[25] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 843–852. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045209

[26] M. Rochan *et al.*, "Future semantic segmentation with convolutional lstm," *arXiv preprint arXiv:1807.07946*, 2018.

[27] S. Kim, S. Hong, M. Joh, and S. Song, "Deeprain: Convlstm network for precipitation prediction using multichannel radar data," *CoRR*, vol. abs/1711.02316, 2017. [Online]. Available: http://arxiv.org/abs/1711.02316

[28] L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen, "Video captioning with attention-based lstm and semantic consistency," *IEEE Transactions on Multimedia*, vol. 19, no. 9, pp. 2045–2055, Sept 2017.

[29] Q. Wang, C. Yuan, J. Wang, and W. Zeng, "Learning attentional recurrent neural network for visual tracking," *IEEE Transactions on Multimedia*, pp. 1–1, 2018.

[30] Z. Fan, X. Zhao, T. Lin, and H. Su, "Attention based multi-view re-observation fusion network for skeletal action recognition," *IEEE Transactions on Multimedia*, pp. 1–1, 2018.

[31] B. Zhao, X. Wu, J. Feng, Q. Peng, and S. Yan, "Diversified visual attention networks for fine-grained object classification," *IEEE Transactions on Multimedia*, vol. 19, no. 6, pp. 1245–1256, June 2017.

[32] Z. Li, E. Gavves, M. Jain, and C. G. M. Snoek, "Videolstm convolves, attends and flows for action recognition," *CoRR*, vol. abs/1607.01794, 2016. [Online]. Available: http://arxiv.org/abs/1607.01794

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[34] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013.

[35] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, ser. CVPR'04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 97–104. [Online]. Available: http://dl.acm.org/citation.cfm?id=1896300.1896315

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

[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 448–456. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045167

[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2670313

[39] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black, "Towards understanding action recognition," in *International Conf. on Computer Vision (ICCV)*, Dec. 2013, pp. 3192–3199.

Sebastian Agethen received his diploma in computer science in 2011 from RWTH Aachen University, Germany. He is currently working towards his Ph.D. degree at the Graduate Institute for Networking and Multimedia at National Taiwan University, Taiwan, which he expects to obtain in 2019. His research interests include deep learning, as well as human action recognition and prediction.

Winston H. Hsu (S'03M'07SM'12) received the Ph.D. degree in electrical engineering from Columbia University, New York, NY, USA. He is keen to realizing advanced researches towards business deliverables via academia-industry collaborations and co-founding startups. Since 2007, he has been a Professor with the Graduate Institute of Networking and Multimedia and the Department of Computer Science and Information Engineering, National Taiwan University. His research interests include large-scale image/video retrieval/mining, visual recognition, and machine intelligence. Dr. Hsu served as the Associate Editor for the IEEE TRANSACTIONS ON MULTIMEDIA and on the Editorial Board for the IEEE MultiMedia Magazine.