What Would You Expect? Anticipating Egocentric Actions With Rolling-Unrolling LSTMs and Modality Attention (Supplementary Material)

Antonino Furnari Giovanni Maria Farinella
University of Catania - Department of Mathematics and Computer Science
http://iplab.dmi.unict.it/fpv/ - {furnari,gfarinella}@dmi.unict.it

This document is intended for the convenience of the reader and contains supplementary material not included in the submitted paper due to space limits. Specifically, Section 1 reports details on implementation and training procedure of the proposed method. Section 2 reports details on implementation and training of the compared methods. Section 3 reports the full set of anticipation and recognition results on EPIC-Kitchens, including precision and recall. We also report related to the compared methods on EGTEA Gaze+ for action recognition. Section 4 reports screenshots of the EPIC-Kitchens egocentric action anticipation leaderboard at end of the competition. Section 5 reports additional qualitative examples. The reader is also referred to the videos included in this supplementary material for qualitative assessment of the proposed method.

1. Implementation Details and Training Procedure of the Proposed Method

This section reports the implementation and training details of both the proposed and compared methods. A diagram of our architecture is reported in Figure 1 for the convenience of the reader. The reader is referred to the paper for a description of the architecture.

1.1. Architectural Details of RU-LSTM and MATT

We use a Batch Normalized Inception CNN [9] (BNInception) in the spatial and flow branches and consider the 1024-dimensional vectors produced by the last global average pooling layer of the network as output representations. Optical flows are extracted using the TVL1 algorithm [20]. Specifically, we use the pre-computed optical flows provided by the authors in the case of EPIC-Kitchens (see `http://EPIC-Kitchens.github.io/`) and compute optical flows on EGTEA Gaze+ using the code provided in `https://github.com/feichtenhofer/gpu_flow` with default parameters. At test time, the CNNs are fed with input images and optical flows resized to 456×256 pixels. Note that, due to global average pooling, the output of the BNInception CNN will be a 1024 feature vector regardless size of the input image. We found this setting leading to better performance as compared to extracting a 224×224 crop from the center of the image. For the object branch, we use a Faster R-CNN object detector [6] with a ResNet-101 backbone [8], as implemented in [7]. Both the Rolling LSTM (R-LSTM) and the Unrolling LSTM (U-LSTM) contain a single layer with 1024 hidden units. Dropout with $p = 0.8$ is applied to the input of each LSTM and to the input of the final fully connected layer used to obtain class scores. The Modality ATTention network (MATT) is a feed-forward network with three fully connected layers containing respectively $h/4$, $h/8$ and 3 hidden units, where $h = 6144$ is the dimension of the input to the attention network (i.e., the concatenation of the hidden and cell states of 1024 units each related to the three R-LSTMs). Dropout with $p = 0.8$ is applied to the input of the second and third layers of the attention network to avoid over-fitting. ReLU activation function are used within the attention network.

1.2. Training Procedure of RU-LSTM and MATT

While the proposed architecture could be in principle trained in an end-to-end fashion, we found it extremely challenging to avoid over-fitting during end-to-end training. This is mainly due to the indirect relationship between input video and future actions. Indeed, differently from *action recognition*, where the objects and actions to be recognized are present or take place in the input video, in the case of *action anticipation*, the system should be able to anticipate objects and actions which do not always appear in the input video, which makes it hard to learn good representations end-to-end. To avoid over-fitting, the proposed architecture is trained as follows. First, we independently train the spatial and motion CNNs for the task of egocentric action recognition within the framework of TSN [19]. Specifically, we set the number of segments to 3 and train the TSN models with Stochastic Gradient Descent (SGD) using standard

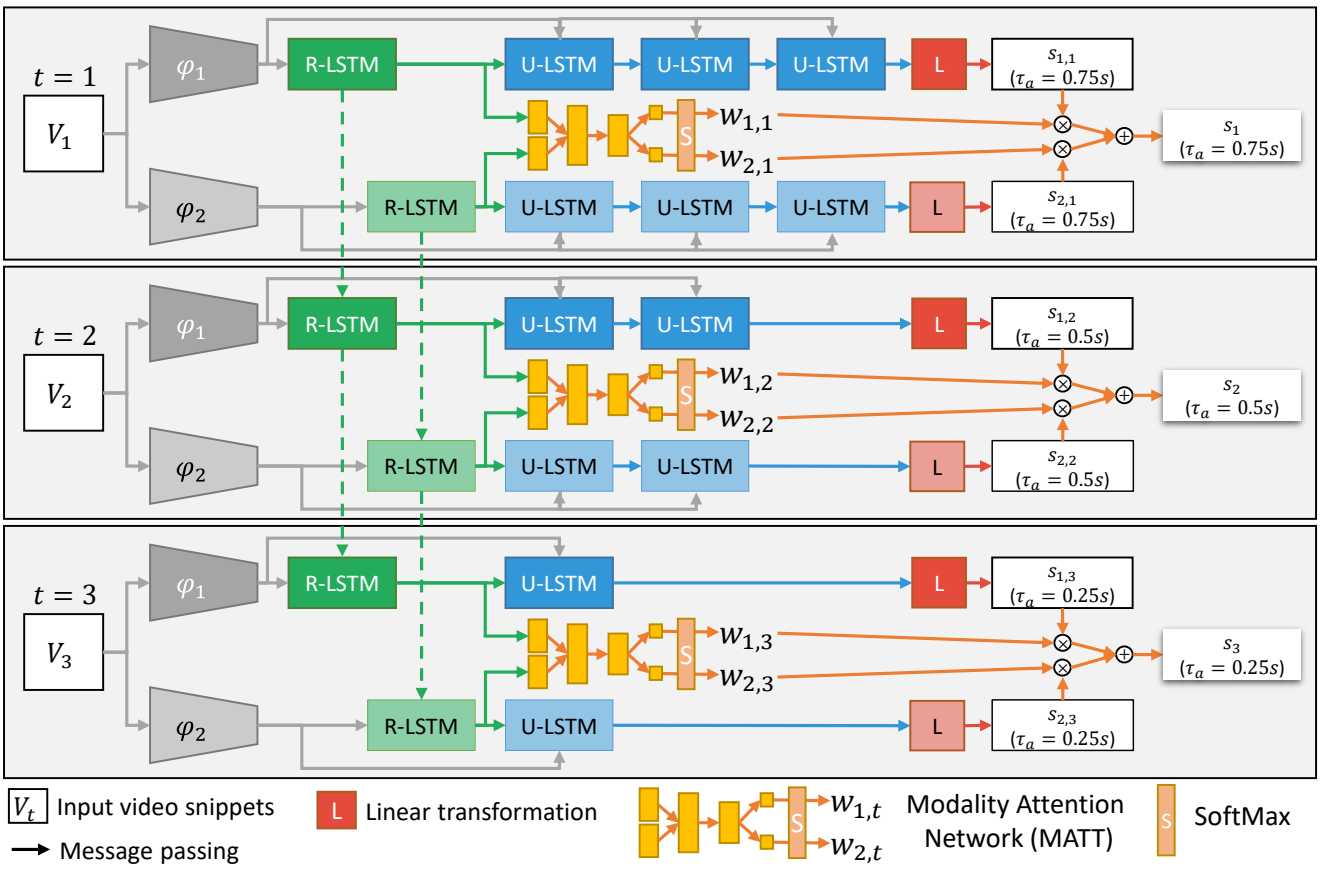

Figure 1. Example of the proposed architecture with $M = 2$ modalities. In our experiments, we use three modalities: RGB, Flow and OBJ. Modules belonging to different branches are illustrated using different color shades.

cross entropy for 160 epochs with an initial learning rate equal to 0.001, which is decreased by a factor of 10 after 80 epochs. We use a mini-batch size of 64 samples and train the models on a single Titan X. For all other parameters, we use the values recommended in [19]. We train the object detector to recognize the 352 object classes of the EPIC-Kitchens dataset. We use the same object detector trained on EPIC-Kitchens when performing experiments on EGTEA Gaze+, as the latter dataset does not contain object bounding box annotations. This training procedure allows to learn the parameters θ^1, θ^2 and θ^3 of the representation functions related to the three modalities (i.e., RGB, Flow, OBJ). After this procedure, these parameters are fixed and they are no more optimized. For efficiency, we pre-compute representations over the whole dataset.

Each branch of the RU-LSTM is training with SGD using the cross entropy loss with a fixed learning rate equal to 0.01 and momentum equal to 0.9. Each branch is first pre-trained with Sequence Completion Pre-training (SCP). Specifically, appearance and motion branches are trained for 100 epochs, whereas the object branch is trained for 200 epochs. Branches are then fine-tuned for the action anticipation task. Once each branch has been trained, the complete architecture with three branches is assembled to form a three-branch network and the model is further fine-tuned for 100 epochs using cross entropy and the same learning parameters. In Figure 1 an example of a two-branches architecture is shown.

In the case of early action recognition, each branch is trained for 200 epochs (both SCP and main task) with a fixed learning rate equal to 0.01 and momentum equal to 0.9.

Note that, in order to improve performances, we apply early stopping at each training stage. This is done by choosing the iterations of the intermediate and final models which obtain the best Top-5 action anticipation accuracy for the anticipation time $\tau_a = 1s$ on the validation set. In the case of *early action recognition*, we choose the epoch obtaining the best average Top-1 action accuracy across observation rates. The same early stopping strategy is applied to all the methods for fair comparison. The proposed RU-LSTM architecture has been implemented using the PyTorch li-

brary [14]. The code will be provided upon publication, together with all details and data useful to replicate the results.

1.2.1 Note on End-To-End Training

We chose to fix the feature extractors in our work as we experienced over-fitting when training the model end-to-end. Specifically, we tried the following: (1) Training the RGB branch end-to-end from scratch (except the CNN, which is pre-trained on Imagenet), (2) Pre-training the CNN on the action recognition task with TSN, then training the RGB branch end-to-end, (3) Training the RGB branch using fixed representations as described in the paper, then fine-tuning the CNN + RU-LSTM model end-to-end. In our experiments, (1) and (2) led to poor performance already at the sequence-completion stage, while (3) did not improve performance. Our insight is that the indirect relationship between the observed scene and the action yet to take place can make learning representations end-to-end much more difficult than in the case of recognition. For instance, when anticipating the action "take cup", the object "cup" may or may not be present in the observed video segment, which makes unclear what visual features the CNN should extract.

1.3. Inference At a Fixed Anticipation Time

Our model makes multiple anticipations at time-steps $7-14$. Also, since the predictions are updated as the video is processed and more evidence is acquired, such predictions may indeed be inconsistent (with anticipations performed closer to the beginning of the action being more likely to be correct). However, it should be noted that each prediction is deemed to be specific to a given anticipation time. For instance, at time-step 11, the model tries to anticipate actions happening in $1s$. Therefore, the proposed approach can be used to anticipate actions at a fixed anticipation time by processing the buffered video up to the related time-step, discarding all other predictions. E.g., if the anticipation time is set to $\tau_a = 1s$, the model should process the last 11 time-steps.

1.4. Choice of Parameters α, S_{enc} and S_{ant}.

In this work, we set $\alpha = 0.25s$ and $S_{ant} = 8$ to generalize the settings of the EPIC-Kitchens anticipation challenge. Indeed, in these settings, we can anticipate actions up to $2s$ in advance ($8 \times 0.25s$), while still being able to produce anticipations at anticipation time $\tau_a = 1s$ ($4 \times 0.25s$) as required for the challenge. We investigated the effect of S_{enc} when we fix $\alpha = 0.25s$ and $S_{dec} = 8$. We noted that the choice of S_{enc} affects performance lightly and hence chose $S_{enc} = 6$ to maximize action anticipation performance for anticipation time $\tau_a = 1s$.

2. Implementation Details of the Compared Methods

Since no official public implementation is available for the compared methods, we performed experiments using our own implementations. In this section, we report the implementation details of each of the compared method.

2.1. Deep Multimodal Regressor (DMR)

We implement the Deep Multimodal Regressor proposed in [18] setting the number of multi-modal branches with interleaved units to $k = 3$. For fair comparisons, we substituted the AlexNet backbone originally considered in [18] with a BNInception CNN pre-trained on ImageNet. The CNN is trained to anticipate future representations extracted using BNInception pre-trained on ImageNet using the procedure proposed by the authors. Specifically, we perform mode update every epoch. Since training an SVM with large number of classes is challenging (in our settings, we have $2,513$ different action classes), we substituted the SVM with a Multi Layer Perceptron (MLP) with 1024 hidden units and dropout with $p = 0.8$ applied to the input of the first and second layer. To comply with the pipeline proposed in [18], we pre-train the model on our training split of EPIC-Kitchens in an unsupervised fashion and train the MLP separately on representations pre-extracted from the training set using the optimal modes found at training time. As a result, during the training of the MLP, the weights of the CNN are not optimized. The DMR architecture is trained with Stochastic Gradient Descent using a fixed learning rate equal to 0.1 and a momentum equal to 0.9. The network is trained for several epochs until the validation loss saturates. Note that training the CNN on the EPIC-Kitchens dataset takes several days on a single Titan X GPU using our implementation. After training, we apply early stopping by selecting the iteration with the lowest validation loss. The MLP is then trained with Stochastic Gradient Descent with fixed learning rate equal to 0.01 and momentum equal to 0.9. Early stopping is applied also in this case considering the iteration of the MLP achieving the highest Top-5 action accuracy on the validation set.

2.2. Anticipation TSN (ATSN)

We implement this model considering the TSN architecture used to pre-train the CNNs employed in the RGB and Flow branches of our architecture. We modify the network to output verb and noun scores and train it summing the cross entropy losses applied independently to verbs and nouns as specified in [2]. At test time, we obtain action probabilities by assuming independence of verbs and nouns as follows: $p(a = (v, n)|x) = p(v|x) \cdot p(n|x)$, where $a = (v, n)$ is an action involving verb v and noun n, x is the input sample, whereas $p(v|x)$ and $p(n|x)$ are the proba-

bilities computed directly by the network.

2.3. ATSN + VNMCE Loss (MCE)

This method is implemented training the TSN architecture used for ATSN with the Verb-Noun Marginal Cross Entropy Loss proposed in [3]. We used the official code provided by the authors (`https://github.com/fpv-iplab/action-anticipation-losses/`).

2.4. Encoder-Decoder LSTM (ED)

We implement this model following the details specified in [4]. For fair comparison with respect to the proposed method, the model takes RGB and Flow features obtained using the representation functions as input for the RGB and Flow modalities used in our RU architecture. Differently from [4], we do not include a reinforcement learning term in the loss as our aim is not to distinguish the action from the background as early as possible as proposed in [4]. The hidden state of the LSTMs is set to 2048 units. The model encodes representations for 20 steps, while decoding is carried out for 10 steps at a step-size of $0.25s$. The architecture is trained on top of pre-extracted representations for 100 epochs with the Adam optimizer and a learning rate of 0.001.

2.5. Feedback-Network LSTM (FN)

The method proposed in [5] has been implemented considering the most performing architecture investigated by the authors, which comprises the "optional" LSTM layer and performs fusion by concatenation. The network uses our proposed video processing strategy. For fair comparison, we implement the network as a two-stream architecture with two branches processing independently RGB and Flow features. Final predictions are obtained with late fusion (equal weights for the two modalities). We use the representation functions of our architecture to obtain RGB and Flow features. The model has hidden layers of 1024 units, which in our experiments leaded to improved results with respect to the 128 features proposed by the authors [5]. The model is trained using the same parameters used in the proposed architecture.

2.6. RL & EL

These two methods are implemented considering a single LSTM with the same parameters of our Rolling LSTM. Similarly to FN, the models are trained as two-stream models with late fusion used to obtain final predictions. The input RGB and Flow features are computed using the representation functions considered in our architecture. The models are trained with the same parameters used in the proposed architecture. RL is trained using the ranking loss on the detection score proposed in [13], whereas EL is trained using the exponential anticipation loss proposed in [10].

3. Additional Results

This section reports the full set of anticipation and recognition results on EPIC-Kitchens, including precision and recall, as well as the full table of comparisons of the proposed method on EGTEA Gaze+ for action recognition.

Table 1 compares the proposed method with respect to the competitors according to the full set of measures proposed along with the egocentric *action anticipation* challenge [2], including precision and recall (which could not be included in the paper due to space limits). The proposed approach outperforms all competitors also according to precision and recall on **S1** and **S2**, except for average verb precision, where it is outperform by the two-stream CNN. Note that, coherently with Top-1 and Top-5 accuracy, the proposed method achieves large gains for noun precision and recall. Also note the small drop in performance between Top-1 noun accuracy and average noun recall on **S1** (from 22.78% to 19.81%), which highlights balanced noun predictions.

Table 2 compares the proposed method with respect to the competitors according to the full set of measures proposed with the *egocentric action recognition* challenge [2]. Similarly to Table 1, this includes precision and recall, which could not be included in the paper due to space limits. Similarly to what observed in the case of top-1 and top-5 accuracy, the proposed method outperforms the competitors according to most of the considered measures, despite not being explicitly designed to tackle the recognition task (i.e., our architecture was designed for the *egocentric action anticipation* task.)

Table 3 compares the proposed RU method against the state-of-the-art when tackling the task of egocentric action recognition on EGTEA Gaze+. It is worth noting that the proposed method outperforms many recent approaches by significant margins. It is also comparable with other state-of-the-art approaches such as the ones proposed in [16, 17]. Again, note that our architecture generalizes despite not being explicitly designed for the recognition task.

4. EPIC-Kitchens Egocentric Action Anticipation Challenge Leaderboards

The proposed RULSTM approach has been used to participate in the EPIC-Kitchens egocentric action anticipation competition. Specifically, we considered an ensemble model including features extracted using a BNInception and a ResNet-50 CNN trained for action recognition. Figure 2 reports a screenshot of the EPIC-Kitchens egocentric action anticipation challenge at the end of the competition. The screenshot has been acquired from `https:`

		Top-1 Accuracy%			Top-5 Accuracy%			Avg Class Precision%			Avg Class Recall%		
		VERB	NOUN	ACTION	VERB	NOUN	ACTION	VERB	NOUN	ACTION	VERB	NOUN	ACTION
S1	2SCNN (Fusion) [2]	29.76	15.15	04.32	76.03	38.56	15.21	13.76	17.19	02.48	07.32	10.72	01.81
	TSN (Fusion) [2]	31.81	16.22	06.00	76.56	42.15	28.21	23.91	19.13	03.13	09.33	11.93	02.39
	VNMCE [3]	27.92	16.09	10.76	73.59	39.32	25.28	23.43	17.53	06.05	14.79	11.65	05.11
	RU-LSTM	**33.04**	**22.78**	**14.39**	**79.55**	**50.95**	**33.73**	**25.50**	**24.12**	**07.37**	**15.73**	**19.81**	**07.66**
	Imp. wrt best	+1.23	+6.56	+3.63	+2.99	+8.80	+5.52	+1.59	+4.99	+1.32	+0.94	+7.88	+2.55
S2	2SCNN (Fusion) [2]	25.23	09.97	02.29	68.66	27.38	09.35	**16.37**	06.98	00.85	05.80	06.37	01.14
	TSN (Fusion) [2]	25.30	10.41	02.39	68.32	29.50	06.63	07.63	08.79	00.80	06.06	06.74	01.07
	VNMCE [3]	21.27	09.90	05.57	63.33	25.50	15.71	10.02	06.88	01.99	07.68	06.61	02.39
	RU-LSTM	**27.01**	**15.19**	**08.16**	**69.55**	**34.38**	**21.10**	13.69	**09.87**	**03.64**	**09.21**	**11.97**	**04.83**
	Imp. wrt best	+1.71	+4.78	+2.59	+0.89	+4.88	+5.39	-2.68	+1.08	+1.65	+1.53	+5.23	+2.44

Table 1. Egocentric action anticipation results on the EPIC-Kitchens test set.

		Top-1 Accuracy%			Top-5 Accuracy%			Avg Class Precision%			Avg Class Recall%		
		VERB	NOUN	ACTION	VERB	NOUN	ACTION	VERB	NOUN	ACTION	VERB	NOUN	ACTION
S1	2SCNN (Fusion) [2]	42.16	29.14	13.23	80.58	53.70	30.36	29.39	30.73	05.53	14.83	21.10	04.46
	TSN (Fusion) [2]	48.23	36.71	20.54	84.09	62.32	39.79	47.26	35.42	10.46	22.33	30.53	08.83
	LSTA [16]	**59.55**	38.35	30.33	**85.77**	61.49	49.97	42.72	36.19	14.46	**38.12**	36.19	17.76
	VNMCE [3]	54.22	38.85	29.00	85.22	61.80	49.62	**53.87**	38.18	18.22	35.88	32.27	16.56
	RU-LSTM	56.93	**43.05**	**33.06**	85.68	**67.12**	**55.32**	50.42	**39.84**	**18.91**	37.82	**38.11**	**19.12**
	Imp.	-2.62	+4.20	+2.73	-0.09	+4.80	+5.35	-3.45	+1.66	+0.69	-0.30	+1.92	+1.36
S2	2SCNN (Fusion) [2]	36.16	18.03	07.31	71.97	38.41	19.49	18.11	15.31	02.86	10.52	12.55	02.69
	TSN (Fusion) [2]	39.40	22.70	10.89	74.29	45.72	25.26	22.54	15.33	05.60	13.06	17.52	05.81
	LSTA [16]	**47.32**	22.16	16.63	**77.02**	43.15	30.93	**31.57**	17.91	08.97	**26.17**	17.80	11.92
	VNMCE [3]	40.90	23.46	16.39	72.11	43.05	31.34	26.62	16.83	07.10	15.56	17.70	10.17
	RU-LSTM	43.67	**26.77**	**19.49**	73.30	**48.28**	**37.15**	23.40	**20.82**	**09.72**	18.41	**21.59**	**13.33**
	Imp.	-3.65	+3.31	+2.86	-3.72	+2.56	+5.81	-8.17	+2.91	+0.75	-7.76	+3.79	+1.41

Table 2. Egocentric action recognition results on the EPIC-Kitchens test set.

Method	Acc.%	Imp.
Lit et al. [12]	46.50	+13.7
Li et al. [11]	53.30	+6.90
Two stream [15]	41.84	+18.7
I3D [1]	51.68	+8.52
TSN [19]	55.93	+4.27
eleGAtt [21]	57.01	+3.19
ego-rnn [17]	60.76	-0.56
LSTA [16]	**61.86**	-1.66
RU	60.20	/

Table 3. Recognition results on EGTEA Gaze+.

`//epic-kitchens.github.io/` on the 1^{st} of August, 2019. Note that our submission (team name "DMI-UNICT") is ranked first on both **S1** and **S2**.

5. Additional Qualitative Examples

Figure 3 reports qualitative results of three additional success action anticipation examples. For improved clarity, we report frames with and without optical flows for each example. In the top example, MATT assigns a small weight to the object branch as the contextual appearance features (i.e., RGB) are already enough to reliably anticipate the next actions. In the middle example object detection is fundamental to correctly anticipate "put down spoon", as soon as the object is detected. The bottom example shows a complex scene with many objects. The ability to correctly recognize objects is fundamental to anticipate certain actions (i.e., "wash spoon"). The algorithm can anticipate "wash" well in advance. As soon as the spoon is detected ($\tau_a = 2s$), "wash spoon" is correctly anticipated. Note that, even if the spoon is not correctly detected at time $\tau_a = 0.5s$, "wash spoon" is still correctly anticipated.

Figure 4 reports three failure examples. In the top example, the model fails to predict "adjust chair", probably due to the inability of the object detector to identify the chair. Note that, when the object "pan" on the table is detected, "take curry" is wrongly anticipated. In the middle example, the algorithm successfully detects the fridge and tries to anticipate "close fridge" and some actions involving the "take" action, with wrong objects. This is probably due to the inability of the detector to detect "mozzarella", which is not yet appearing in the scene. In the bottom example, the method tries to anticipate actions involving "jar", as soon as "jar" is detected. This misleads the algorithm as the correct action is "pour coffee".

The reader is referred to the videos in the supplementary material for additional success and failure qualitative examples. The supplementary material also reports qualita-

Seen Kitchens (S1)

#	User	Entries	Date of Last Entry	Team Name	Top-1 Accuracy (%)			Top-5 Accuracy (%)			Precision (%)			Recall (%)		
					Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
1	antoninofurnari	9	05/05/19	DMI-UNICT	31.13 (5)	22.93 (2)	15.25 (1)	78.03 (2)	51.05 (1)	35.13 (1)	22.58 (3)	24.26 (1)	8.41 (1)	17.71 (2)	20.05 (1)	8.05 (1)
2	Nour	13	05/30/19	RML-Ryerson University	34.40 (2)	23.36 (1)	13.20 (2)	79.07 (1)	47.57 (2)	31.80 (2)	26.36 (1)	21.81 (3)	5.28 (2)	19.47 (1)	20.01 (2)	5.20 (2)
3	masterchef	14	10/03/18	Inria / Facebook	30.74 (6)	16.47 (4)	9.74 (3)	76.21 (5)	42.72 (3)	25.44 (3)	12.42 (7)	16.67 (5)	3.67 (3)	8.80 (6)	12.66 (4)	3.85 (3)
4	EPIC_TSN_RGB	1	09/05/18		31.81 (3)	16.22 (5)	6.00 (4)	76.56 (3)	42.15 (4)	18.21 (4)	23.91 (2)	19.13 (4)	3.47 (4)	9.33 (4)	11.93 (5)	2.64 (5)
5	zhe2325138	2	05/31/19	NTU CML Mira	31.15 (4)	16.84 (3)	5.72 (5)	76.43 (4)	40.60 (5)	15.53 (7)	15.24 (6)	15.49 (6)	3.38 (5)	9.16 (5)	13.91 (3)	3.13 (4)
6	yassersouri	3	05/31/19	UniBonn	34.94 (1)	13.06 (7)	5.23 (6)	79.07 (1)	37.00 (7)	16.27 (5)	18.88 (4)	9.22 (8)	1.50 (8)	14.47 (3)	9.77 (6)	2.36 (6)
7	EPIC_TSN_Fusion	1	09/05/18		30.66 (7)	14.86 (6)	4.62 (7)	75.32 (6)	40.11 (6)	16.01 (6)	8.84 (8)	21.85 (2)	2.49 (6)	6.76 (8)	9.15 (7)	1.72 (7)
8	EPIC_TSN_Flow	1	09/05/18		29.64 (8)	10.30 (8)	2.93 (8)	73.70 (7)	30.09 (8)	10.92 (8)	18.34 (5)	10.70 (7)	1.56 (7)	6.99 (7)	5.48 (8)	1.11 (8)

Unseen Kitchens (S2)

#	User	Entries	Date of Last Entry	Team Name	Top-1 Accuracy (%)			Top-5 Accuracy (%)			Precision (%)			Recall (%)		
					Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
1	antoninofurnari	9	05/05/19	DMI-UNICT	26.63 (5)	15.47 (2)	9.12 (1)	68.11 (7)	35.27 (1)	21.88 (1)	16.58 (2)	9.93 (2)	3.16 (2)	11.08 (2)	11.70 (2)	4.55 (1)
2	Nour	13	05/30/19	RML-Ryerson University	27.89 (3)	15.53 (1)	8.50 (2)	70.47 (2)	34.28 (2)	20.38 (2)	17.77 (1)	12.32 (1)	3.28 (1)	9.35 (3)	12.11 (1)	3.84 (2)
3	masterchef	14	10/03/18	Inria / Facebook	28.37 (2)	12.43 (3)	7.24 (3)	69.96 (3)	32.20 (3)	19.29 (3)	11.62 (6)	8.36 (4)	2.20 (3)	7.80 (5)	9.94 (3)	3.36 (3)
4	yassersouri	3	05/31/19	UniBonn	32.37 (1)	9.66 (6)	3.52 (4)	73.51 (1)	30.83 (4)	12.67 (4)	15.60 (3)	6.51 (5)	1.44 (4)	12.77 (1)	7.22 (5)	2.39 (4)
5	zhe2325138	2	05/31/19	NTU CML Mira	27.59 (4)	9.05 (7)	2.77 (5)	69.27 (4)	25.78 (7)	7.82 (8)	13.66 (4)	5.94 (6)	1.25 (5)	7.82 (4)	7.63 (4)	1.45 (5)
6	EPIC_TSN_RGB	1	09/05/18		25.30 (8)	10.41 (4)	2.39 (6)	68.32 (5)	29.50 (5)	9.63 (5)	7.63 (8)	8.79 (3)	0.89 (8)	6.06 (7)	6.74 (6)	1.20 (6)
7	EPIC_TSN_Flow	1	09/05/18		25.61 (6)	8.40 (8)	1.78 (7)	67.57 (8)	24.62 (8)	8.19 (7)	10.80 (7)	4.99 (8)	1.14 (6)	6.34 (6)	4.72 (8)	0.94 (7)
8	EPIC_TSN_Fusion	1	09/05/18		25.37 (7)	9.76 (5)	1.74 (8)	68.25 (6)	27.24 (6)	9.05 (6)	13.03 (5)	5.13 (7)	1.00 (7)	5.65 (8)	5.58 (7)	0.88 (8)

Figure 2. Screenshots of the EPIC-Kitchens Egocentric Action Anticipation Challenge Leaderboards at the end of the competition, acquired from `https://epic-kitchens.github.io/` on the 1^{st} of August, 2019. The team name of the proposed method is "DMI-UNICT".

tive examples of the proposed method when applied to the problem of early action recognition.

Acknowledgment

This research is supported by Piano della Ricerca 2016-2018, linea di Intervento 2 of DMI, University of Catania.

Figure 3. Success action anticipation example qualitative results (best seen on screen).

References

[1] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition*, pages 4724–4733, 2017.

[2] D. Damen, H. Doughty, G. M. 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 *European Conference on Computer Vision*, pages 720–736, 2018.

[3] A. Furnari, S. Battiato, and G. M. Farinella. Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation. In *European Conference on Computer Vision Workshops*, 2018.

[4] Y. Gao, Z. Yang, and R. Nevatia. RED: Reinforced encoder-decoder networks for action anticipation. *British Machine Vision Conference*, 2017.

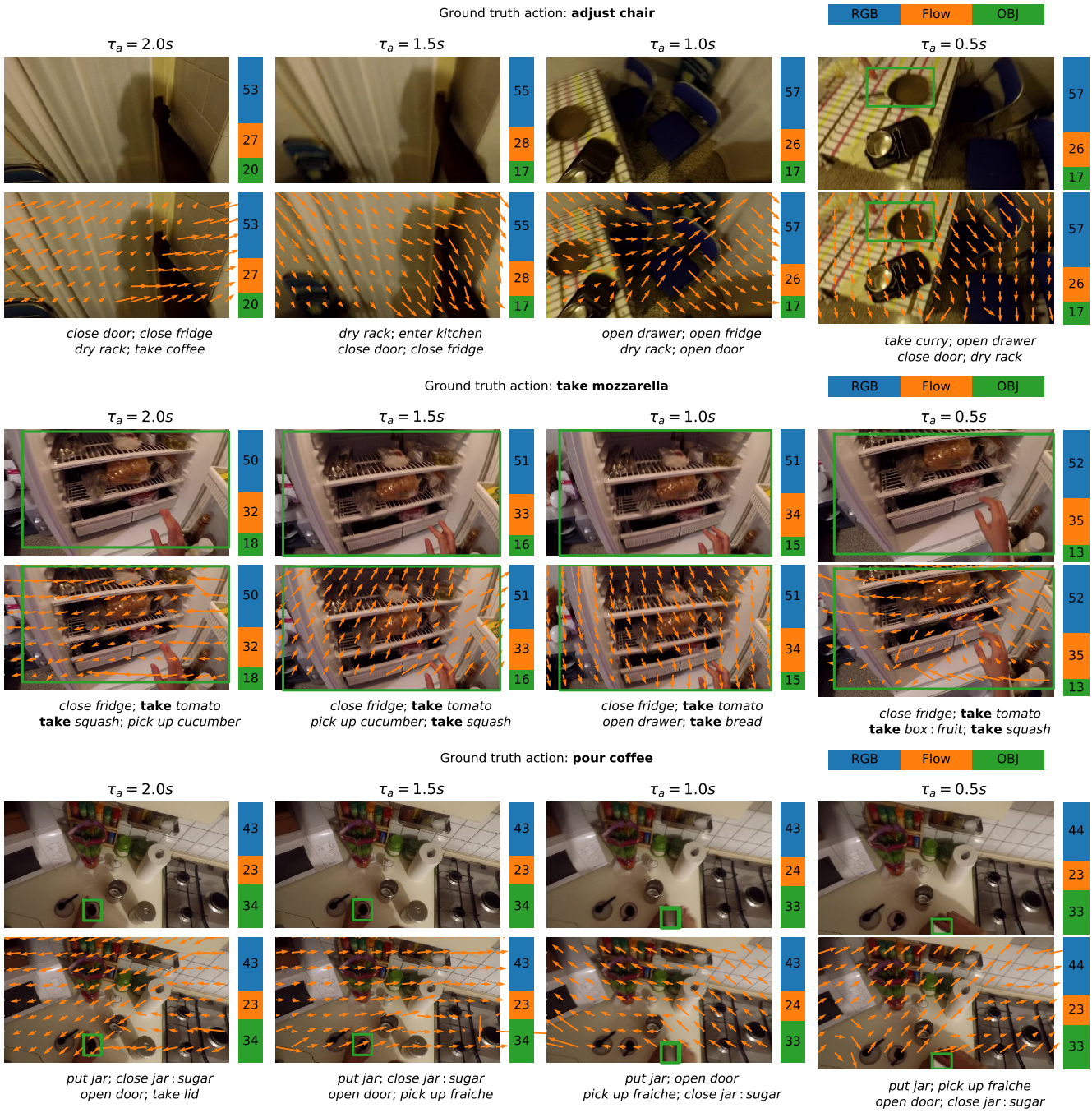

Figure 4. Failure action anticipation example qualitative results (best seen on screen).

[5] R. D. Geest and T. Tuytelaars. Modeling temporal structure with lstm for online action detection. In *Winter Conference on Applications in Computer Vision*, 2018.

[6] R. Girshick. Fast R-CNN. In *International Conference on Computer Vision*, pages 1440–1448, 2015.

[7] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[10] A. Jain, A. Singh, H. S. Koppula, S. Soh, and A. Saxena.

Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In *International Conference on Robotics and Automation*, pages 3118–3125. IEEE, 2016.

[11] Y. Li, M. Liu, and J. M. Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *European Conference on Computer Vision*, 2018.

[12] Y. Li, Z. Ye, and J. M. Rehg. Delving into egocentric actions. In *Computer Vision and Pattern Recognition*, pages 287–295, 2015.

[13] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *Computer Vision and Pattern Recognition*, 2016.

[14] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[15] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[16] S. Sudhakaran, S. Escalera, and O. Lanz. Lsta: Long short-term attention for egocentric action recognition. *arXiv preprint arXiv:1811.10698*, 2018.

[17] S. Sudhakaran and O. Lanz. Attention is all we need: Nailing down object-centric attention for egocentric activity recognition. *British Machine Vision Conference*, 2018.

[18] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating visual representations from unlabeled video. In *Computer Vision and Pattern Recognition*, pages 98–106, 2016.

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

[20] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint Pattern Recognition Symposium*, pages 214–223, 2007.

[21] P. Zhang, J. Xue, C. Lan, W. Zeng, Z. Gao, and N. Zheng. Adding attentiveness to the neurons in recurrent neural networks. In *European Conference on Computer Vision*, pages 135–151, 2018.