LSTA: Long Short-Term Attention for Egocentric Action Recognition

Swathikiran Sudhakaran[1,2], Sergio Escalera[3,4], Oswald Lanz[1]

[1]Fondazione Bruno Kessler, Trento, Italy
[2]University of Trento, Trento, Italy
[3]Computer Vision Center, Barcelona, Spain
[4]University of Barcelona, Barcelona, Spain

{sudhakaran,lanz}@fbk.eu, sergio@maia.ub.es

Abstract

Egocentric activity recognition is one of the most challenging tasks in video analysis. It requires a fine-grained discrimination of small objects and their manipulation. While some methods base on strong supervision and attention mechanisms, they are either annotation consuming or do not take spatio-temporal patterns into account. In this paper we propose LSTA as a mechanism to focus on features from spatial relevant parts while attention is being tracked smoothly across the video sequence. We demonstrate the effectiveness of LSTA on egocentric activity recognition with an end-to-end trainable two-stream architecture, achieving state of the art performance on four standard benchmarks.

1. Introduction

Recognizing human actions from videos is a widely studied problem in computer vision. Most research is devoted to the analysis of video captured from distant, third-person views. Egocentric (first-person) video is an important and relatively less explored branch with potential applications in robotics, indexing and retrieval, human-computer interaction, or human assistance, just to mention a few. Recent advances in deep learning highly benefited problems such as image classification [10, 37] and object detection [17, 9]. However, the performance of deep learning action recognition from videos is still not comparable to the advances made in object recognition from still images [10]. One of the main difficulties in action recognition is the huge variations present in the data caused by the highly articulated nature of the human body. Human kinesics, being highly flexible in nature, results in high intra-subject and low inter-subject variabilities. This is further challenged by the variations introduced by the unconstrained nature of the environment where the video is captured. Since videos are composed of image frames, this introduces an additional dimension to the data, making it more difficult to define a model that properly focuses on the regions of interest that better discriminate particular action classes. In order to mitigate these problems, one approach could be the design of a large scale dataset with fine-grain annotations covering the space of spatio-temporal variabilities defined by the problem domain, which would be unfeasible in practice.

Here, we consider the problem of identifying fine-grained egocentric activities from trimmed videos. This is a comparatively difficult task considered to action recognition since the activity class depends on the action and the object on to which the action is applied to. This requires the development of a method that can simultaneously recognize the action as well as the object. In addition, the presence of strong ego-motion caused by the sharp movements of the camera wearer introduces noise to the video that complicates the encoding of motion in the video frame. While incorporating object detection can help the task of egocentric action recognition, still this would require fine-grain frame level annotations, becoming costly and impractical in a large scale setup.

Attention in deep learning was recently proposed to guide networks to focus on regions of interest relevant for a particular recognition task. This prunes the network search space and avoids computing features from irrelevant image regions, resulting in a better generalization. Existing works explore both bottom-up [39] and top-down attention mechanisms [30]. Bottom-up attention relies on the salient features of the data and is trained to identify such visual patterns that distinguish one class from another. Top-down attention applies prior knowledge about the data for developing attention, e.g. the presence of certain objects which can be obtained from a network trained for a different task. Recently, attention mechanisms have been successfully applied to egocentric action recognition [13, 30], surpassing the performance of non-attentive alternatives. Still, very few attempts have been done to track attention into spatio-temporal egocentric action recognition data. As a result,

current models may lose a proper smooth tracking of attention regions in egocentric action videos. Furthermore, most current models base on separate pre-training with strong supervision, requiring complex annotation operations.

To address these limitations, in this work we investigate on the more general question of *how a video CNN-RNN can learn to focus on the regions of interest to better discriminate* the action classes. We analyze the shortcomings of LSTMs in this context and derive Long Short-Term Attention (LSTA), a new recurrent neural unit that augments LSTM with built-in spatial attention and a revised output gating. The first enables LSTA to attend the feature regions of interest while the second constraints it to expose a distilled view of internal memory[1]. Our study confirms that it is effective to improve the output gating of recurrent unit since it does not only affect prediction overall but controls the recurrence, being responsible for a smooth and focused tracking of the latent memory state across the sequence. Our main contributions can be summarized as follows:

- We present Long Short-Term Attention (LSTA), a new recurrent unit that addresses shortcomings of LSTM when the discriminative information in the input sequence can be spatially localized;

- We deploy LSTA into a two stream architecture with cross-modal fusion, a novel control of the bias parameter of one modality by using the other;

- We report an ablation analysis of the model and evaluate it on egocentric activity recognition, providing state-of-the-art results in four public datasets.

2. Related Work

We discuss the most relevant deep learning methods for addressing egocentric vision problems in this section.

2.1. First Person Action Recognition

The works of [19, 28, 41] train specialized CNN for hand segmentation and object localization related to the activities to be recognized. These methods base on specialized pre-training for hand segmentation and object detection networks, requiring high amounts of annotated data for that purpose. Additionally, they just base on single RGB images for encoding appearance without considering temporal information. In [22, 38] features are extracted from a series of frames to perform temporal pooling with different operations, including max pooling, sum pooling, or histogram of gradients. Then, a temporal pyramid structure allows the encoding of both long term and short term characteristics. However, all these methods do not take into consideration the temporal order of the frames. Techniques that use a

recurrent neural network such as Long Short-Term Memory (LSTM) [2, 34] and Convolutional Long Short-Term Memory (ConvLSTM) [29, 30] are proposed to encode the temporal order of features extracted from a sequence of frames. Sigurdsson *et al*. [26] proposes a triplet network to develop a joint representation of paired third person and first person videos. Their method can be used for transferring knowledge from third person domain to first person domain thereby partially solving the problem of lack of large first person datasets. Tang *et al*. [32, 33] add an additional stream that accepts depth maps to the two stream network for first person action recognition, thereby enabling the network to encode 3D information present in the scene. Li *et al*. [13] propose a deep neural network to jointly predict the gaze and action from first person videos, which requires gaze information during training.

Majority of the state-of-the-art techniques rely on additional annotations such as hand segmentation, object bounding box or gaze information. This allows the network to concentrate on the relevant regions in the frame and helps in distinguishing each activity from one another better. However, manually annotating all the frames of a video with these information is impractical. For this reason, development of techniques that can identify the relevant regions of a frame without using additional annotations is crucial.

2.2. Attention

Attention mechanism was proposed for focusing attention on features that are relevant for the task to be recognized. This includes [30, 13, 24] for first person action recognition, [1, 18, 35] for image and video captioning and [20, 1, 16] for visual question answering. The works of [23, 8, 31, 30, 39, 13] use an attention mechanism for weighting spatial regions that are representative for a particular task. Sharma *et al*. [23] and Zhang *et al*. [39] generate attention masks implicitly by training the network with video labels. Authors of [8, 31, 30] use top-down attention generated from the prior information encoded in a CNN pretrained for object recognition while [13] uses gaze information for generating attention. The work of [21, 24] uses attention for weighting relevant frames, thereby adding temporal attention. This is based on the idea that not all frames present in a video are equally important for understanding the action being carried out. In [21] a series of temporal attention filters is learnt that weight frame level features depending on their relevance for identifying actions. [24] uses change in gaze for generating the temporal attention. [15, 5] apply attention on both spatial and temporal dimensions to select relevant frames and the regions present in them.

Most of the existing techniques for generating spatial attention in videos consider each frame independently. Since videos are of sequential nature and have an absolute temporal consistency, per frame processing results in the loss of

---
[1]Code available in here.

valuable information.

2.3. Relation to state-of-the-art alternatives

The proposed LSTA method generates the spatial attention map in a top-down fashion utilizing prior information encoded in a CNN pre-trained for object recognition and another pre-trained for action recognition. [30] proposes a similar top-down attention mechanism. However, they generate the attention map independently in each frame whereas in the proposed approach, the attention map is generated in a sequential manner. This is achieved by propagating the attention map generated from past frames across time by maintaining an internal state for attention. Our method uses attention on the motion stream followed by a cross-modal fusion of the appearance and motion streams, thereby enabling both streams to interact earlier in the layers to facilitate flow of information between them. [39] proposes an attention mechanism that takes in to consideration the inputs from past frames. Their method is based on bottom-up attention and generates a single weight matrix which is trained with the video level label. However, the proposed method generates attention, based on the input, from a pool of attention maps which are learned using video level label alone.

3. Analysis of LSTM

LSTM is the widely adopted neuron design for processing and/or predicting sequences. A latent memory state $\mathbf{c_t}$ is tracked across a sequence with a forget-update mechanism

$$\mathbf{c_t} = f \odot \mathbf{c_{t-1}} + i \odot c \qquad (1)$$

where (f, i) have a gating function on the previous state $\mathbf{c_{t-1}}$ and an innovation term c. (f, i, c) are parametric functions of input $\mathbf{x_t}$ and a gated non-linear view of previous memory state $\mathbf{o_{t-1}} \odot \eta(\mathbf{c_{t-1}})$

$$(i, f, \mathbf{o_t}, c) = (\sigma, \sigma, \sigma, \eta)(W \cdot [\mathbf{x_t}, \mathbf{o_{t-1}} \odot \eta(\mathbf{c_{t-1}})]) \quad (2)$$

The latter, referred to as hidden state $\mathbf{h_t} = \mathbf{o_t} \odot \eta(\mathbf{c_t})$, is often exposed to realize a sequence prediction. For sequence classification instead, the final memory state can be used as a fixed-length descriptor of the input sequence.

Two features of LSTM design explain its success. First, the memory update (Eq. 1) is flexibly controlled by (f, i): a state can, in a single iteration, be erased $(0, 0)$, reset $(0, 1)$, left unchanged $(1, 0)$, or progressively memorize new input. $(1, 1)$ resembles residual learning [10], a key design pattern in very deep networks - depth here translates to sequence length. Indeed, LSTMs has strong gradient flow and learn long-term dependencies [11]. Second, the gating functions (Eq. 2) are learnable neurons and their interaction in memory updating is transparent (Eq. 1). When applied to video classification, a few limitations are to be discussed:

1. *Memory*. Standard LSTMs use fully connected neuron gates and consequently, the memory state is unstructured. This may be desired *e.g.* for image captioning where one modality (vision) has to be translated into another (language). For video classification it might be advantageous to preserve the spatial layout of images and their convolutional features by propagating a memory tensor instead. ConvLSTM [25] addresses this shortcoming through convolutional gates in the LSTM.

2. *Attention*. The discriminative information is often confined locally in the video frame. Thus, not all convolutional features are equally important for recognition. In LSTMs the filtering of irrelevant features (and memory) is deferred to the gating neurons, that is, to a linear transformation (or convolution) and a non-linearity. Attention neurons were introduced to suppress activations from irrelevant features ahead of gating. We augment LSTM with built-in attention that directly interacts with the memory tracking in Sec. 4.1.

3. *Output gating*. Output gating not only impacts sequence prediction but it critically affects memory tracking too, *cf.* Eq 2. We replace the output gating neuron of LSTM with a high-capacity neuron whose design is inspired by that of attention. There is indeed a relation among them, we make this explicit in Sec. 4.2.

4. *External bias control*. The neurons in Eq. 2 have a bias term that is learnt from data during training, and it is fixed at prediction time in standard LSTM. We leverage on adapting the biases based on the input video for each prediction. State-of-the-art video recognition is realized with two-stream architectures, we use flow stream to control appearance biases in Sec. 5.3

4. Long Short-Term Attention

We present LSTA recurrent unit in Fig. 1. The model is

$$\nu_a = \varsigma(\mathbf{x_t}, w_x) \qquad (3)$$

$$(i_a, f_a, \mathbf{s_t}, a) = (\sigma, \sigma, \sigma, \eta)(W_a * [\nu_a, \mathbf{s_{t-1}} \odot \eta(\mathbf{a_{t-1}})]) \qquad (4)$$

$$\mathbf{a_t} = f_a \odot \mathbf{a_{t-1}} + i_a \odot a \qquad (5)$$

$$s = \text{softmax}(\nu_a + \mathbf{s_t} \odot \eta(\mathbf{a_t})) \qquad (6)$$

$$(i_c, f_c, c) = (\sigma, \sigma, \eta)(W_c * [s \odot \mathbf{x_t}, \mathbf{o_{t-1}} \odot \eta(\mathbf{c_{t-1}})]) \qquad (7)$$

$$\mathbf{c_t} = f_c \odot \mathbf{c_{t-1}} + i_c \odot c \qquad (8)$$

$$\nu_c = \varsigma(\mathbf{c_t}, w_c + w_o \cdot \epsilon(s \odot \mathbf{x_t})) \qquad (9)$$

$$\mathbf{o_t} = \sigma(W_o * [\nu_c \odot \mathbf{c_t}, \mathbf{o_{t-1}} \odot \eta(\mathbf{c_{t-1}})]) \quad (10)$$

Eqs. 3-6 implement our recurrent attention as detailed in Sec. 4.1, Eqs. 9-10 is our coupled output gating of Sec. 4.2. Bold symbols represent the recurrent variables: $(\mathbf{a_t}, \mathbf{s_t})$ of shape $N \times 1$, $(\mathbf{c_t}, \mathbf{o_t})$ of shape $N \times K$. Trainable parameters are: (W_a, W_c) are both K convolution kernels,
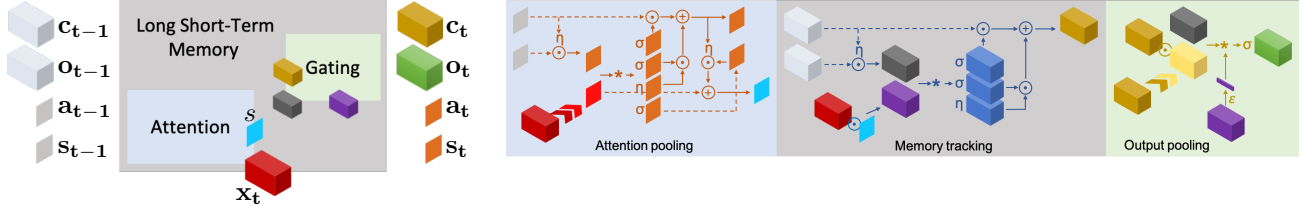
Figure 1: LSTA enhances LSTM with built-in attention (blue block) and flexible output gating (green block). Standard recurrence (gray block, Eqs. 7-8) tracks a memory state $\mathbf{c_t}$ using spatially attended features $s \odot \mathbf{x_t}$ and a gated view of previous memory $\mathbf{o_{t-1}} \odot \eta(\mathbf{c_{t-1}})$. To compute s (Eq. 6) we introduce a second internal state $\mathbf{a_t}$ to track attention (Eqs. 4-5) with the pooling scheme of Sec. 4.1 (Eq. 3). We use pooling (Eqs. 9) again to expose the output gate (Eqs. 10) in Sec. 4.2.

(w_a, w_c) have shape $K \times C$, w_o has shape $C \times C$. N, K, C are introduced below. σ, η are sigmoid and tanh activation functions, $*$ is convolution, $\cdot$ is tensor product, $\odot$ is pointwise multiplication. ς, π are from the pooling model presented next. For a sequence prediction task, a hidden state $\mathbf{h}_t = \mathbf{o}_t \odot \eta(\mathbf{c}_t)$ can be exposed at each iteration.

4.1. Attention Pooling

Given a matrix view $\mathbf{x}_{ik}$ of convolutional feature tensor $\mathbf{x}$ where i indexes one of N spatial locations and k indexes one of K feature planes, we aim at suppressing those activations $\mathbf{x}_i$ that are uncorrelated with the recognition task. That is, we seek a $\varsigma(\mathbf{x}, w)$ of shape $1 \times N$ such that parameters w can be tuned in a way that $\varsigma(\mathbf{x}, w) \odot \mathbf{x}$ are the discriminative features for recognition. For egocentric activity recognition these can be from objects, hands, or implicit patterns representing object-hand interactions during manipulation.

Our design of $\varsigma(\mathbf{x}, w)$ is grounded on the assumption that there is a limited number of pattern categories that are relevant for an activity recognition task. Each category itself can, however, instantiate patterns with high variability during and across executions. We therefore want ς to select from a pool of category-specific mappings, based on the current input $\mathbf{x}$. We want both the selector and the pool of mappings be learnable and self-consistent, and realized with fewer tunable parameters.

A selector with parameters w maps an image features $\mathbf{x}$ into a category-score space $\mathcal{C}$ from which the category $c^* \in \mathcal{C}$ obtaining the highest score is returned. Our selector is of the form $c^* = \arg\max_c \pi(\epsilon(\mathbf{x}), \theta_c)$ where ϵ is a reduction and $\theta_c \in w$ are the parameters for scoring $\mathbf{x}$ against category c. If π is chosen to be equivariant to reduction ϵ then $\pi(\epsilon(\mathbf{x}), \theta_c) = \epsilon(\pi(\mathbf{x}, \theta_c))$ and we can use $\{\epsilon^\perp(\pi(\cdot, \theta_c)), c \in \mathcal{C}\}$ as the pool of category-specific mappings associated to ϵ. Here $\epsilon^\perp$ denotes the ϵ-orthogonal reduction, e.g. if ϵ is max-pooling along one dimension then $\epsilon^\perp$ is max-pooling along the other dimensions. That is, our pooling model is determined by the triplet

$$(\varsigma) = (\epsilon, \pi, \{\theta_c\}), \quad \pi \text{ is } \epsilon\text{-equivariant} \quad (11)$$

and realized on a feature tensor $\mathbf{x}$ by

$$\varsigma(\mathbf{x}, \{\theta_c\}) = \epsilon^\perp(\pi(\mathbf{x}, \theta_{c^*})) \quad (12)$$
$$\text{where } c^* = \arg\max_c \pi(\epsilon(\mathbf{x}), \theta_c) \quad (13)$$

In our model we choose

$$\epsilon(\mathbf{x}) \leftarrow \text{spatial average pooling}$$
$$\pi(\epsilon, \theta_c) \leftarrow \text{linear mapping}$$

so $\varsigma(\mathbf{x}, \{\theta_c\})$ is a differentiable spatial mapping, i.e., we can use ς as a trainable attention model for $\mathbf{x}$. This is related to class activation mapping [40] introduced for discriminative localization. Note however that, in contrast to [40] that uses strong supervision to train the selector directly, we leverage video-level annotation to implicitly learn an attention mechanism for video classification. Our formulation is also a generalization: other choices are possible for the reduction ϵ, and the use of differentiable structured layers [12] in this context are an interesting direction for future work.

To inflate attention in LSTA, we introduce a new state tensor $\mathbf{a_t}$ of shape $N \times 1$. Its update rule is that of standard LSTM (Eq. 5) with gatings $(f_a, i_a, \mathbf{s_t})$ and innovation a computed from the pooled $\nu_a = \varsigma(\mathbf{x_t}, w_a)$ as input (Eq. 4). We compute the attention tensor s using the hidden state $\mathbf{s_t} \odot \eta(\mathbf{a_t})$ as residual (Eq. 6), followed by a softmax calibration. Eqs. 7-10 implement the LSTA memory update based on the filtered input $s \odot \mathbf{x_t}$, this is described next.

4.2. Output Pooling

If we analyze standard LSTM Eq. 2 with input $s \odot \mathbf{x_t}$ instead of $\mathbf{x_t}$, it becomes evident that $\mathbf{o_{t-1}}$ (output gating) has on $\eta(\mathbf{c_{t-1}})$ a same effect as s (attention) has on $\mathbf{x_t}$. Indeed, in Eq. 7 the gatings and innovation are all computed from $[s \odot \mathbf{x_t}, \mathbf{o_{t-1}} \odot \eta(\mathbf{c_{t-1}})]$. We build upon this analogy to enhance the output gating capacity of LSTA and, consequently, its forget-update behavior of memory tracking.

We introduce attention pooling in the output gating update. Instead of computing $\mathbf{o_t}$ as by Eq. 2 we replace $s \odot \mathbf{x_t}$

with $\nu_c \odot \mathbf{c_t}$ to obtain update Eqs. 9-10, that is

$$\sigma(W_o * [s \odot \mathbf{x_t}, \mathbf{o_{t-1}} \odot \eta(\mathbf{c_{t-1}})]) \quad \leftarrow \quad \text{standard gating}$$

$$\sigma(W_o * [\nu_c \odot \mathbf{c_t}, \mathbf{o_{t-1}} \odot \eta(\mathbf{c_{t-1}})])$$
$$\text{with } \nu_c = \varsigma(\mathbf{c_t}, w_c + w_o \cdot \epsilon(s \odot \mathbf{x_t})) \quad \leftarrow \quad \text{output pooling}$$

This choice is motivated as follows. We want to preserve the recursive nature of output gating, which is we keep right-concatenating $\mathbf{o_{t-1}} \odot \eta(\mathbf{c_{t-1}})$ to obtain the $2N \times K$-shaped tensor to convolve and tanh point-wise. Since the new memory state $\mathbf{c_t}$ is available at this stage, which already integrates $s \odot \mathbf{x_t}$, we can use this for left-concatenating instead of the raw attention-pooled input tensor. We can even produce a filtered version $\nu_c \odot \mathbf{c_t}$ of it if we introduce a second attention pooling neuron for localizing the actual discriminative memory component of $\mathbf{c_t}$, that is via ν_c, Eq. 9. Note that $\mathbf{c_t}$ integrates information from past memory updates by design, so localizing current activations this is pretty much required here. Consequently, and in contrast to feature tensors $\mathbf{x}$, the memory activations might not be well localized spatially. We thus use a slightly different version of Eq. 12 for output pooling, we remove $\epsilon^\perp$ to obtain a full-rank $N \times K$-shaped attention tensor ν_c.

To further enhance active memory localization, we use $s \odot \mathbf{x_t}$ to control the bias term of attention pooling, Eq. 9. We apply a reduction $\epsilon(s \odot \mathbf{x_t})$ followed by a linear regression with learnable parameters w_o to obtain the instance-specific bias $w_o \cdot \epsilon(s \odot \mathbf{x_t})$ for activation mapping. Note that ϵ is the reduction associated to ς so this is consistent. We will use a similar idea in Sec. 5.3 for cross-modal fusion in two-stream architecture. Our ablation study in Sec. 6.4 confirms that this further coupling of $\mathbf{c_t}$ with $\mathbf{x_t}$ boosts the memory distillation in the LSTA recursion, and consequently its tracking capability, by a significant margin.

5. Two Stream Architecture

In this section, we explain our network architecture for egocentric activity recognition incorporating the LSTA module of Sec. 4. Like the majority of the deep learning methods proposed for action recognition, we also follow the two stream architecture; one stream for encoding appearance information from RGB frames and the second stream for encoding motion information from optical flow stacks.

5.1. Attention on Appearance Stream

The network consists of a ResNet-34 pre-trained on imageNet for image recognition. We use the output of the final convolutional layer in layer 4 as the input of the LSTA module. From this frame level features, LSTA generates the attention map which is used to weight the input features. We select 512 as the depth of LSTA memory and all the gates use a kernel size of 3×3.

We follow a two stage training. In the first stage, the fully-connected (FC) layer in the classifier and the LSTA modules are trained while in the second stage, the convolutional layers in the layer 4 and the FC layer of ResNet-34 along with the layers trained in stage 1 are trained.

5.2. Attention on Motion Stream

We use a network trained on optical flow stacks for explicit motion encoding. For this, we use a ResNet-34 network. The network is first trained for action verbs (take, put, pour, open, etc.) recognition using an optical flow stack of 5 frames. We average the weights in the input convolutional layer of an imagenet pre-trained network and replicate it 10 times to initialize the input layer. This is analogous to the imageNet pre-training done on the appearance stream. The network is then trained for activity recognition as follows. We use the action-pretrained ResNet-34 FC weights as the parameter initialization of attention pooling (Eqs. 12-13) on Layer-4 flow features. We use this attention map to weight the features for classification. Since the activities are temporally located in the videos and they are not sequential in nature, we take the optical flow corresponding to the five frames located in the temporal center of the videos.

5.3. Cross-modal Fusion

Majority of the existing methods with two stream architecture perform a simple late fusion by averaging for combining the outputs from the appearance and motion streams [27, 36]. Feichtenhofer *et al.* [7] propose a convolutional pooling strategy at the output of the final convolutional layer for improved fusion of the two streams. In [6] the authors observe that adding a residual connection from the motion stream to the appearance stream enables the network to improve the joint modeling of the information flowing through the two streams. Inspired by the aforementioned observations, we propose a novel cross-modal fusion strategy in the earlier layers of the network in order to facilitate the flow of information across the two modalities thereby improving the learnt video representation.

Fig. 2 illustrates the proposed cross-modal fusion architecture for two stream networks. The appearance and motion streams are as described in Secs. 5.1 and 5.2. In this architecture, each stream is used to control the biases of other. The output of the motion stream is applied as bias to the gates of the LSTA layer. The output of the RGB CNN from all the input frames are applied to a 3D convolutional layer to obtain a summary feature of the input frames. We add a ConvLSTM in the motion stream as an embedding layer that is similar in functionality to the gates of the LSTA layer. The output of the 3D convolutional layer is then applied as bias to the gates of the ConvLSTM layer in the motion stream. In this way, each individual stream is made to influence the encoding of the other so that we have
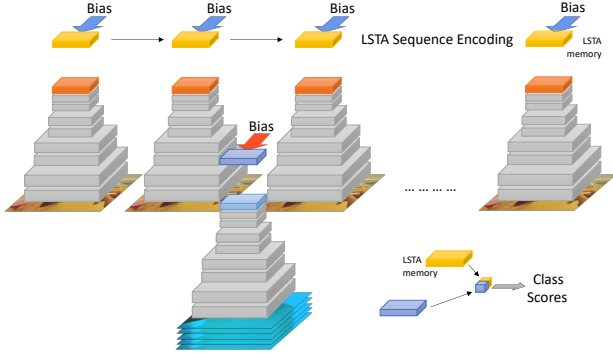
Figure 2: Cross-modal fusion in our two stream architecture: we use the stack-of-flow feature tensor to control LSTA bias, and the sequence of RGB feature tensors to control the bias of a ConvLSTM on flow features. We use ResNet-34 for feature extraction in both modalities, and perform standard late fusion for video-level prediction.

a flow of information between them deep inside the neural network. We then perform a late average fusion of the two individual streams' output to obtain the class scores.

6. Experiments and Results

6.1. Datasets

We evaluate the proposed method on four standard first person activity recognition datasets namely, GTEA 61, GTEA 71, EGTEA Gaze+ and EPIC-KITCHENS. GTEA 61 and GTEA 71 are relatively small scale datasets with 61 and 71 activity classes respectively. EGTEA Gaze+ is a recently developed large scale dataset with approximately 10K samples having 106 activity classes. EPIC-KITCHENS dataset is the largest egocentric activities dataset available now. The dataset consists of more than 28K video samples with 125 verb and 352 noun classes.

6.2. Experimental Settings

The appearance and motion networks are first trained separately followed by a combined training of the two stream cross-modal fusion network. We train the networks for minimizing the cross-entropy loss. The appearance stream is trained for 200 epochs in stage 1 with a learning rate of 0.001 which is decayed after 25, 75 and 150 epochs at a rate of 0.1. In the second stage, the network is trained with a learning rate of 0.0001 for 100 epochs. The learning rate is decayed by 0.1 after 25 and 75 epochs. We use ADAM as the optimization algorithm. 25 frames uniformly sampled from the videos are used as input. The number of classes used in the output pooling (w_c in 4.2) is chosen as 100 for GTEA 61 and GTEA 71 datasets after empirical evaluation on the fixed split of GTEA 61. For EGTEA Gaze+ and EPIC-KITCHENS datasets, the value is scaled to 150 and 300 respectively, in accordance with the relative

increase in the number of activity classes.

For the pre-training of the motion stream on action classification task, we use a learning rate of 0.01 which is reduced by 0.5 after 75, 150, 250 and 500 epochs and is trained for 700 epochs. In the activity classification stage, we train the network for 500 epochs with a learning rate of 0.01. The learning rate is decayed after 50 and 100 epochs by 0.5. SGD algorithm is used for optimizing the parameter updates of the network.

The two stream network is trained for 200 epochs for GTEA 61 and GTEA 71 datasets while EGTEA is trained till 100 epochs, with a learning rate of 0.01 using ADAM algorithm. Learning rate is reduced by 0.99 after each epoch. We use a batch size of 32 for all networks. We use random horizontal flipping and multi-scale corner cropping techniques proposed in [36] during training and the center crop of the frame is used during the evaluation stage.

6.3. Ablation Study

An extensive ablation analysis[2] has been carried out, on the fixed split of GTEA 61 dataset, to determine the performance improvement obtained by each component of the proposed method. The results are shown in Tab. 1. The table compares the performance of RGB and two stream networks on the top and bottom sections respectively. We choose a network with vanilla ConvLSTM as the baseline since LSTA without attention and output pooling converges to the standard ConvLSTM. The baseline model results in an accuracy of 51.72%. We then analyze the impact of each of the contributions explained in Sec 4. We first analyze the effect of output pooling on the baseline. By adding output pooling the performance is improved by 8%. We analyzed the classes that are improved by adding output pooling over the baseline model and observe that the major improvement is achieved by predicting the correct action classes. Output pooling enables the network to propagate a filtered a version of the memory which is localized on the most discriminative components.

Adding attention pooling to the baseline improves the performance by 14%. Attention pooling enables the network to identify the relevant regions in the input frame and to maintain a history of the relevant regions seen in the past frames. This enables the network to have a smoother tracking of attentive regions. Detailed analysis show that attention pooling enables the network to correctly classify activities with multiple objects. It should be noted that this is equivalent to a network with two ConvLSTMs, one for attention tracking and one for frame level feature tracking.

Incorporating both attention and output pooling to the baseline results in a gain of 16%. By analyzing the top improved classes, we found that the model has increased its capacity to correctly classify both actions and objects. By

---

[2]Detailed analysis available in the appendix.

Ablation	Accuracy (%)
Baseline	51.72
Baseline + output pooling	62.07
Baseline + attention pooling	66.38
Baseline + pooling	68.1
LSTA	**74.14**
LSTA two stream late fusion	78.45
LSTA two stream cross-modal fusion	**79.31**

Table 1: Ablation analysis on GTEA 61 fixed split.

adding bias control, as explained in section 4, we obtain the proposed LSTA model and gains an additional improvement of 6% in recognition accuracy.

Compared to the network with the vanilla ConvLSTM, LSTA achieves an improvement of 22%. From the previous analyses we have seen the importance of attention pooling and output pooling present in LSTA. This enables the network to focus on encoding the features more relevant for the concrete classification task. Detailed analysis shows ConvLSTM confuses with both activities involving same action with different objects as well as activities consisting of different action with same objects. With the attention mechanism, LSTA weights the most discriminant features, thereby allowing the network to distinguish between the different activity classes.

We also evaluated the performance improvement achieved by applying attention to the motion stream. The baseline is a ResNet-34 pre-trained on actions followed by training for activities. We obtained an accuracy of 40.52% for the network with attention compared to the 36.21% of the baseline. Fig. 3 (fourth row) visualizes the attention map generated by the network. For visualization, we overlay the resized attention map on the RGB frames corresponding to the optical flow stack used as input. From the figure, it can be seen that the network generates the attention map around/near the hands, where the discriminant motion is occurring, thereby enabling the network to recognize the activity undertaken by the user. It can also be seen that the attention maps generated by the appearance stream and the flow stream are complementary to each other; appearance stream focuses on the object regions while the motion stream focuses on hand regions. We also analyzed the classes where the network with attention performs better compared to the standard flow network and found that the network with attention is able to recognize actions better than the standard network. This is because the attention mechanism enables the network to focus on regions where motion is occurring in the frame.

Next we compare the performance obtained by adding the cross-modal fusion technique explained in section 5.3 over traditional late fusion two stream approach. The cross-modal fusion approach improves by 1% over late fusion. Analysis shows that the cross-modal fusion approach is able

to correctly identify activities with same objects. The fifth and sixth rows of Fig. 3 visualize the attention maps generated after two stream cross-modal fusion training. It can be seen that the motion stream attention expands to regions containing objects. This validates the effect of cross-modal fusion where the two networks are made to interact deep inside the network.

6.4. Comparative Analysis

In this section, we compare the performance of LSTA over two closely related methods, namely, eleGAtt [39] and ego-rnn [30]. Results are shown in Tab. 2. EleGAtt is an attention mechanism which can be applied to any generic RNN using its hidden state for generating the attention map. We evaluated eleGAtt on LSTM, consisting of 512 hidden units, with the same training setting as LSTA for fair comparison. EleGAtt learns a single weight matrix for generating the attention map irrespective of the input whereas LSTA generates the attention map from a pool of weights which are selected in a top-down manner based on input. This enables the selection of a proper attention map for each input activity class. This leads to a performance gain of 13% over eleGAtt. Analyzing the classes with the highest improvement by LSTA compared to eleGAtt reveals that eleGAtt fails in identifying the object while correctly classifying the action. Ego-rnn [30] derives an attention map generated from class activation map to weight the discriminant regions in the image which are then applied to a ConvLSTM cell for temporal encoding. It generates a per frame attention map which has no dependency on the information present in the previous frames. This can result in selecting different objects in adjacent frames. On the contrary, LSTA uses an attention memory to track the previous attention maps enabling their smooth tracking. This results in a 10% improvement obtained by LSTA over ego-rnn. Detailed analysis on the classification results show that ego-rnn struggles to classify activities involving multiple objects. Since the attention map generated in each frame is independent of the previous frames, the network fails to track previously activated regions, thereby resulting in wrong predictions. This is further illustrated by visualizing the attention maps produced by ego-rnn and LSTA in Fig. 3. From the figure, one can see that ego-rnn (second row) fails to identify the relevant object in the case of close chocolate example and it failed to track the object in the final frames in the case of the scoop coffee example. LSTA with cross-modal fusion performs 2% better than ego-rnn two stream. Analysis shows that the cross-modal fusion approach is able to correctly identify activities with same objects.

6.5. State-of-the-art comparison

Our approach is compared against the state-of-the-art methods on Tab. 3. The methods listed in the first sec-
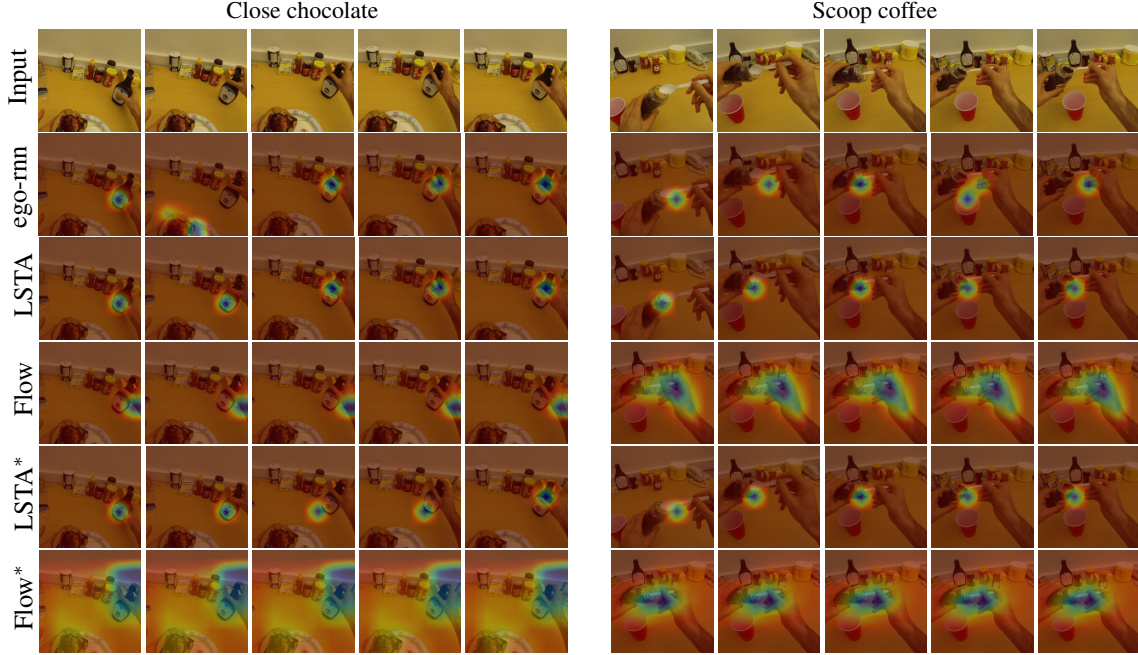
Figure 3: Attention maps generated by ego-rnn (second row) and LSTA (third) for two video sequences. We show the 5 frames that are uniformly sampled from the 25 frames used as input to the corresponding networks. Fourth row shows the attention map generated by the motion stream. Fifth and sixth rows show the attention map generated by the appearance and flow streams after two stream cross-modal training. For flow, we visualize the attention map on the five frames corresponding to the optical flow stack given as input. (*: Attention map obtained after two stream cross-modal fusion training).

Method	Accuracy (%)
eleGAtt [39]	59.48
ego-rnn [30]	63.79
LSTA	**74.14**
ego-rnn two stream [30]	77.59
LSTA two stream	**79.31**

Table 2: Comparative analysis on GTEA 61 fixed split.

Methods	GTEA61*	GTEA61	GTEA71	EGTEA
Li *et al*. [14]**	66.8	64	62.1	46.5
Ma *et al*. [19]**	75.08	73.02	73.24	-
Li *et al*. [13]**	-	-	-	53.3
Two stream [27]	57.64	51.58	49.65	41.84
I3D [3]	-	-	-	51.68
TSN [36]	67.76	69.33	67.23	55.93
eleGAtt [39]	59.48	66.77	60.83	57.01
ego-rnn [30]	77.59	79	77	60.76
LSTA-RGB	74.14	71.32	66.16	57.94
LSTA	**79.31**	**80.01**	**78.14**	**61.86**

Table 3: Comparison with state-of-the-art methods on popular egocentric datasets, we report recognition accuracy in %. (*: fixed split; **: trained with strong supervision).

tion of the table uses strong supervision signals such as gaze [14, 13], hand segmentation [19] or object bounding boxes [19] during the training stage. Two stream [27], I3D [3] and TSN [36] are methods proposed for action recognition from third person videos while all other methods except eleGAtt [39] are proposed for first-person activity recognition. eleGAtt [39] is proposed as a generic method for incorporating attention mechanism to any RNN modules. From the table, we can see that the proposed method outperforms all the existing methods for egocentric activity recognition.

6.6. EPIC-KITCHENS

In this dataset, the labels are provided in the form of `verb` and `noun`, which are combined to form an activity class. The fact that not all combinations of verbs and nouns are feasible and that not all test classes might have a representative training sample make it a challenging problem. We train the network for multi-task classification with

`verb` and `noun` and `activity` supervision. We use activity classifier activations to control the bias of `verb` and `noun` level classifiers. The dataset provides two evaluation settings, seen kitchens (S1) and unseen kitchens (S2). We obtained an accuracy of 30.16% (S1) and 15.88% (S2) using just RGB frames. The best performing baseline is a two stream TSN that achieves 20.54% (S1) and 10.89% (S2) [4]. Our model is particularly strong on `verb` prediction (58%) where we gain +10% points over TSN. `verb` in this context is typically describing actions that develop into an activity over time, confirming once more LSTA efficiently learns

encoding of sequences with localized patterns.

7. Conclusion

We presented LSTA that extends LSTM with two core features: 1) attention pooling that spatially filters the input sequence and 2) output pooling that exposes a distilled view of the memory at each iteration. As shown in a detailed ablation study, both contributions are essential for a smooth and focused tracking of a latent representation of the video to achieve superior performance in classification tasks where the discriminative features can be localized spatially. We demonstrate its practical benefits for egocentric activity recognition with a two stream CNN-LSTA architecture featuring a novel cross-modal fusion and we achieve state-of-the-art accuracy on four standard benchmarks.

References

[1] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proc. CVPR*, 2018. 2

[2] C. Cao, Y. Zhang, Y. Wu, H. Lu, and J. Cheng. Egocentric gesture recognition using recurrent 3d convolutional neural networks with spatiotemporal transformer modules. In *Proc. ICCV*, 2017. 2

[3] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. CVPR*, 2017. 8

[4] D. Damen, H. Doughty, G. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling egocentric vision: The epic-kitchens dataset. In *Proc. ECCV*, 2018. 8, 11

[5] W. Du, Y. Wang, and Y. Qiao. Recurrent spatial-temporal attention network for action recognition in videos. *IEEE Transactions on Image Processing*, 27(3):1347–1360, 2018. 2

[6] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Proc. NIPS*, 2016. 5

[7] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proc. CVPR*, 2016. 5

[8] R. Girdhar and D. Ramanan. Attentional pooling for action recognition. In *Proc. NIPS*, 2017. 2

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proc. ICCV*, 2017. 1

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proc. CVPR*, 2016. 1, 3

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. 3

[12] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix Backpropagation for Deep Networks with Structured Layers. In *Proc. CVPR*, 2015. 4

[13] Y. Li, M. Liu, and J. Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *Proc. ECCV*, 2018. 1, 2, 8

[14] Y. Li, Z. Ye, and J. Rehg. Delving into Egocentric Actions. In *Proc. CVPR*, 2015. 8

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

[16] J. Liang, L. Jiang, L. Cao, L. Li, and A. Hauptmann. Focal visual-text attention for visual question answering. In *Proc. CVPR*, 2018. 2

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg. Ssd: Single shot multibox detector. In *Proc. ECCV*, 2016. 1

[18] C. Ma, A. Kadav, I. Melvin, Z. Kira, G. AlRegib, and H. Graf. Attend and interact: Higher-order object interactions for video understanding. In *Proc. CVPR*, 2018. 2

[19] M. Ma, H. Fan, and K. Kitani. Going deeper into first-person activity recognition. In *Proc. CVPR*, 2016. 2, 8

[20] D. Nguyen and T. Okatani. Improved fusion of visual and language representations by dense symmetric co-attention for visual question answering. In *Proc. CVPR*, 2018. 2

[21] A. Piergiovanni, C. Fan, and M. Ryoo. Learning latent subevents in activity videos using temporal attention filters. In *Proc. AAAI Conference on Artificial Intelligence*, 2017. 2

[22] M. Ryoo, B. Rothrock, and L. Matthies. Pooled motion features for first-person videos. In *Proc. CVPR*, 2015. 2

[23] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. In *Proc. ICLRW*, 2015. 2

[24] Y. Shen, B. Ni, Z. Li, and N. Zhuang. Egocentric activity prediction via event modulated attention. In *Proc. ECCV*, 2018. 2

[25] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Proc. NIPS*, 2015. 3

[26] G. Sigurdsson, A. Gupta, C. Schmid, A. Farhadi, and K. Alahari. Actor and observer: Joint modeling of first and third-person videos. In *Proc. CVPR*, 2018. 2

[27] K. Simonyan and A. Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Proc. NIPS*, 2014. 5, 8

[28] S. Singh, C. Arora, and C. Jawahar. First person action recognition using deep learned descriptors. In *Proc. CVPR*, 2016. 2

[29] S. Sudhakaran and O. Lanz. Convolutional long short-term memory networks for recognizing first person interactions. In *Proc. ICCVW*, 2017. 2

[30] S. Sudhakaran and O. Lanz. Attention is all we need: Nailing down object-centric attention for egocentric activity recognition. In *Proc. BMVC*, 2018. 1, 2, 3, 7, 8, 10

[31] S. Sudhakaran and O. Lanz. Top-down attention recurrent vlad encoding for action recognition in videos. In *Proc. 17th International Conference of the Italian Association for Artificial Intelligence*, 2018. 2

[32] Y. Tang, Y. Tian, J. Lu, J. Feng, and J. Zhou. Action recognition in rgb-d egocentric videos. In *Proc. ICIP*, 2017. 2

[33] Y. Tang, Z. Wang, J. Lu, J. Feng, and J. Zhou. Multi-stream deep neural networks for rgb-d egocentric action recognition.

IEEE Transactions on Circuits and Systems for Video Technology, 2018. 2

[34] S. Verma, P. Nagar, D. Gupta, and C. Arora. Making third person techniques recognize first-person actions in egocentric videos. In *Proc. ICIP*, 2018. 2

[35] J. Wang, W. Jiang, L. Ma, W. Liu, and Y. Xu. Bidirectional attentive fusion with context gating for dense video captioning. In *Proc. CVPR*, 2018. 2

[36] 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 *Proc. ECCV*, 2016. 5, 6, 8

[37] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proc. CVPR*, 2017. 1

[38] H. Zaki, F. Shafait, and A. Mian. Modeling sub-event dynamics in first-person action recognition. In *Proc. CVPR*, 2017. 2

[39] P. Zhang, J. Xue, C. Lan, W. Zeng, Z. Gao, and N. Zheng. Adding attentiveness to the neurons in recurrent neural networks. In *Proc. ECCV*, 2018. 1, 2, 3, 7, 8, 11

[40] B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. In *Proc. CVPR*, 2016. 4

[41] Y. Zhou, B. Ni, R. Hong, X. Yang, and Q. Tian. Cascaded interactional targeting network for egocentric video analysis. In *Proc. CVPR*, 2016. 2

Appendix

This appendix provides additional details on the analysis carried out in Sec. 6 of the main manuscript, as well as more visualizations of the attention maps generated by the network.

Ablation Analysis

Figs. 4 - 7 show details of the classes which are improved by proposed LSTA variants over the baseline (ConvLSTM) and the difference of the confusion matrices. We show the top 25 improved classes in the comparison graphs and those with less number list all the improved classes. The difference of confusion matrices show the overall details of the classes which are improved. Ideally, the positive values should be in the diagonal and the negative values off-diagonal. Tab. 4 lists a breakdown of the recognition performance. For this, we compute the action recognition and object recognition performance of a network trained for activity recognition. There are some activity classes with multiple objects and these objects are combined to form a meta-object class for this analysis.

Fig. 4 compares the baseline (ConvLSTM) with a network having baseline+output pooling, as explained in Sec. 4.2. It can be seen that adding output pooling to the ConvLSTM improves the network's capability in recognizing different actions with the same objects (`take_water/pour_water,cup` and `close_water/take_water`). This confirms our hypothesis that the output gating of LSTM affects memory tracking, replacing the output gating of LSTM with the proposed output pooling technique localizes the active memory component. This improves the tracking of relevant spatio-temporal patterns in the memory and consequently boosts recognition performance. A gain of 13.79% is achieved for action recognition as shown in Tab. 4.

In Fig. 5, we can see that the network with the attention pooling described in Sec. 4.1 improves the categories with different actions and same objects as well as activity classes with multiple objects (`stir_spoon,cup/pour_sugar,spoon,cup;` `put_cheese,bread/take_bread;` `pour_coffee,spoon,cup/scoop_coffee,spoon,` etc.).

Attention helps the network to encode the features from the spatially relevant areas. This allows the network to keep a track of the active object regions and improves the performance. From Tab. 4, a gain of 20.69% is obtained for object recognition which gives further validation regarding the importance of attention.

Adding both attention pooling and output pooling further improves the network's capability in distinguishing between different actions with same objects and same actions with different objects. This is visible in Fig. 6 and also from the 13.72% and 18.1% performance gain obtained for action and object recognition, respectively.

Incorporating bias control, introduced in Sec. 4.2, to the output pooling results in the proposed method, LSTA, which further improves the capacity of the network in recognizing activities (Fig. 7). This further verifies the hypothesis in Sec. 4.2 that bias control increases the active memory localization of the network. This is also evident from Tab. 4 where an increase of 22.41% is obtained for action recognition.

It is worth noting that output pooling boosts action recognition performance more (+13.79% action vs +12,07% object) while with attention pooling the object recognition performance receives a higher gain (+12,93% vs +16,38%). Coupling attention and output pooling through bias control finally boosts performance by a significant margin on both (+22.41% vs +21,55%). This provides further evidence that the two contributions are complementary and reflects the intuitions behind the design choices of LSTA, making the improvements explainable and the benefits of each of the contributions transparently confirmed by this analysis.

Comparative Analysis

Figs. 8 - 10 compares our method with state-of-the-art alternatives discussed in Sec. 2.3, ego-rnn [30]

Method	Accuracy (%)		
	Activity	Action	Object
Baseline	51.72	65.52	57.76
Baseline+output pooling	62.07	79.31 (**+13.79**)	69.83 (+12,07)
Baseline+attention pooling	66.38	78.45 (+12,93)	74.14 (**+16,38**)
Baseline+pooling	68.1	79.31 (+13.79)	75.86 (+18,10)
LSTA	74.14	87.93 (**+22.41**)	79.31 (**+21,55**)

Table 4: Detailed ablation analysis on GTEA 61 fixed split. We compute the action and object recognition score by decomposing the action and objects from the predicted activity label.

and eleGatt [39]. Compared to ego-rnn, LSTA is capable of identifying activities involving multiple objects (`pour_mustard,hotgog,bread/pour_mustard, cheese,bread; pour_honey,cup/pour_honey, bread; put_hotdog,bread/spread_peanut, spoon,bread`, etc.). This may be attributed to the attention mechanism with memory for tracking previously attended regions, helping the network attending to the same objects in subsequent frames. From Fig. 9 it can be seen that eleGAtt-LSTM fails to identify the objects correctly (`take_mustard/take_honey; take_bread/take_spoon; take_spoon/take_honey`, etc.). This shows the attention map generated by LSTA selects more relevant regions compared to eleGAtt-LSTM .

Confusion Matrix

Figs. 11 - 13 show the confusion matrix of the LSTA (two stream cross-modal fusion) for all the datasets explained in Sec. 6.1 of the manuscript. We average the confusion matrices of each of the available train/test splits to generate a single confusion matrix representing the dataset under consideration.

EPIC-KITCHENS

We compare the recognition accuracies obtained for EPIC-KITCHENS dataset with the currently available baselines [4] in Tab. 5. As explained in Sec. 6.6 in the paper, we train the network for predicting `verb` and `noun` and `activity` classes. Our two stream cross-modal fusion model obtains an activity recognition performance of 30.33% and 16.63% on S1 and S2 settings as opposed to the 48.23% and 39.4% obtained by TSN strongest baseline (two stream). It is also worth noting that our model is strong on predicting `verb` (+11.32% points on S1 setting over strongest baseline). This indicates LSTA accurately performs encoding of sequences, indeed `verb` in this context is typically describing actions that develop into an activity over time, and this is learned effectively with LSTA just using video-level supervision.

Attention Map Visualization

Figs 14 - 18 visualize the generated attention maps for different video sequences. In Figs. 14 - 16, one can see that LSTA is able to successfully identify the relevant regions and track them across the sequences while ego-rnn misses the regions in some frames. This shows the ability of LSTA in identifying and tracking the discriminant regions that are relevant for classifying the activity category. However, in Figs. 17 and 18, the network fails to recognize the relevant regions. In both of these video sequences, the object is not present in the first few frames and the network attends to wrong regions, failing to move its attention towards the object when it appears. Since the proposed method maintains a memory of attention maps, occlusion of the relevant object in the initial frames results in the network attending to the wrong regions in the frame.
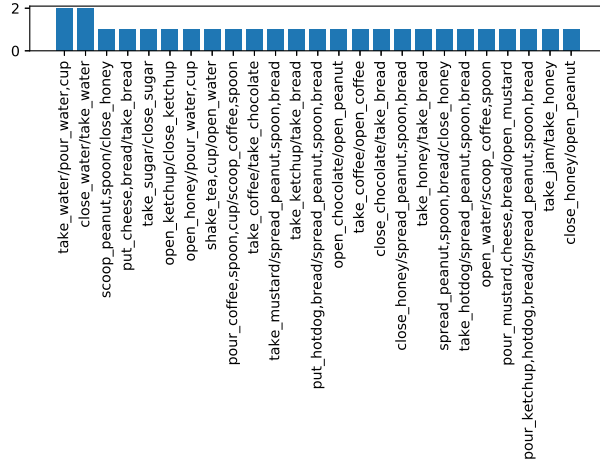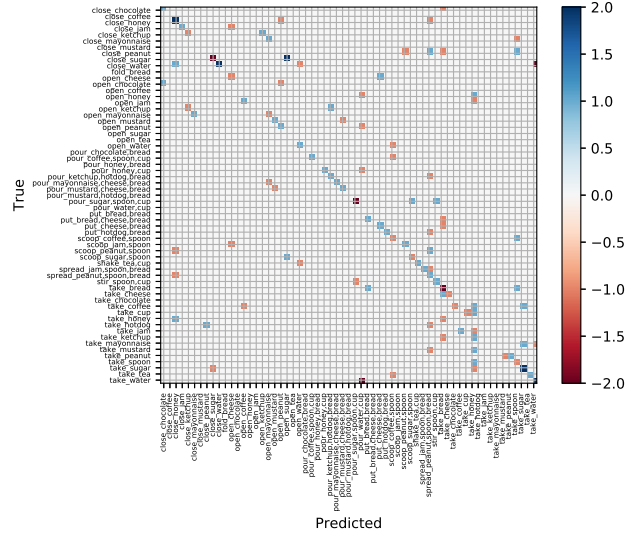
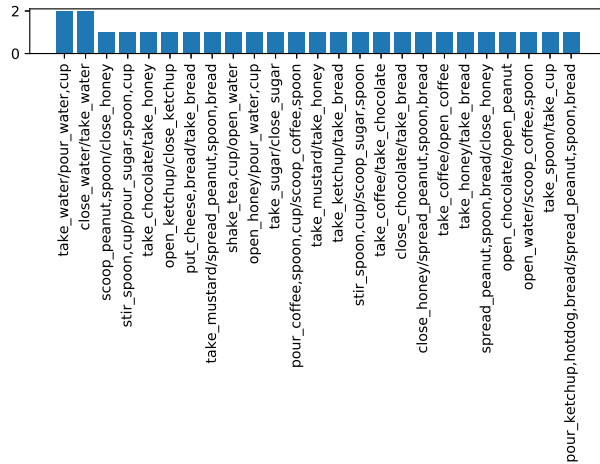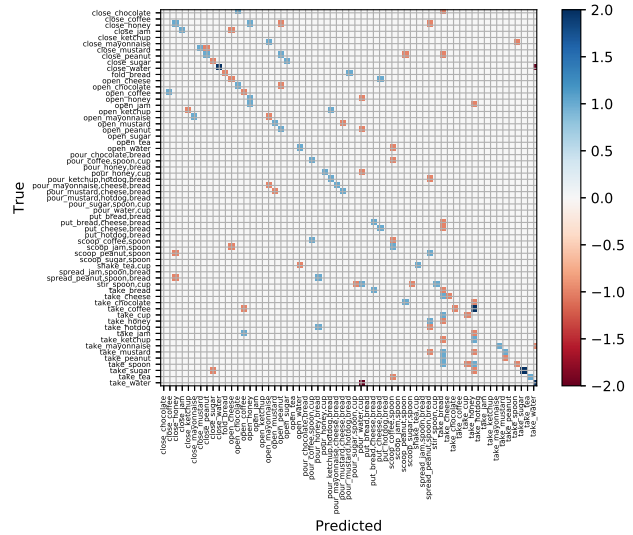(a)   (b)

Figure 4: (a) Most improvement categories by adding output pooling to the baseline on GTEA 61 fixed split. X axis labels are in the format true label (baseline + output pooling)/predicted label (baseline). Y axis shows the number of corrected samples for each class. (b) shows the difference of confusion matrices.
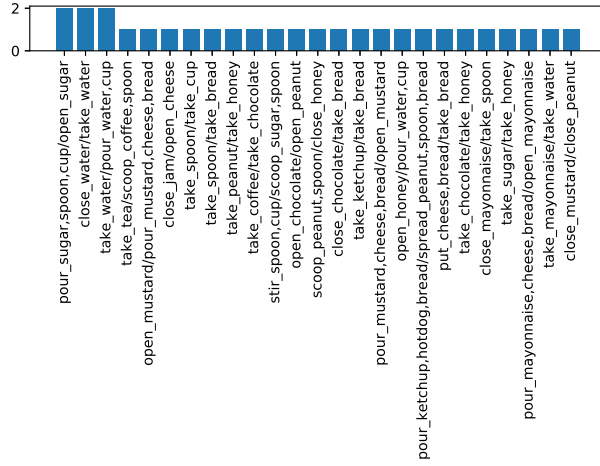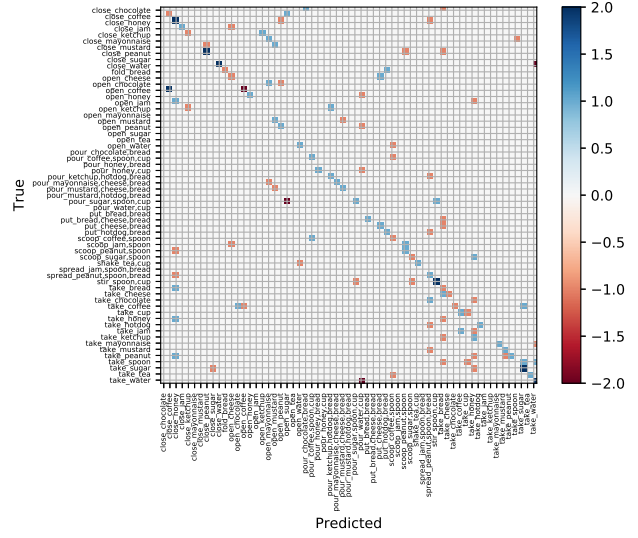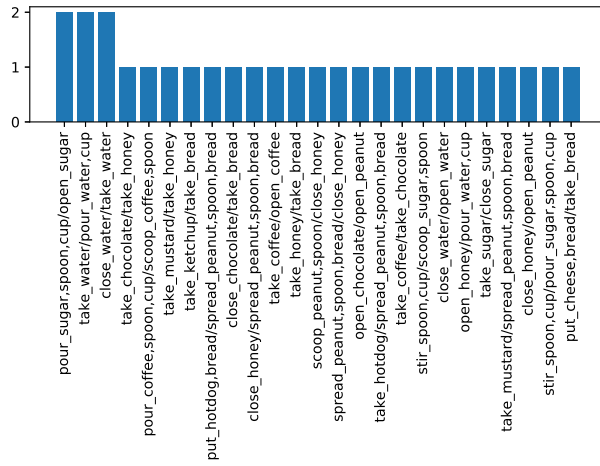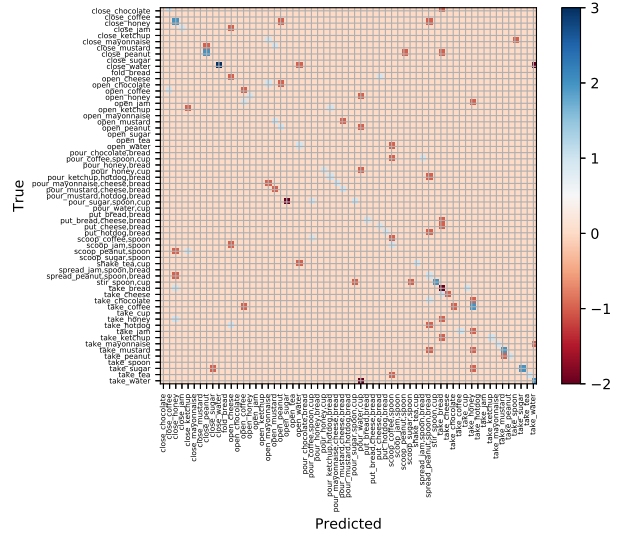

(a)   (b)

Figure 5: (a) Most improvement categories by adding attention pooling to the baseline on GTEA 61 fixed split. X axis labels are in the format true label (baseline + attention pooling)/predicted label (baseline). Y axis shows the number of corrected samples for each class. (b) shows the difference of confusion matrices.

Figure 6: Most improvement categories by adding both attention and output pooling to the baseline on GTEA 61 fixed split. X axis labels are in the format true label (baseline + pooling)/predicted label (baseline). Y axis shows the number of corrected samples for each class. (b) shows the difference of confusion matrices.

Figure 7: Most improvement categories by adding attention and output pooling with bias control (full LSTA model) to the baseline on GTEA 61 fixed split. X axis labels are in the format true label (LSTA)/predicted label (baseline). Y axis shows the number of corrected samples for each class. (b) shows the difference of confusion matrices.
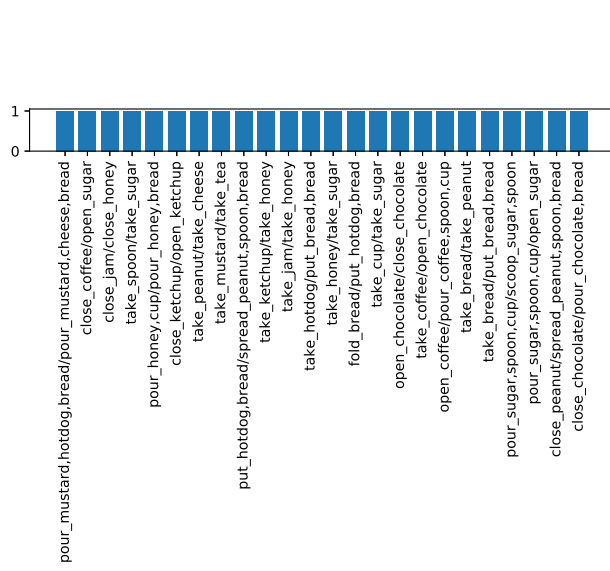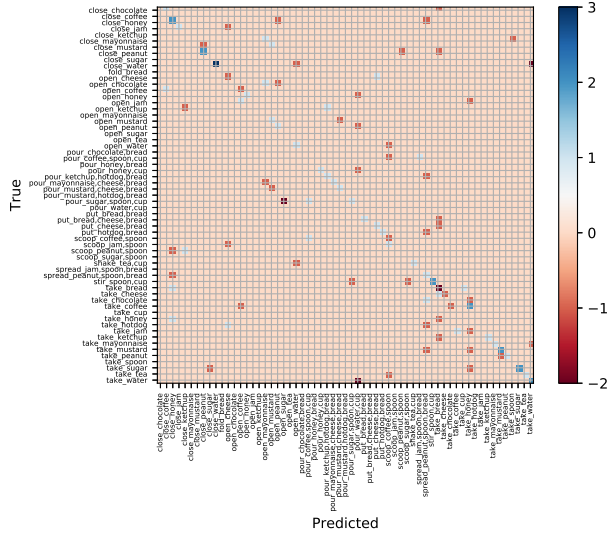
Figure 8: (a) Most improvement categories by LSTA over ego-rnn on GTEA 61 fixed split. X axis labels are in the format true label (LSTA)/predicted label (ego-rnn). Y axis shows the number of corrected samples for each class. (b) shows the difference of confusion matrices.
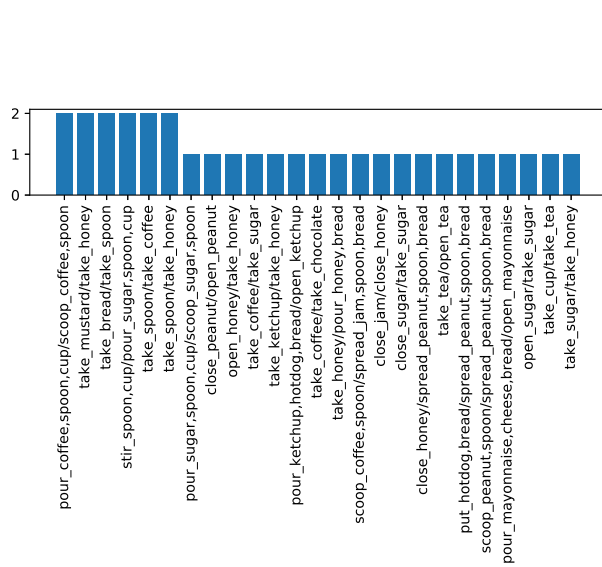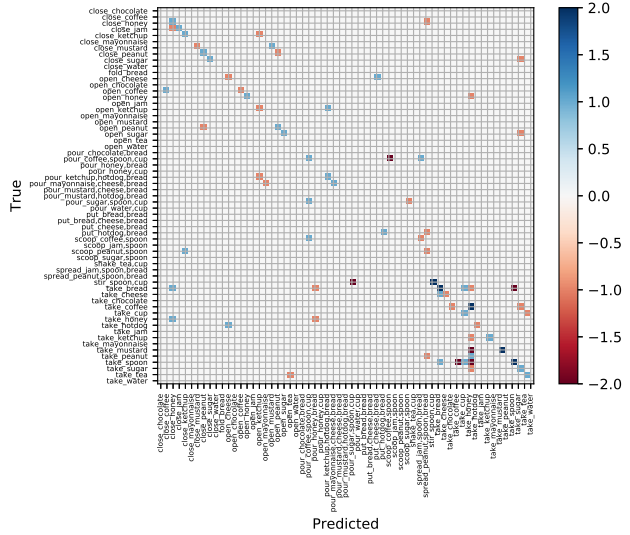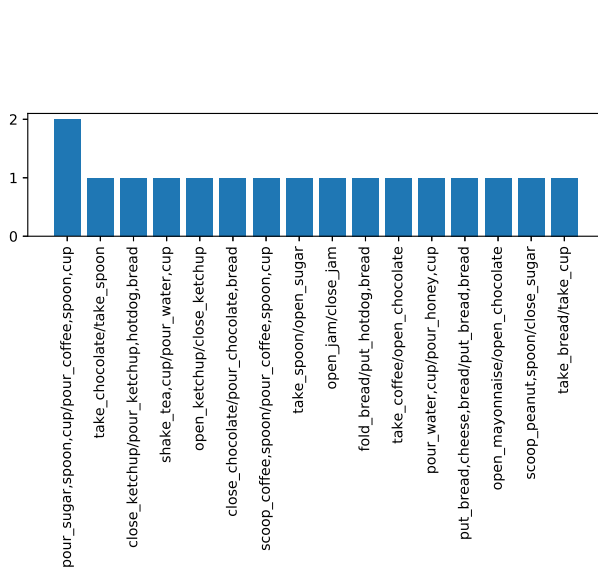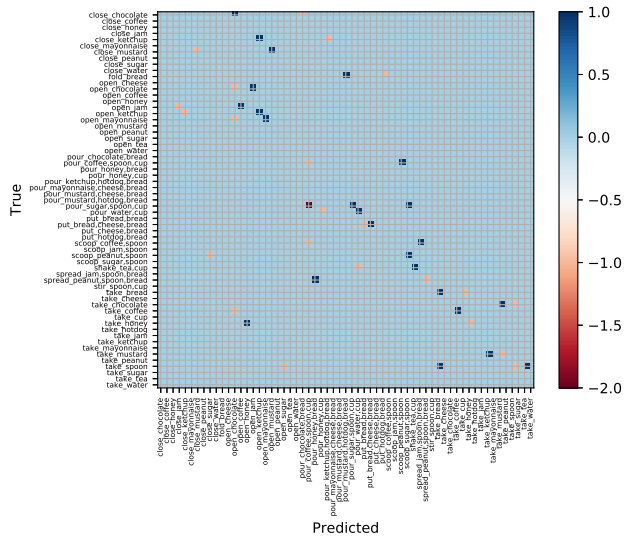

Figure 9: (a) Most improvement categories by LSTA over eleGAtt-LSTM on GTEA 61 fixed split. X axis labels are in the format true label (LSTA)/predicted label (eleGAtt-LSTM). Y axis shows the number of corrected samples for each class. (b) shows the difference of confusion matrices.

(a)   (b)

Figure 10: (a) Most improvement categories by two stream cross-modal fusion over two stream on GTEA 61 fixed split. X axis labels are in the format true label (two stream cross-modal fusion)/predicted label (two stream late fusion). Y axis shows the number of corrected samples for each class. (b) shows the difference of confusion matrices.

	Method	Top-1 Accuracy (%)			Top-5 Accuracy (%)			Precision (%)			Recall (%)		
		Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
S1	2SCNN (RGB)	40.44	30.46	13.67	83.04	57.05	33.25	34.74	28.23	6.66	15.90	23.23	5.47
	2SCNN (two stream)	42.16	29.14	13.23	80.58	53.70	30.36	29.39	30.73	5.92	14.83	21.10	4.93
	TSN (RGB)	45.68	36.80	19.86	85.56	64.19	41.89	**61.64**	34.32	11.02	23.81	31.62	9.76
	TSN (two stream)	48.23	36.71	20.54	84.09	62.32	39.79	47.26	35.42	11.57	22.33	30.53	9.78
	LSTA (RGB)	58.25	**38.93**	30.16	**86.57**	**62.96**	**50.16**	44.09	**36.30**	**16.54**	37.32	**36.52**	**19.00**
	LSTA (two stream)	59.55	38.35	**30.33**	85.77	61.49	49.97	42.72	36.19	14.46	**38.12**	36.19	17.76
S2	2SCNN (RGB)	34.89	21.82	10.11	74.56	45.34	25.33	19.48	14.67	5.32	11.22	17.24	6.34
	2SCNN (two stream)	36.16	18.03	7.31	71.97	38.41	19.49	18.11	15.31	3.19	10.52	12.55	3.00
	TSN (RGB)	34.89	21.82	10.11	74.56	45.34	25.33	19.48	14.67	5.32	11.22	17.24	6.34
	TSN (two stream)	39.4	22.7	10.89	74.29	**45.72**	25.26	22.54	15.33	6.21	13.06	17.52	6.49
	LSTA (RGB)	45.51	**23.46**	15.88	75.25	43.16	30.01	26.19	17.58	8.44	20.80	**19.67**	11.29
	LSTA (two stream)	**47.32**	22.16	**16.63**	**77.02**	43.15	**30.93**	**31.57**	**17.91**	**8.97**	**26.17**	17.80	**11.92**

Table 5: Comparison of recognition accuracies with state-of-the-art in EPIC-KITCHENS dataset.
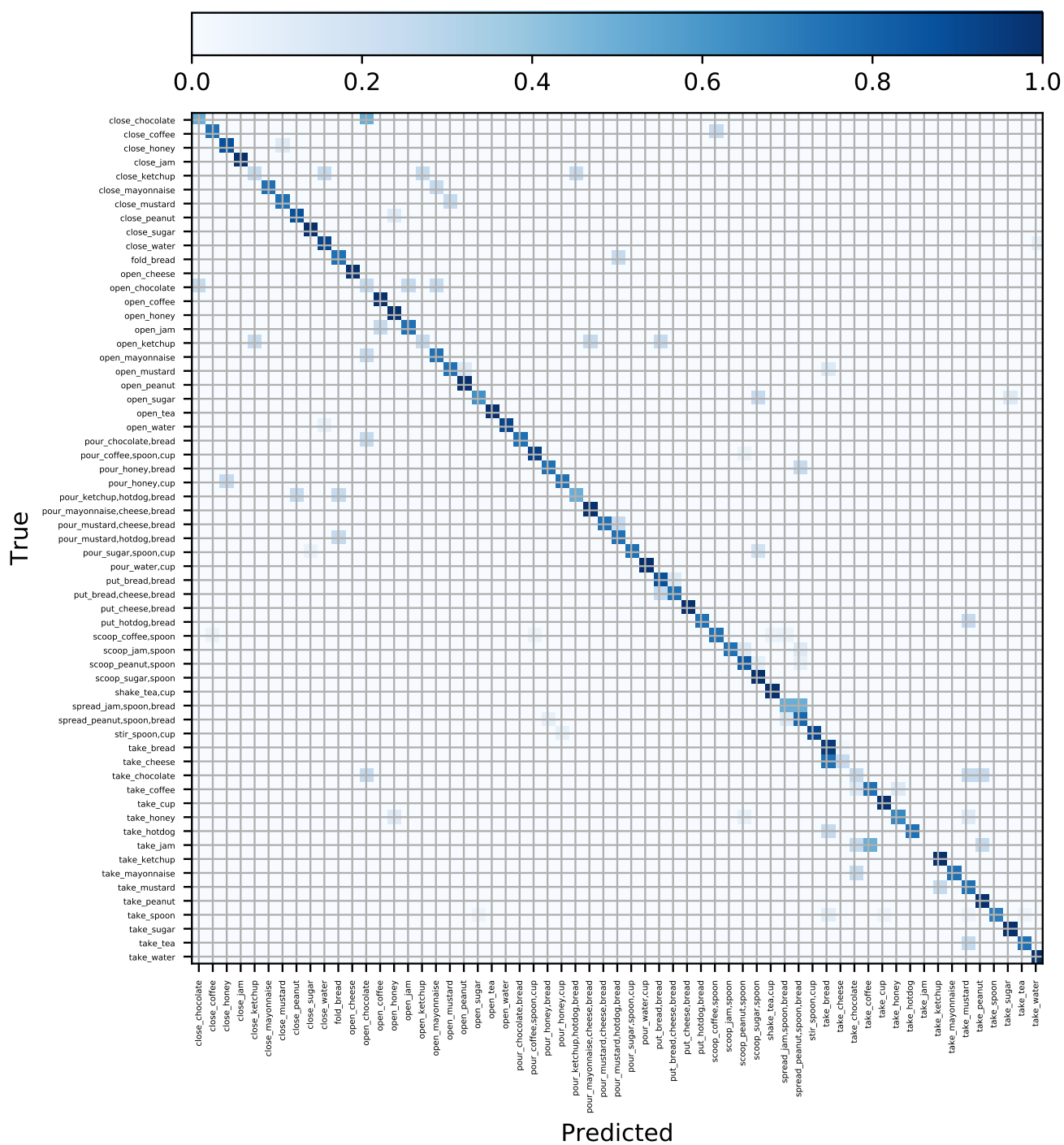
Figure 11: Confusion matrix of GTEA 61 averaged across the four train/test splits.
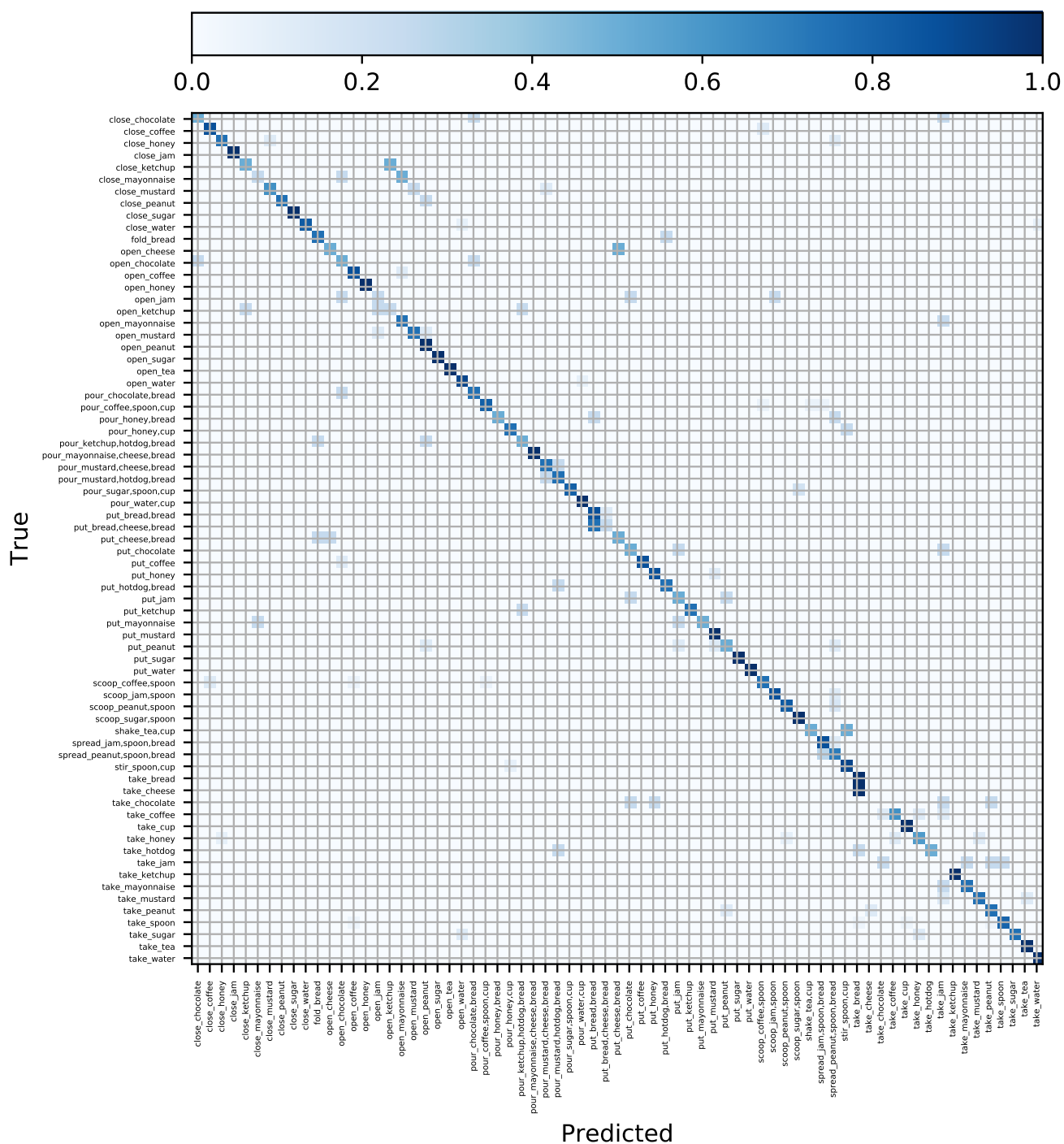
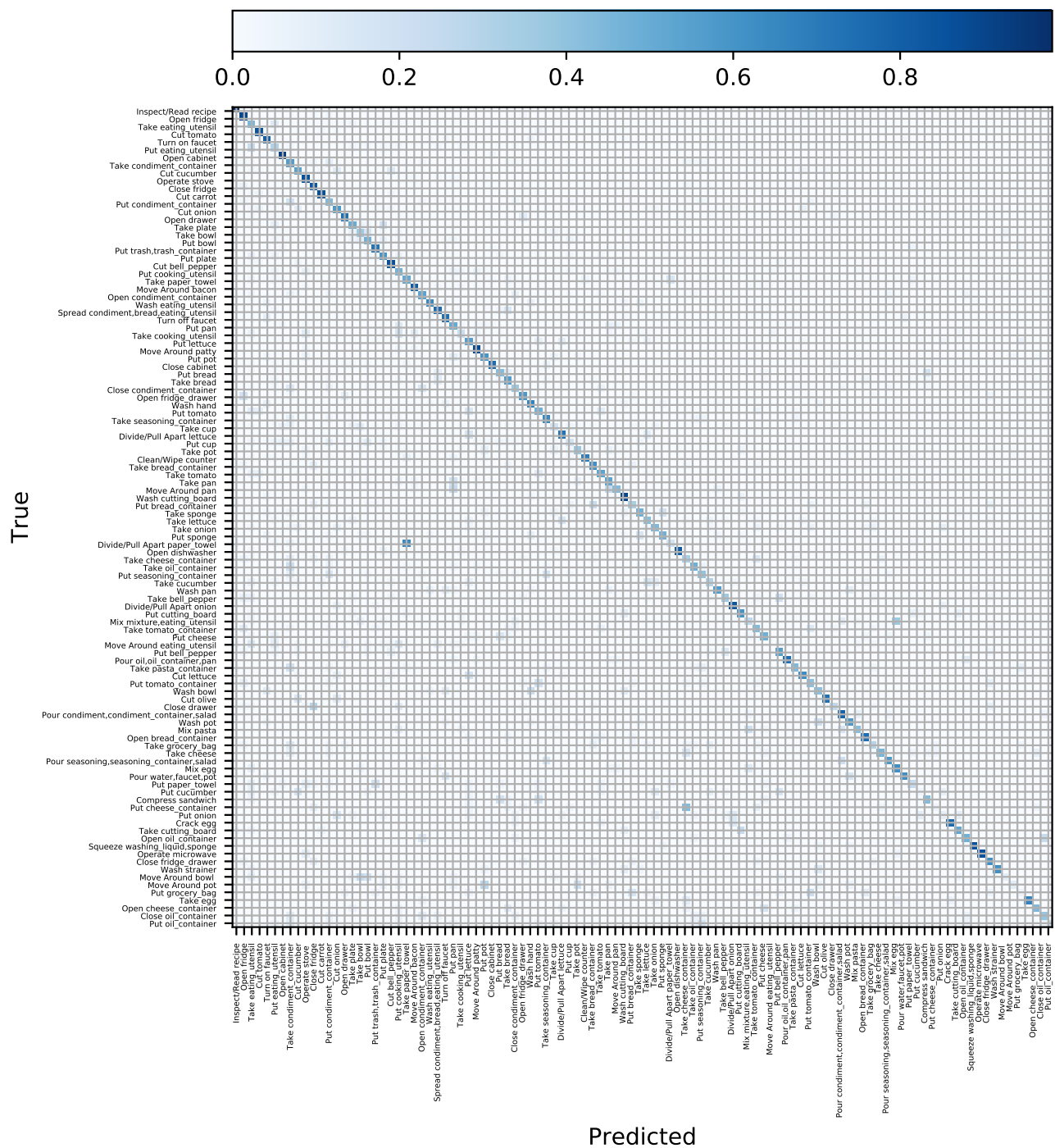Figure 12: Confusion matrix of GTEA 71 averaged across the four train/test splits.

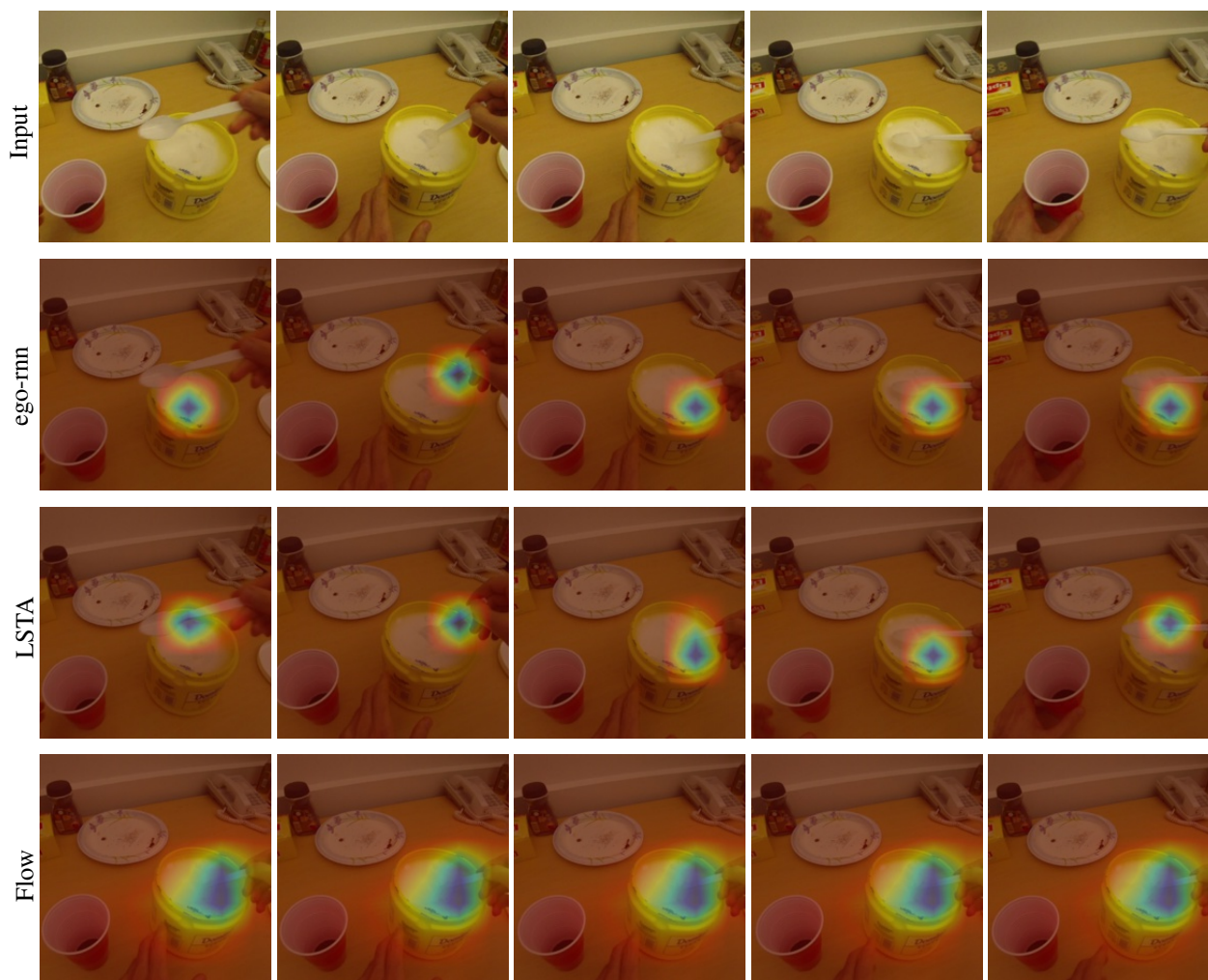Figure 13: Confusion matrix of EGTEA Gaze+ averaged across the three train/test splits.

Figure 14: Attention maps generated by ego-rnn (second row) and LSTA (third) for scoop_sugar,spoon video sequence. We show the 5 frames that are uniformly sampled from the 25 frames used as input to the corresponding networks. Fourth row shows the attention map generated by the motion stream. For flow, we visualize the attention map on the five frames corresponding to the optical flow stack given as input.
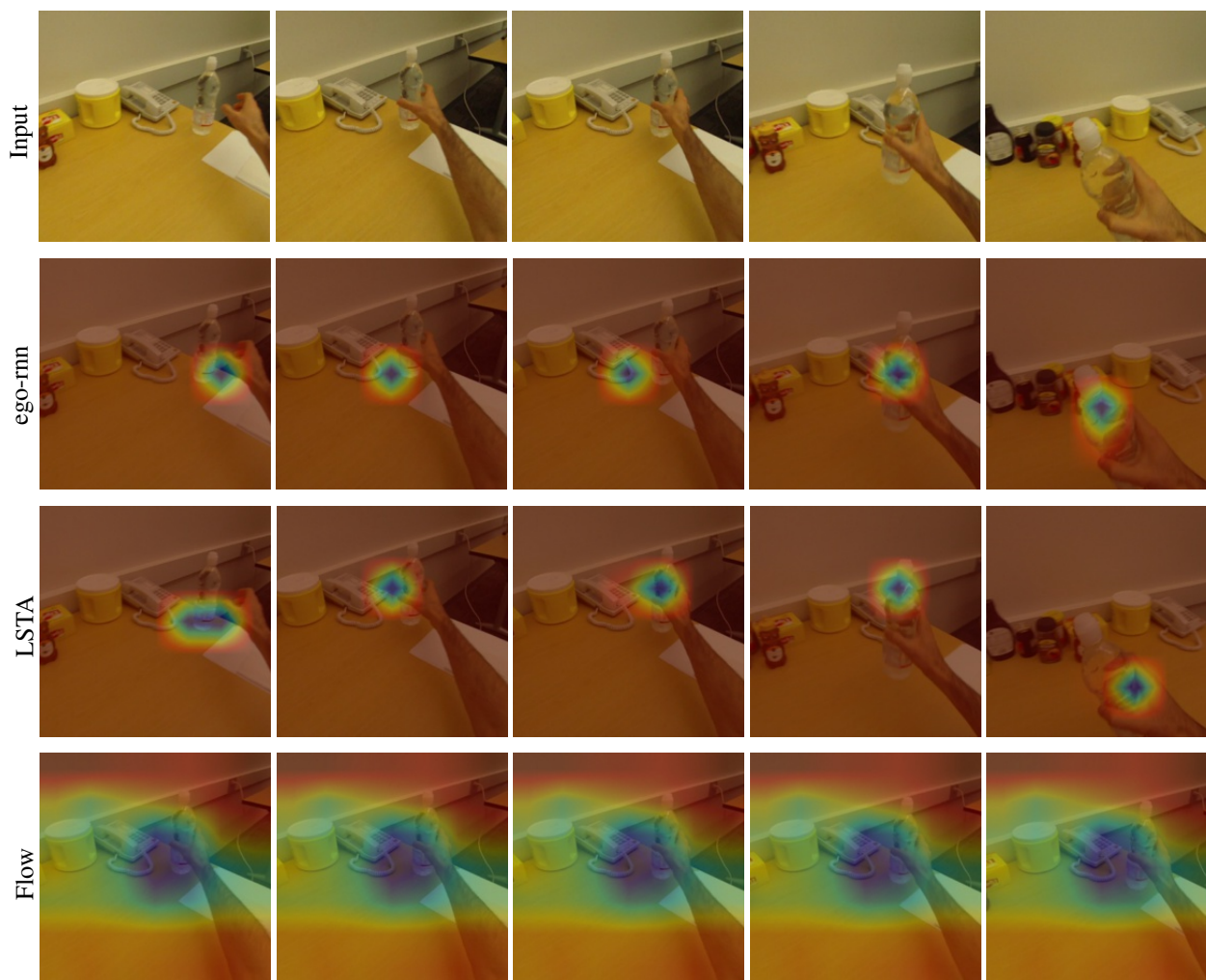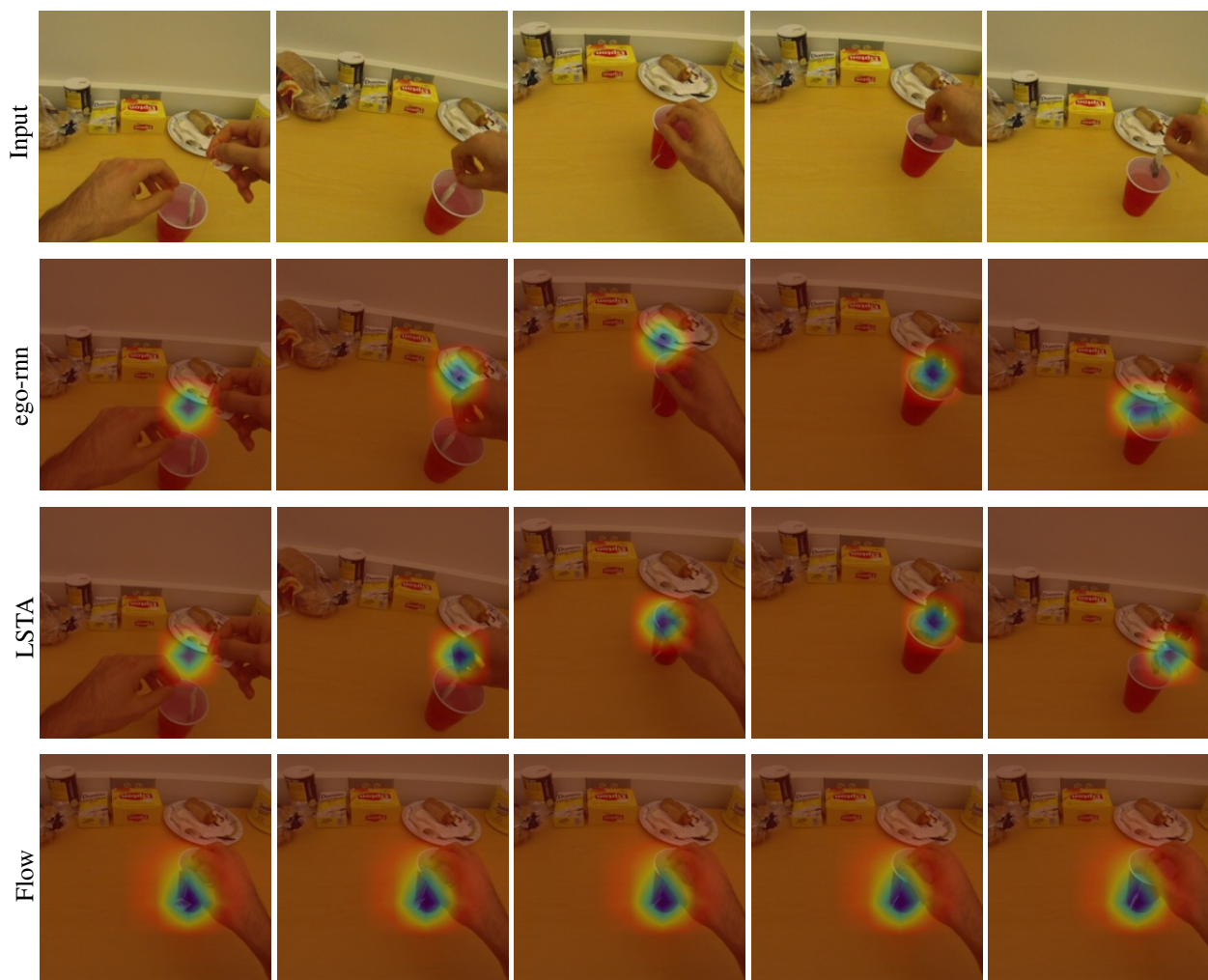
Figure 15: Attention maps generated by ego-rnn (second row) and LSTA (third) for take_water video sequence. We show the 5 frames that are uniformly sampled from the 25 frames used as input to the corresponding networks. Fourth row shows the attention map generated by the motion stream. For flow, we visualize the attention map on the five frames corresponding to the optical flow stack given as input.

Figure 16: Attention maps generated by ego-rnn (second row) and LSTA (third) for shake_tea,cup video sequence. We show the 5 frames that are uniformly sampled from the 25 frames used as input to the corresponding networks. Fourth row shows the attention map generated by the motion stream. For flow, we visualize the attention map on the five frames corresponding to the optical flow stack given as input.
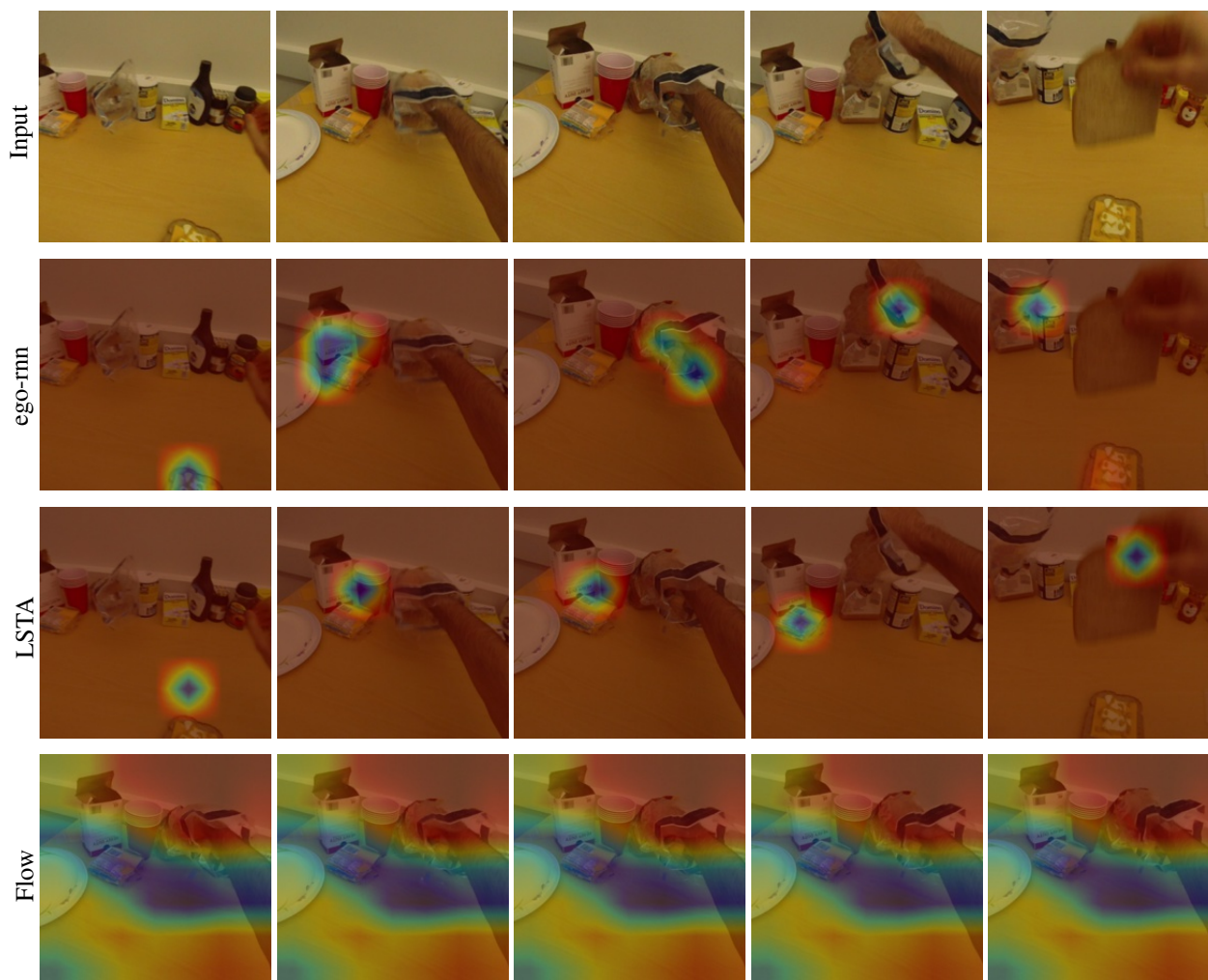
Figure 17: Attention maps generated by ego-rnn (second row) and LSTA (third) for take_bread video sequence. We show the 5 frames that are uniformly sampled from the 25 frames used as input to the corresponding networks. Fourth row shows the attention map generated by the motion stream. For flow, we visualize the attention map on the five frames corresponding to the optical flow stack given as input.
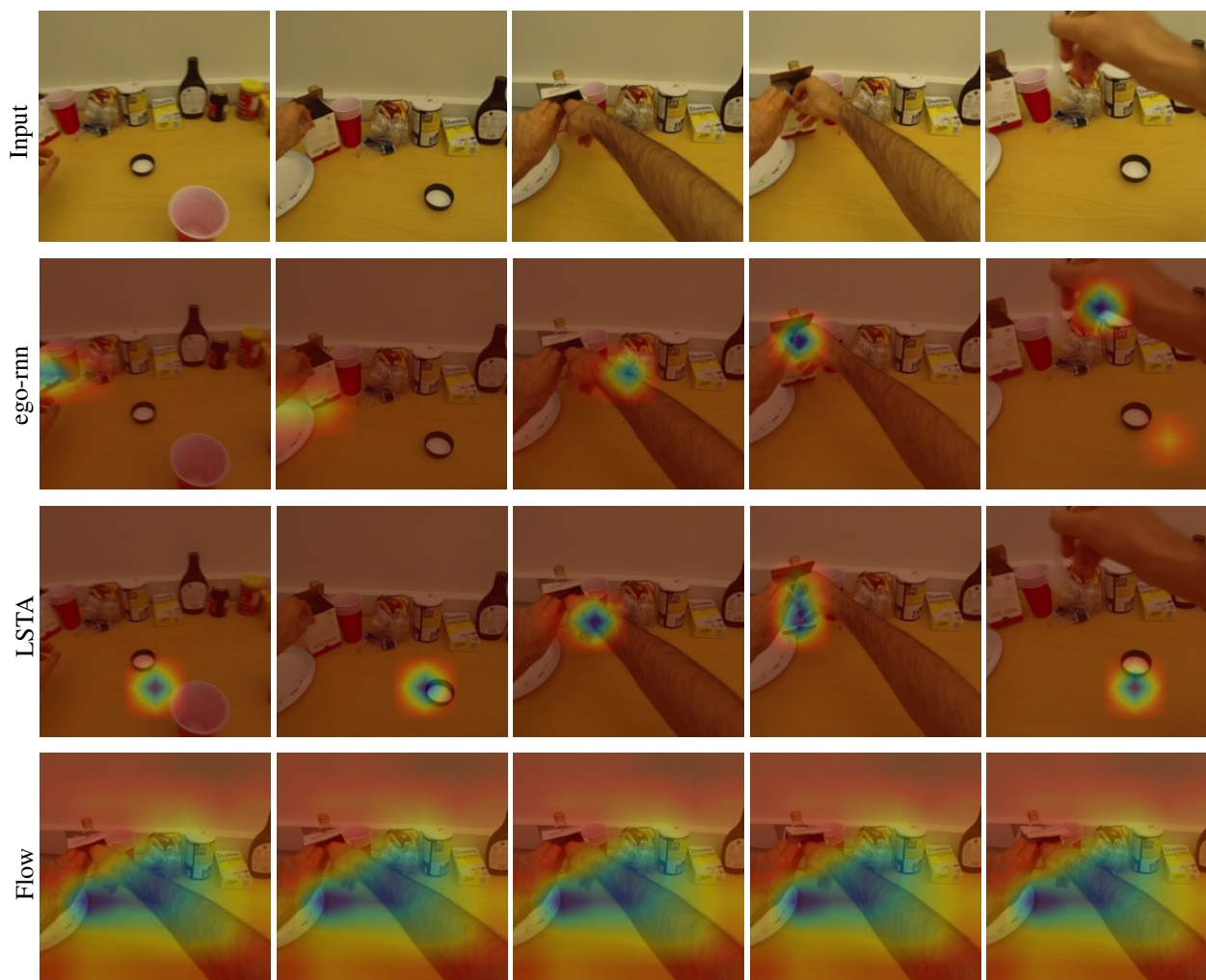
Figure 18: Attention maps generated by ego-rnn (second row) and LSTA (third) for take_spoon video sequence. We show the 5 frames that are uniformly sampled from the 25 frames used as input to the corresponding networks. Fourth row shows the attention map generated by the motion stream. For flow, we visualize the attention map on the five frames corresponding to the optical flow stack given as input.