

EPIC-KITCHENS - 2019 Challenges Report

Dima Damen, Will Price, Evangelos Kazakos
University of Bristol, UK

Antonino Furnari, Giovanni Maria Farinella
University of Catania, Italy

Abstract

This report summarises the EPIC-KITCHENS 2019 challenges, and their findings. It serves as an introduction to all technical reports that were submitted to the EPIC@CVPR2019 workshop, and an official announcement of the winners.

1. EPIC-KITCHENS

The largest dataset in egocentric vision has a number of unique features that distinguished its collection. Primarily, the dataset was collected in a *non-scripted* manner. Participants were asked to record all kitchen interactions in their *native environments*, i.e. their kitchens, for three consecutive days. This enabled capturing daily interactions that are often not included in scripted recordings, such as baking or emptying the bin. More importantly, the frequencies of interactions form a valid prior to daily interactions and demonstrate a long-tail unbalanced distribution of labels.

In addition to its natural interactions, EPIC-KITCHENS proposed approaches to enable scalability of collecting annotations in video. Videos were narrated by the participants themselves, providing weak supervision of temporal boundaries and an open vocabulary description of captured actions in people’s native languages. While the vocabulary was refined using clustering into semantic classes, the temporal bounds were altered through Amazon Mechanical Turk (AMT) providing start/end time annotations for around 40K action segments. The annotations were further enriched by annotating bounding boxes of active objects within each interaction. Around half a million bounding boxes were annotated.

As specified in [2], the test set has been divided into two distinct subsets, “seen test set (S1)” and “unseen test set (S2)”. The first includes sequences from the same kitchens/environments in the training set. S2, on the other hand, only includes sequences from novel environments, not observed during training.

Following the release, three challenges were officially launched via CodaLab on the 20th of September 2019. Users were requested to submit their predictions to the evaluation server, with a maximum daily limit of 1 submission per team. In Sec. 2, we detail the general statistics of dataset usage in its first year. The results for the Action Recognition and Action Anticipation challenges are provided in Sec. 3 and 4 respectively. The winners of the 2019 edition of these challenges are noted in Sec. 5.

2. Reception and User Statistics

Since its introduction, EPIC-KITCHENS received significant attention with a total of 13K page views since April 2018. The dataset has been downloaded 1.5K times, with international coverage (Fig 1), and the CodaLab competitions have 170 accepted participants. The Action Recognition challenge received the largest number of participants (103 participants) and submissions (230 submissions). The Action Anticipation challenge has 44 participants, and received 46 submissions. Of these, 10 teams have declared their affiliation and submitted technical reports for the Action Recognition challenge compared to 5 in the Action Anticipation challenge. This report includes details of these teams’ submissions. A snapshot of the complete leaderboard, when the 2019 challenge concluded, is available at <http://epic-kitchens.github.io/2019#results>.

The Object Detection challenge has not received submissions that outperform the baseline. This is, up to our knowledge, due to two key factors. The first is the duration required to train the models. In [2], we clarify that the model required 2 weeks to train on an 8-GPU node. The second is the distinction from other datasets that are typically used for object detection (e.g. [6, 7]). In EPIC-KITCHENS , the same instance of an object is labelled several times during its usage. This introduces dependencies between the annotations which are not typical for object detection from individual images. We will be revisiting this challenge in 2020.

3. Action Recognition Challenge

The Action Recognition challenge has been set similar to previous challenges [1, 8]. In both train and test sets, the start and end times of an action are given. Different

Rank	Team	Submissions		Top-1 Accuracy			Top-5 Accuracy			Avg Class Precision			Avg Class Recall			
		Entries	Date	VERB	NOUN	ACTION	▲	VERB	NOUN	ACTION	VERB	NOUN	ACTION	VERB	NOUN	ACTION
S1	1 UTS-Baidu	16	05/30/19	69.80	52.27	41.37		90.95	76.71	63.59	63.55	46.86	25.13	46.94	49.17	26.39
	2 Bristol-Oxford	2	05/30/19	66.10	47.89	36.66		91.28	72.80	58.62	60.74	44.90	24.02	46.82	43.89	22.92
	3 FAIR	9	10/30/19	64.14	47.65	35.75		87.64	70.66	54.65	43.64	40.53	18.95	38.31	45.29	21.13
	4 FBK-HUPBA	42	05/29/19	63.34	44.75	35.54		89.01	69.88	57.18	63.21	42.26	19.76	37.77	41.28	21.19
	5 UNICT	5	05/05/19	58.99	45.00	35.14		86.70	69.08	57.62	52.23	40.06	19.40	42.12	39.32	20.28
	7 NTU	12	05/30/19	61.65	43.63	30.55		87.09	68.65	40.11	48.63	39.62	16.92	33.41	40.57	16.68
	11 CA	4	05/31/19	63.29	44.03	29.18		86.24	65.99	46.94	56.35	39.95	12.86	34.83	38.76	10.53
	13 RML	47	05/26/19	52.75	39.42	25.04		86.26	64.45	45.22	42.80	34.51	10.66	31.44	35.48	9.51
	14 [2] (baseline)	-	09/06/18	48.23	36.71	20.54		84.09	62.32	39.79	47.26	35.42	11.57	22.33	30.53	9.78
	15 Inria-Facebook	1	10/02/18	43.51	32.94	20.19		84.38	61.66	43.57	28.42	27.99	7.62	24.18	26.83	8.85
	17 UGA	34	05/23/19	47.41	28.31	19.76		81.33	53.77	36.98	31.20	21.21	9.83	20.43	22.48	10.23
S2	1 UTS-Baidu	16	05/30/19	58.96	33.90	25.20		82.69	62.27	45.48	30.33	28.83	15.73	28.54	30.52	18.90
	2 FAIR	9	10/30/19	55.24	33.87	23.93		80.23	58.25	40.15	25.71	28.19	15.72	25.69	29.51	17.06
	3 Bristol-Oxford	2	05/30/19	54.46	30.39	21.99		81.22	55.68	40.59	32.56	21.67	9.83	27.60	25.58	13.52
	4 UNICT	5	05/05/19	47.35	28.64	21.37		73.75	51.01	39.47	26.88	22.09	10.53	22.12	23.31	13.98
	5 FBK-HUPBA	42	05/29/19	49.37	27.11	20.25		77.50	51.96	37.56	31.09	21.06	9.18	18.73	21.88	14.23
	7 CA	4	05/31/19	53.36	28.37	18.47		77.47	50.15	33.63	31.23	22.12	7.24	21.29	22.56	9.40
	11 NTU	12	05/30/19	52.78	24.62	16.35		79.72	49.61	22.81	23.31	17.91	9.00	22.02	19.29	9.62
	12 RML	47	05/26/19	45.51	24.41	16.08		77.47	50.15	34.14	21.70	17.12	5.16	18.33	18.82	9.27
	14 Inria-Facebook	1	10/02/18	39.30	22.43	14.10		76.41	47.35	32.43	20.42	15.96	4.83	16.95	17.72	8.46
	15 [2] (baseline)	-	09/06/18	39.40	22.70	10.89		74.29	45.72	25.26	22.54	15.33	6.21	13.06	17.52	6.49
	17 UGA	34	05/23/19	34.35	17.48	9.08		69.24	37.56	19.46	15.09	10.71	3.68	11.00	12.55	4.77

Table 1: Results on EPIC-KITCHENS Action Recognition challenge - 1 June 2019

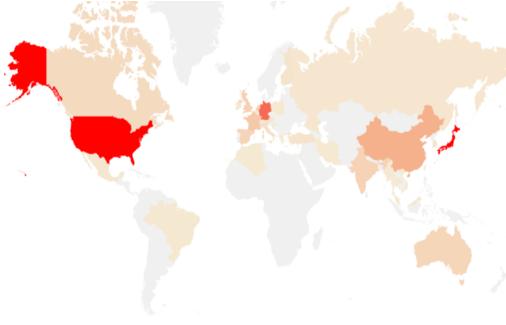


Figure 1: Heatmap of countries based on EPIC-KITCHENS download statistics.

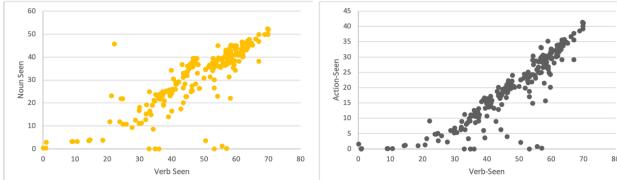


Figure 2: Scatter plot on S1 for all submissions relating top-1 verb accuracy to top-1 noun accuracy (left) and top-1 action accuracy (right).

from previous approaches, we split the action label into its parts of speech: ‘verb’ and ‘noun’, where ‘noun’ presents the prime active object in the interaction. For example, the action “put apple into bag” would consider ‘apple’ as the prime noun. Detecting ‘bag’ as the prime noun would be considered incorrect.

Table 1 shows the challenge results for 2019. This only lists entries for which a team has been declared and a technical report submitted. Methods are ranked based on top-1

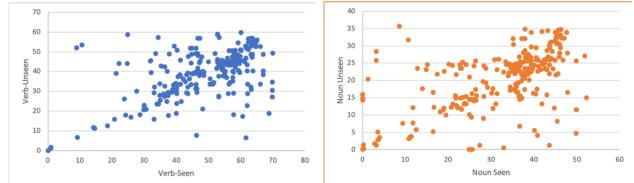


Figure 3: Scatter plot relating S1 to S2 top-1 accuracy from all submissions for verbs (left) and nouns (right).

action accuracy (noted by arrow), which was used to decide on the overall rank. The top-3 submissions are highlighted in bold. Overall, submissions achieved an overall improvement of 22%, 16% and 21% compared to the baseline published in [2] for top-1 verb, noun and action on S1.

Before presenting individual contributions, Fig. 2 shows an overall scatter plot relating the three top-1 accuracy metrics: verb, noun and action. As the figure shows, correlation is high between predicting verbs, nouns, and actions. In almost all cases, top-1 noun accuracy is ~20% worse than its verb counterpart. This could be explained by two facts. First, the number of noun classes is significantly larger than verbs classes. Second, most methods use the same architectures, originally designed for video understanding, employing the full image and global features as input. In fact, the size of objects, relative to the image, is significantly smaller than other action datasets. The top-performing method (UTS-Baidu) utilised Region-of-Interest (RoI) detection to focus on the object of interest for better noun classification.

We also relate the performance of the same method across both S1 (seen kitchens) and S2 (unseen kitchens) subsets in Fig 3. The overall trend shows ~ 10% drop between seen and unseen verb-1 accuracy. The figure shows a wider spread with some methods overfitting to the seen en-

vironments and under-performing on S2. Results are further scattered for seen vs. unseen nouns with bigger differences of $\sim 15\%$.

Following this introduction, we include a further detailed evaluation of the [Action Recognition](#) challenge, along with the release of 7 pre-trained models for a variety of state of the art architectures. We next describe the contributions of each of the teams, based on their technical reports.

3.1. Technical Reports

Technical reports for the [Action Recognition](#) challenge, in order of their overall rank on the public leaderboard, are:

UTS-Baidu (Rank 1) is the top-ranking entry by University of Technology Sydney (UTS) and Baidu Research (BAIDU), referred to as UTS-Baidu in the leaderboard. This is the only submission to utilise the active object bounding-box annotations to train for the challenge. The top-K ROI features are max pooled and used to gate the video's 3D CNN features.

Bristol-Oxford (Rank 2 - S1, Rank 3 - S2) The second-ranking entry is multi-modal, utilising RGB, Flow and Audio, along with a novel approach for temporal binding of modalities, i.e. fusing modalities with a variety of temporal offsets. An extended version of this work is available at [5].

FAIR (Rank 3 - S1, Rank 2 - S2) In contrast to other submissions, this entry was related to already published work by the time the challenge has closed. We thus refer the reader to the publication at [4].

FBK-HUPBA (Rank 4 - S1, Rank 5 - S2) uses an ensemble of variants of long-short term attention (LSTA) models and hierarchical feature aggregations of a temporal segment network. While two-stream LSTA is the best individual model, additional improvement was achieved using the ensemble.

NTU (Rank 7 - S1, Rank 11 - S2) uses hand pose and explores a variety of backbones (TSN, I3D, TSM) along with both early and late fusion of modalities. The report describes several failure cases on the dataset including challenging hand detection and the impact of ambiguous or inactive object detection on the noun accuracy in the dataset.

UGA (Rank 17 - S1, Rank 17 - S2) focuses on actions as state transformations. While the method does not rank highly on the leaderboard, the report offers a novel insight to understanding actions as linear transformations between two states. The work thus focuses on the actions for which this applies (e.g. peel, cut, open).

4. Action Anticipation Challenge

EPIC-KITCHENS offered the first action anticipation challenge, for the research community to explore action anticipation from videos, and compare the methods' performances. Fig 4 summarises the instructions available to participants of the challenge. For every annotated action, par-

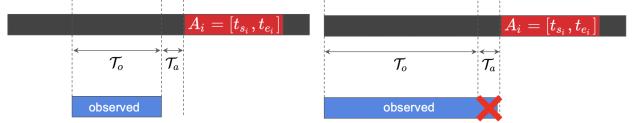


Figure 4: Expected (left) and rejected (right) action anticipation challenge instructions.

ticipants predicted the action by observing a video segment of any length T_o (observation time) that precedes the start of the action by a fixed and predefined anticipation time T_a of 1 second. All submitted technical reports were checked and participating teams were contacted for further clarification when needed.

The submissions followed the same format as that of the recognition challenge, i.e. predicting verbs, nouns and actions. Table 2 show this challenge's entries, along with their public leaderboard rankings. Results are reported using both test sets (S1 and S2) and the same evaluation metrics as in the recognition challenge. Overall, submissions have improved published baseline in [2] by 3%, 7%, and 9% for top-1 verb, noun and action. We next describe the contributions based on the team's technical reports.

4.1. Technical Reports

Technical reports for the [Action Anticipation](#) challenge, in order of their overall rank on the public leaderboard, are:

UNICT (Rank 1) The method uses a novel combination of rolled and unrolled LSTMs, as well as modality attention for RGB, flow and object-based features. The report also analyses the effect of the anticipation time on the predicted action accuracy. An extended version of this work is available at [3].

RML (Rank 2) combines the Multi-Fiber Network with the Non-Local Neural Network. The network is trained end-to-end using focal loss. High performance, particularly in noun prediction is evident in the results.

Inria-Facebook (Rank 3) is made up of two complementary modules: a predictive model which anticipates actions directly from visual inputs, and a transitional model which first predicts the current action, then anticipates the future from the current timestep onwards. The two predictions are fused with a linear combination.

NTU (Rank 5 - S1, Rank 5 - S2) uses a variety of CNN backbones to anticipate verbs and nouns separately, discovering that best performance is achieved by different backbones for the two tasks (verbs and nouns). The report also explores different fusion strategies.

Bonn (Rank 6 - S1, Rank 4 - S2) investigates whether looking longer into the past (i.e., a longer observation time) may be beneficial to action anticipation. The approach uses WaveNet to process I3D features, with a multiheaded atten-

Rank	Team	Submissions		Top-1 Accuracy			Top-5 Accuracy			Avg Class Precision			Avg Class Recall		
		Entries	Date	VERB	NOUN	ACTION ▲	VERB	NOUN	ACTION	VERB	NOUN	ACTION	VERB	NOUN	ACTION
S1	UNICT	9	05/05/19	31.13	22.93	15.25	78.03	51.05	35.13	22.58	24.26	8.41	17.71	20.05	8.05
	RML	13	05/30/19	34.40	23.36	13.20	79.07	47.57	31.80	26.36	21.81	5.28	19.47	20.01	5.20
	Inria-Facebook	14	10/03/18	30.74	16.47	9.74	76.21	42.72	25.44	12.42	16.67	3.67	8.80	12.66	3.85
	[2] (baseline)	-	09/05/18	31.81	16.22	6.00	76.56	42.15	18.21	23.91	19.13	3.47	9.33	11.93	2.64
	NTU	2	05/21/19	31.15	16.84	5.72	76.43	40.60	15.53	15.24	15.49	3.38	9.16	13.91	3.13
	Bonn	3	05/31/19	34.94	13.06	5.23	79.07	37.00	16.27	18.88	9.22	1.50	14.47	9.77	2.36
S2	UNICT	9	05/05/19	26.63	15.47	9.12	68.11	35.27	21.88	16.58	9.93	3.16	11.08	11.70	4.55
	RML	13	05/30/19	27.89	15.53	8.50	70.47	34.28	20.38	17.77	12.32	3.28	9.35	12.11	3.84
	Inria-Facebook	14	10/03/18	28.37	12.43	7.24	69.96	32.20	19.29	11.62	8.36	2.20	7.80	9.94	3.36
	Bonn	3	05/31/19	32.37	9.66	3.52	73.51	30.83	12.67	15.60	6.51	1.44	12.77	7.22	2.39
	NTU	2	05/21/19	27.59	9.05	2.77	69.27	25.78	7.82	13.66	5.94	1.25	7.82	7.63	1.45
	[2] (baseline)	-	09/05/18	25.30	10.41	2.39	68.32	29.50	9.63	7.63	8.79	0.89	6.06	6.74	1.20

Table 2: Results on EPIC-KITCHENS Action Anticipation challenge - 1 June 2019

	Team	Member	Affiliations
Action Recognition	① UTS-Baidu (wasun)	Xiaohan Wang Yu Wu Linchao Zhu Yi Yang	University of Technology Sydney, Baidu Research University of Technology Sydney, Baidu Research University of Technology Sydney University of Technology Sydney
	② FAIR (deeptigp)	Deepthi Ghadiyaram Matt Feiszli Du Tran Xuetong Yan Heng Wang	Facebook AI Facebook AI Facebook AI Facebook AI Facebook AI
	③ FBK-HUPBA (sudhakran)	Dhruv Mahajan Swathi Kiran Sudhakaran Sergio Escalera Oswald Lanz	FBK, University of Trento FBK, University of Trento CVC, Universitat de Barcelona FBK, University of Trento
	① RML (Nour)	Nour Eldin Elmahdany Yifeng He Ling Guan	Ryerson University Ryerson University Ryerson University
	② Inria-Facebook (masterchef)	Antoine Miech Ivan Laptev Josef Sivic Heng Wang Lorenzo Torresani	Inria, Ecole Normale Supérieure Inria, Ecole Normale Supérieure Inria, Ecole Normale Supérieure, CIRC Facebook AI Facebook AI
	③ NTU (zhe2325138)	Du Train Zhe-Yu Liu Ya-Liang Chung Chih-Hung Liang Yun-Hsuan Liu Ke-Jyun Wang Winston Hsu Yaser Souris	National Taiwan University National Taiwan University National Taiwan University National Taiwan University National Taiwan University National Taiwan University University of Bonn
Action Anticipation	① Bonn (yassersouri)	Tridivraj Bhattacharyya Juergen Gall Luca Minciullo	University of Bonn University of Bonn Toyota Motor Europe
	②		
	③		
	④		
	⑤		
	⑥		

Table 3: Top-3 Winners - 2019 EPIC-KITCHENS Action Recognition and Action Anticipation challenges

tion module. The proposed approach achieves the highest verb anticipation performance on both S1 and S2 but underperforms on noun and hence on action accuracy.

5. 2019 Challenge Winners

Two entries have been excluded from the challenge competitions. These are **Bristol-Oxford**, ranked second in Action Recognition and **UNICT**, ranked first in Action Anticipation. This is because both entries include challenge organisers, and the teams had longer access to the dataset, before its official release. Accordingly, Table 3 details the winners of the 2019 EPIC challenge, announced in Long Beach as part of EPIC@CVPR2019 workshop. A photo of the EPIC-KITCHENS team and winners is also in Fig 5.

References

- [1] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. CVPR*, 2017. 1



Figure 5: Organisers as well as winners of the two challenges, during EPIC@CVPR2019 Workshop in Long Beach, 17 June 2019.

- [2] D. Damen, H. Doughty, G. Maria 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. 1, 2, 3, 4
- [3] A. Furnari and G. M. Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *ICCV*, 2019. 3
- [4] D. Ghadiyaram, D. Tran, and D. Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *CVPR*, 2019. 3
- [5] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *ICCV*, 2019. 3
- [6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 1
- [7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1
- [8] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. 1

An Evaluation of Action Recognition Models on EPIC-Kitchens

Will Price
University of Bristol, UK
will.price@bristol.ac.uk

Dima Damen
University of Bristol, UK
dima.damen@bristol.ac.uk

1. Introduction

We benchmark contemporary action recognition models (TSN [12], TRN [14], and TSM [7]) on the recently introduced EPIC-Kitchens dataset [1] and release pre-trained models on GitHub¹ for others to build upon. In contrast to popular action recognition datasets like Kinetics [5], Something-Something [2], UCF101 [10], and HMDB51 [6], EPIC-Kitchens is shot from an egocentric perspective and captures daily actions in-situ. In this report, we aim to understand how well these models can tackle the challenges present in this dataset, such as its long tail class distribution, unseen environment test set, and multiple tasks (verb, noun and, action classification). We discuss the models' shortcomings and avenues for future research.

2. Models

We benchmark 3 models: Temporal Segment Networks (TSN) [12], Temporal Relational Networks (TRN) [14], and Temporal Shift Module (TSM) based networks [7], including a variety of their variants. These models are evaluated under a uniform training and testing regime to ensure the results are directly comparable. TSN is the earliest model of the three and both TRN and TSM can be viewed as evolutionary descendants of TSN, integrating temporal modelling. In the following paragraphs, we provide an explanation of how network inputs are sampled and a brief summary of the design of each network.

Sampling Inputs to the models, *snippets*, are sampled according to the TSN sampling strategy. An action clip is split into n equally sized *segments* and a snippet is sampled at a random position within each of these. For an RGB network, the input is a single frame and for a flow network it is a stack of 5 (u, v) optical flow pairs (proposed in the two-stream CNN [9]).

TSN [12] Temporal Segment Networks propagate each snippet through a 2D CNN backbone and aggregate the class scores across segments through average or max pooling. As a consequence, TSN is unable to learn temporal cor-

relations across segments. TSN is typically trained on RGB and optical flow modalities and combined by late-fusion.

TRN [14] Temporal Relation Networks propagate snippets through a 2D CNN, like in TSN, up to the pre-classification layer. These produce features rather than class confidence scores. In order to support inter-segment temporal modelling, these segment-level features are then processed by a modified relational module [8] sensitive to item ordering. Two variants of the TRN module exists: a single scale version which computes a single n -segment relation, and a multi-scale (M-TRN) variant which computes relations over ordered sets of segment features of size 2 to n . Once the relational features have been computed, they are summed and fed to a classification layer.

TSM [7] These networks functionally operate just like TSN, snippets are sampled per segment, propagated through the backbone, and then averaged. However, unlike TSN, the backbone is modified to support reasoning across segments by shifting a proportion of the filter responses across the temporal dimension. This opens the possibility for subsequent convolutional layers to learn temporal correlations.

3. Experiments

In this section, we examine how a variety of factors impact model performance such as backbone choice, input modality, and temporal support. We analyse model performance across tasks from the perspective of the more defining characteristics of the dataset: the long-tail class distribution, and the domain gap between the seen and unseen kitchen test sets.

3.1. Experimental details

Tasks EPIC-Kitchens has three tasks within the action recognition challenge: classifying the verb, noun, and action (the verb-noun pair) of a given trimmed video. We follow the approach in [1], and replace the classification layer of each model with two output FC layers, one for verbs v and one for nouns n . The models are trained with an averaged softmax cross-entropy loss over each classification layer: $\mathcal{L} = 0.5(\mathcal{L}_n + \mathcal{L}_v)$. We obtain action predictions

¹github.com/epic-kitchens/action-models

from verb and noun predictions assuming the tasks are independent. Later, we examine the impact of integrating action priors computed from the training set for action classification. Performance on these tasks are evaluated on two test sets: seen kitchens (S1) and unseen kitchens (S2). The unseen kitchens test set contains videos from novel environments, whereas the seen kitchens split contains videos from the same environments used in training.

Training We train all models with a batch size of 64 for 80 epochs using an ImageNet pretrained model for initialisation. SGD is used for optimisation with momentum of 0.9. A weight decay of 5×10^{-4} is applied and gradients are clipped at 20. We replace the backbone’s classification layer with a dropout layer, setting $p = 0.7$. We train RGB models with an initial learning rate (LR) of 0.01 for ResNet-50 based models and 0.001 for BN-Inception models. flow models are trained with an LR of 0.001. These LRs were the maximum we could achieve whilst maintaining convergence. The LR is decayed by a factor of 10 at epochs 20 and 40.

Testing Models are evaluated using 10 crops (center and corner crops as well as their horizontal flips) for each clip. The scores from these are averaged pre-softmax to produce a single clip-level score. Fusion results are obtained by averaging the softmaxed scores obtained for each modality.

3.2. Results

Backbone choice To choose a high performing backbone, we compare BN-Inception [4][11] to ResNet-50 [3] across the 3 models, training and testing with 8 segments. We did not test TSM with BN-Inception as the authors state that the shift module is harmful unless placed in a residual branch [7]. The top-1/5 accuracy across tasks is reported in Table I where the results show ResNet-50 to be superior to BN-Inception in 14/18 cases when examining top-1 action accuracy across both test sets.

Aggregate performance We now compare models with ResNet-50 backbones across tasks in Table I using top-1/5 accuracy. On the verb task, an intrinsically more temporal problem than classifying nouns, both M-TRN and TSM out perform TSN, especially when operating on RGB frames instead of flow. This can be explained by TSN’s inability to learn inter-segment correlations as only average or max pooling is used in aggregating class scores across segments. TSN flow models outperform their RGB counterparts; this can be attributed to the network being passed temporal information in the form of stacked optical flow frames. The 2D convolutions inside the network can learn temporal relations within the stack. Both (M-)TRN and TSM flow models outperform TSN flow showing that inter-segment reasoning is complimentary to intra-segment reasoning.

Unlike verb classification, noun classification does not rely on temporal modelling *as much* since objects can be

recognised from a single frame. TSM and TSN perform best on this task, with TRN models lagging 2–3% points behind. A possible explanation for the observed drop is that the relation module within TRN places heavy emphasis on extracting temporal relational information, which is of little relevance in recognising objects. Noun performance drops considerably across models when switching from RGB to flow as the former is a much better modality for recognising objects. Unexpectedly, TSM improves top-1 noun accuracy by 1% point over TSN. Additionally, we find all fusion models improve over the RGB models alone. We hypothesise that the temporal information here is helping disambiguate the action relevant object from those that are simply present in the environment.

Classifying actions, the joint task of classifying both verb and noun, is clearly very challenging, with the best top-1 accuracy on actions being 29.9% and 17.9% for the seen and unseen test set respectively. Even at top-5, the best results are 49.8% and 32.8%. Despite flow’s superior results on verb classification on the unseen test set, the inferior noun performance drags flow models below RGB models on both test sets.

An enduring approach, pioneered by the 2SCNN [9], has been to ensemble networks trained on different modalities through late fusion at test-time. Averaged across all model variants, fusing both modalities results in a 2.9%, 5.8%, and 9.7% relative improvement over the best performing single modality model for verb, noun, and action classification respectively. The best model on the seen test set is TSM fusion, followed by M-TRN fusion. On the unseen test set, the trend is reversed with M-TRN out-performing TSM.

Novel environment robustness It is interesting to examine the relative drop in model performance from the seen to unseen test set to determine the models’ ability to generalise to new environments. Table II shows that flow models are more robust to the domain gap between the seen kitchens and unseen kitchens test sets only suffering an average 22% relative drop in top-1 action accuracy compared to a 44% drop for RGB models, and 39% for fused models. The domain gap on fused models suggests that the RGB model’s predictions dominates those of the flow model. We find that flow models consistently outperform RGB models for verb classification on the unseen test set. We hypothesis this is due to the absence of appearance information in optical flow, forcing flow models to focus on motion. Motion is more environment-invariant and salient to the classification of verbs than the visual cues the RGB models will use.

Class performance analysis To further understand the differences between models, we look at confusion amongst the top-20 most frequent classes in training in Fig. I.

The verb classification results show the top-3 verbs (accounting for 53% of the actions in training) dominate predictions due to the dataset imbalance, with this effect being

BB	Model	Modality	Verb				Noun				Action			
			Top-1		Top-5		Top-1		Top-5		Top-1		Top-5	
			S1	S2	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2
BN-Inception	TSN	RGB	47.97	36.46	87.03	74.36	38.85	22.64	65.54	46.94	22.39	11.30	44.75	26.32
		Flow	51.68	47.35	84.63	76.95	26.82	21.20	50.64	42.47	16.76	13.49	33.75	27.52
		Fusion	54.70	46.06	87.24	76.65	40.11	24.27	65.81	49.27	25.43	14.78	45.69	29.81
	TRN	RGB	58.26	47.29	87.14	76.54	36.32	22.91	63.30	44.73	25.46	15.06	45.66	28.99
		Flow	55.20	50.32	84.04	77.67	23.95	19.02	47.02	40.25	16.03	12.77	32.92	27.62
		Fusion	61.04	51.83	87.46	79.11	37.90	24.75	63.69	47.35	26.54	16.59	46.37	31.14
	M-TRN	RGB	57.66	45.41	86.91	76.34	37.94	23.90	63.78	46.33	26.62	15.57	46.39	29.57
		Flow	55.92	51.38	84.44	77.74	24.88	20.69	48.37	40.83	16.78	14.00	34.09	28.75
		Fusion	61.12	51.62	87.71	78.42	39.28	26.02	64.36	48.99	27.86	17.34	47.56	32.57
ResNet-50	TSN	RGB	49.71	36.70	87.19	73.64	39.85	23.11	65.93	44.73	23.97	12.77	46.14	26.08
		Flow	53.14	47.56	84.88	76.89	27.76	20.28	51.29	42.23	18.03	13.11	35.18	27.83
		Fusion	55.50	45.75	87.85	77.40	41.28	25.13	66.53	48.11	26.89	15.40	47.35	30.01
	TRN	RGB	58.82	47.32	86.60	76.92	37.27	23.69	62.96	46.02	26.62	15.71	46.09	30.01
		Flow	55.16	50.39	83.87	77.71	23.19	18.50	47.33	40.70	15.77	12.02	33.08	27.42
		Fusion	61.60	52.27	87.20	79.55	38.41	25.74	63.37	47.87	27.58	17.79	46.44	32.20
	M-TRN	RGB	60.16	46.94	87.18	75.21	38.36	24.41	64.67	46.71	28.23	16.32	47.89	29.74
		Flow	56.79	50.36	84.91	77.67	25.00	20.28	48.70	41.45	17.24	13.42	34.80	29.02
		Fusion	62.68	52.03	87.96	78.90	39.82	25.88	64.94	49.03	29.41	17.86	48.91	32.54
	TSM	RGB	57.88	43.50	87.14	73.85	40.84	23.32	66.10	46.02	28.22	14.99	49.12	28.06
		Flow	58.08	52.68	85.88	79.11	27.49	20.83	50.27	43.70	19.14	14.27	36.90	29.60
		Fusion	62.37	51.96	88.55	79.21	41.88	25.61	66.43	49.47	29.90	17.38	49.81	32.67

Table 1: Backbone (BB) comparison using 8 segments in both training and testing evaluating top-1/5 accuracy across tasks. S1 denotes the seen test set, and S2 the unseen test set. Cells are coloured on a per column basis: low high .

especially pronounced in the unseen test set. Classes outside the top-20 are rarely correctly classified and instead are classified into one of the majority classes. The fine-grained nature of the verbs seems to pose challenges, particularly in the unseen test set, with similar classes being confused, such as ‘move’ with ‘put’/‘take’, ‘turn’ with ‘mix’, and ‘insert’ with ‘put’. TSN shows increased confusion between classes that differ primarily in their temporal aspects (e.g. ‘put’ vs ‘take’), compared to TSM and TRN. The generic class ‘move’ is hardest to classify, for all models.

For noun classification, the confusion matrices show the models don’t struggle as much to classify less frequent classes compared to verb classification. This is likely as a result of the models benefiting from pretraining on the large-scale ImageNet dataset. However, when fine-tuned, some overfitting to seen environments is observed, as the unseen test set matrices demonstrate that the models generalise less well to new objects. Like the verb results, the fine-grained classes pose a challenge with confusion between similar objects like ‘fork’ with ‘spoon’, and ‘bowl’ with ‘plate’ occurring. Another interesting contrast between the verb and noun tasks is that the top-20 verbs almost never get misclassified into any of the classes outside the top-20, whereas for nouns, there are more misclassifications of top-20 nouns into the long-tail.

For action classification, the models perform well on fre-

quent actions, but suffer more misclassifications into the long tail than nouns (as evidenced by the confusion into ‘other’ classes). Confusion within the top-20 actions highlight an issue not visible from the verb and noun matrices: semantically identical classes like ‘turn-on tap’ are confused with ‘open tap’. Whilst these are different classes in the dataset, they refer to the same action. This highlights an issue with the open vocabulary annotation process employed by the dataset: annotators may use different phrases for describing the same action.

We provide qualitative examples in Fig. 2 where TSM and M-TRN correctly classify the actions, but TSN fails. In the top example TSN confuses ‘put’ and ‘take’ as a result of averaging the scores across segments, and thus discarding temporal ordering. M-TRN and TSM show a much larger disparity between the scores of these classes indicating they have better learnt the difference. In the bottom example, TSN again struggles to correctly classify the action. The temporal bounds are quite wide and capture frames just after someone has picked up a bowl, they then open the cupboard and are about to place the bowl. M-TRN and TSM, through their ability to draw correlations across segments, are able to disambiguate the correct class from the action which came before and comes after.

Temporal support How many frames/optical flow snippets does the network need to see before performance sat-

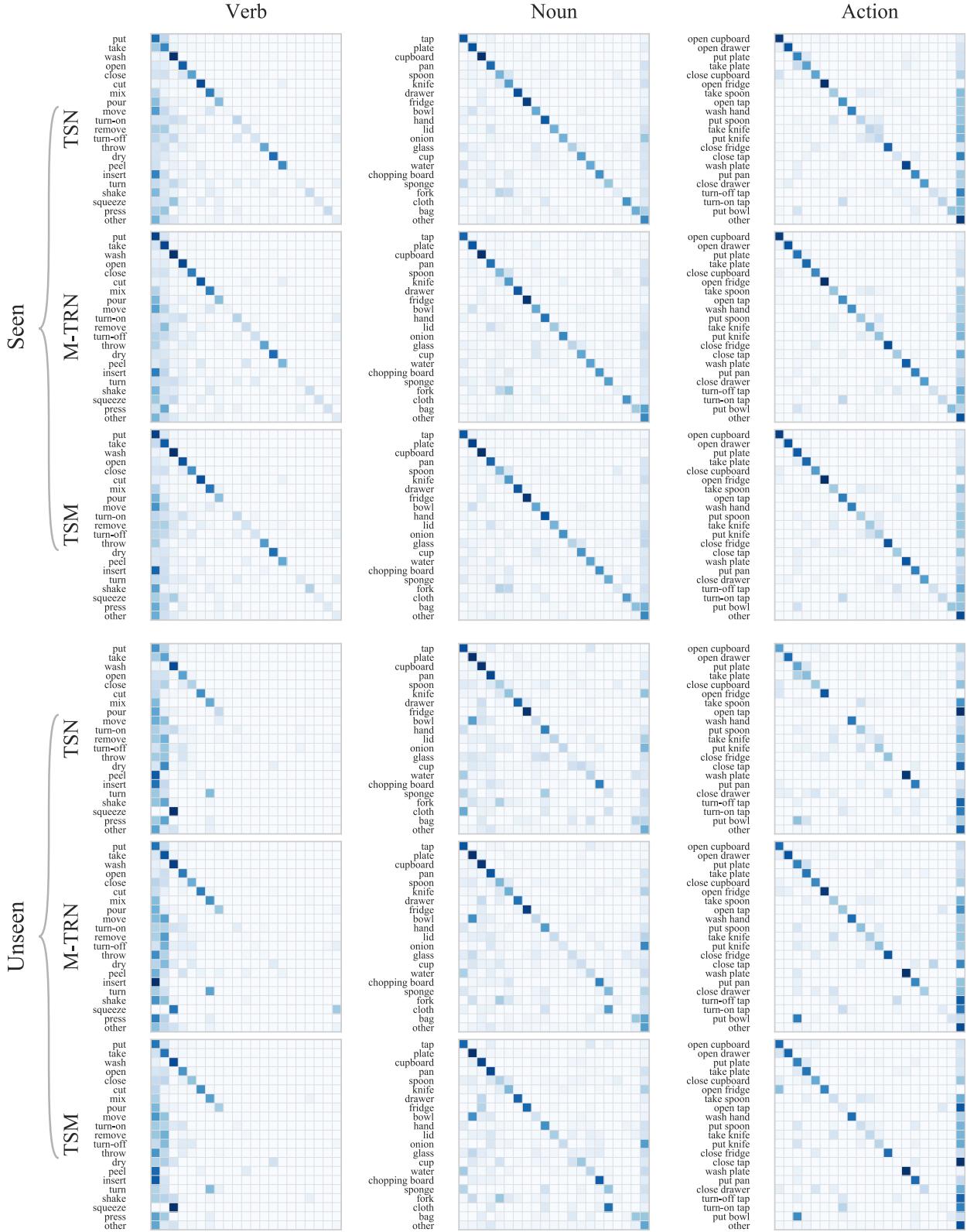


Figure 1: Fusion models' performance on top-20 most frequent classes in training. Classes are ordered from top to bottom in descending order of frequency and any classes outside the top-20 are grouped into a super-class labelled 'other'. [Best viewed on screen]

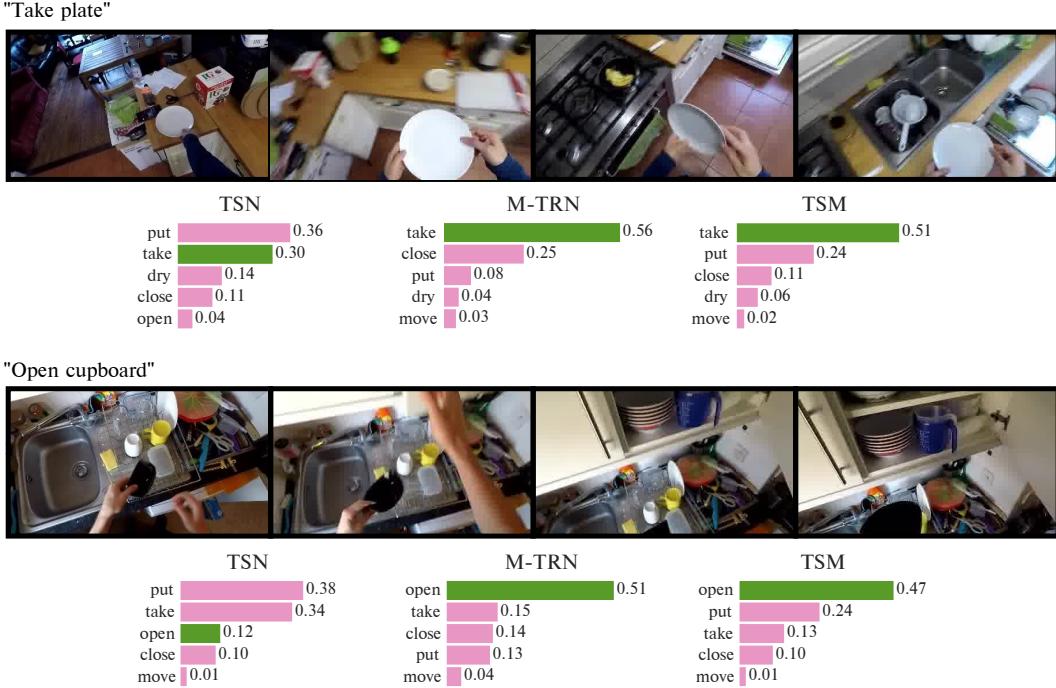


Figure 2: Two examples demonstrating where models capable of temporal reasoning, TRN and TSM, improve over TSN. The bar charts show the model’s scores on the above example with the correct class’ score shown in green.

urates? We examine the answer to this question by training models with different numbers of segments, presenting results in Fig. 3. Overall, flow models benefit more from increasing temporal support, showing monotonically increasing performance, unlike RGB models whose performance saturates at 8 frames, even dropping for the action task when using 16 frames. Curiously, the RGB TSM model is severely harmed by using 16 segments instead of 8, unlike its flow counterpart whose performance improves moving from 8 to 16 segments. This is in contrast to the authors results on Kinetics and Something-something which show an improvement in using 16 frames over 8. This drop was consistently observed across varying LRs suggesting this is not due to a suboptimal learning rate.

Action priors In the previous sections, action predictions have been computed assuming independence between verbs and nouns

$$P(A = (v, n)) = P(V = v)P(N = n), \quad (1)$$

however this is naïve as verb-noun combinations aren’t all as equally likely. For example, it is much more probable to observe ‘cut onion’ than ‘cut chopping board’. In Long-term Feature Banks [13], the authors propose leveraging the prior knowledge of verb-noun co-occurrence in the training set $\mu(v, n)$ to weight the action prediction, *i.e.*

$$P(A = (v, n)) \propto \mu(v, n)P(V = v)P(N = n). \quad (2)$$

Model	Modality	Top-1		Top-5	
		S1	S2	S1	S2
TRN	RGB	+0.05	+1.33	+0.14	+1.43
	Flow	+0.01	+1.43	-0.50	+0.75
M-TRN	RGB	-0.14	+0.99	+0.70	+2.80
	Flow	-0.25	+0.68	-0.61	+0.24
TSM	RGB	+0.02	+0.82	+0.24	+2.42
	Flow	-0.25	+0.89	-0.83	+0.44

Table 2: Percentage point improvement on action task when using action prior across 8-segment ResNet-50 models.

The method in Eq. 2 does not allow zero-shot learning of unseen verb-noun combinations. To remedy this, we apply Laplace smoothing to μ to avoid eliminating the possibility of recognising unseen actions. We evaluate the relative benefit of using action priors in Table 2 finding it provides little benefit on the seen test set, but improves performance on the unseen test set by $\sim 1\%$ point for top-1 accuracy.

4. Released Models

All models required to reproduce the results in Table 1 are made available. We release both RGB and flow models whose predictions can be combined to produce fusion results. To reproduce or compare to these results, the test set predictions should be submitted to the EPIC-Kitchens

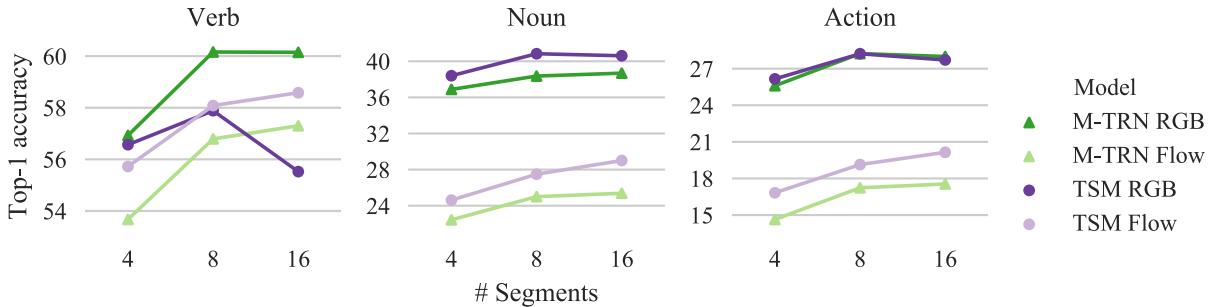


Figure 3: Top-1 accuracy on the seen test set when varying number of segments (during both training/testing) for M-TRN and TSM.

Model	GFLOP/s		Params (M)	
	RGB	Flow	RGB	Flow
TSN	33.12	35.33	24.48	24.51
TRN	33.12	35.32	25.33	25.35
M-TRN	33.12	35.33	27.18	27.21
TSM	33.12	35.33	24.48	24.51

Table 3: Model parameter and FLOP/s count using a ResNet-50 backbone with 8 segments for a single video.

leaderboard² to calculate the performance.

The complexity of the models using ResNet-50 backbone is compared in Table 3.

5. Conclusion

We have benchmarked 3 contemporary models for action recognition and analysed their performance, highlighting areas of good and poor performance. TSM is competitive with M-TRN, and both outperform TSN. These results highlight the necessity for temporal reasoning to recognise actions in EPIC-Kitchens. Yet, the relatively low scores for top-1 accuracy show the challenge is far from solved. Particular issues common to all models are the long-tailed nature of the dataset, fine-grained classes, and difficulty in generalising to unseen environments where we observe a significant drop across all metrics.

References

- [1] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018.
- [2] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017.
- [3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.
- [5] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017.
- [6] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011.
- [7] J. Lin, Chuang. Gan, and S. Han. Temporal shift module for efficient video understanding. *arXiv*, 2018.
- [8] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *NeurIPS*, 2017.
- [9] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NeurIPS*, 2014.
- [10] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild, 2012.
- [11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
- [12] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks for action recognition in videos. *TPAMI*, 2019.
- [13] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krähenbühl, and Ross Girshick. Long-Term Feature Banks for Detailed Video Understanding. In *CVPR*, 2019.
- [14] B. Zhou, A. Andonian, A. Oliva, and A. Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018.

²<https://epic-kitchens.github.io/2019#challenges>

Baidu-UTS Submission to Epic-Kitchens Action Recognition Challenge 2019

Xiaohan Wang^{1,2}, Yu Wu^{1,2}, Linchao Zhu², Yi Yang²

{xiaohan.wang-3, yu.wu-3, linchao.zhu}@student.uts.edu.au; yi.yang@uts.edu.au

¹Baidu Research, ²CAI, University of Technology Sydney

Abstract

In this report, we introduce our submission to the Epic-Kitchens Action Recognition Challenge. We utilize object detection features to guide the training of 3D CNN. Additionally, to strengthen the interaction between the features and avoid gradient exploding, we introduce a Gated Feature Aggregator module. Experimental results demonstrate our approach can significantly improve the action recognition accuracy of egocentric videos.

1. Introduction

Egocentric action recognition is a challenging task. Due to the intense camera moving and the absence of pose information, it is difficult to locate which object human is interacting with. To address this problem, we utilize object-related features to guide 3D CNN. Specifically, we extract the object feature from the context frames using detection models. And this feature is sent to a Gated Feature Aggregator module with the clip feature to produce a new representation for the final classification. This module can stabilize the training process and strengthen the interaction of the two kinds of activations. Our method outperforms the baseline models and achieves the state-of-the-art on the test sets.

2. Our Approach

As shown in Fig. 1, our framework consists of two branches. The first 3D CNN branch takes the sampled video clip as input and produces a clip feature. The second branch aims to extract the object-related features from the context frames. We sampled the frames within a window size w at the center of the current clip. Then the pretrained object detector processes them frame by frame. We choose the top-K bounding boxes with the highest score and use RoIAlign [7] to get the features from the feature maps of the

*Work was done when Xiaohan Wang and Yu Wu interned at Baidu Research. Part of this work was done when Yi Yang was visiting Baidu Research during his Professional Experience Program.

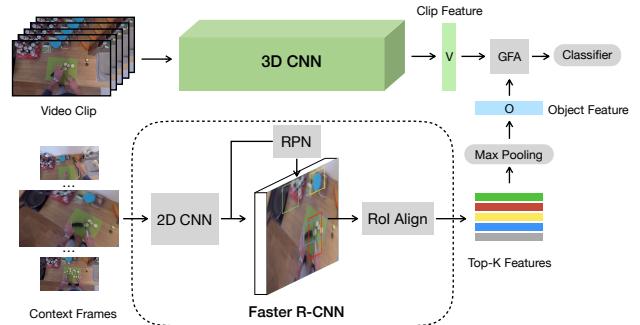


Figure 1. The overall framework of our approach.

2D CNN. After that, the top-K features are max pooled and send to the Gated Feature Aggregator (GFA) module with the clip feature. This module can guide the model to utilize the object-related information and find more discriminative channels. We describe the details of GFA in Sec. 2.2. The output of GFA is our final feature and used to classify verbs and nouns.

2.1. Base Models

We use three 3D CNN backbones to extract video clip features. The first one is I3D [3] which is proposed by Carreira and Zisserman. They inflate 2D CNN architectures to 3D and initialize the network with ImageNet [5] pretrained weights. We use the two stream I3D for verb classification and RGB I3D for noun classification since we suppose optical flow doesn't contain enough information of object appearance. The other two backbones we used are 3D ResNet-50 [6] and 3D ResNeXt-101[6]. They have similar architectures as the 2D models, but all convolutional kernels of them have spatial-temporal three dimensions. All the three 3D CNN backbone are pretrained on the Kinetics-400 dataset [3].

We use Faster R-CNN [11] pipeline to detect objects and extract object features. The backbone of the detector is 2D ResNeXt-101 [13] with FPN [9], which is trained on 1600-class Visual Genome [8, 1] and then finetuned on Epic-Kitchens [4] detection dataset. We train two detectors following the above steps. One is $32 \times 8d$ ResNeXt-101

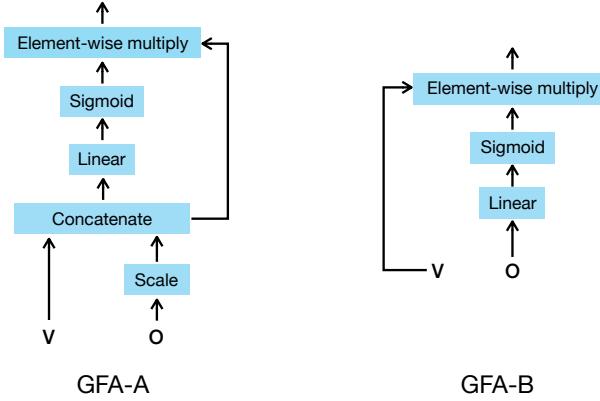


Figure 2. The two different types of GFA.

with 1024-dim output, and the other is $64 \times 4d$ ResNeXt-101 with 2048-dim output.

2.2. Gated Feature Aggregator

Wu et al. [12] propose to concatenate the object feature and the clip feature directly as the final representation. However, in our experiments, this method is sensitive to the backbones of 3D CNN and detector. When the two branches have different backbones, e.g., I3D and ResNext, the training loss is difficult to converge thus the final performance is not improved. To stabilize the training process and leverage the interdependencies of these two features, we design a Gated Feature Aggregator (GFA) module. As illustrated in Fig. 2, GFA has two types.

GFA-A. Since the amplitudes of the object feature o and the clip feature v might be different, we scale o to enforce its amplitude be approximate with v . The scaling operation can be performed by dividing a scalar. Another way of scaling is to multiply the ℓ_2 -normed o by the amplitude of v . After that, the concatenated o and v is transformed to a new representation by self-gating mechanism [10]. Formally, the output feature is computed as follows,

$$F = \sigma(W[v, scale(o)] + b) \cdot [v, scale(o)], \quad (1)$$

where $[]$ indicates concatenation. We have two motivations behind this design. First, we wish to avoid the gradient explosion by scale operation. Second, we want to strengthen the object-related channel using the gating operation.

GFA-B. The instability of the training process is mainly caused by the concatenation operation. In this type, we multiply gated o by v in an element-wise manner instead of concatenation. The final representation F is obtained as follows,

$$F = \sigma(Wo + b) \cdot v. \quad (2)$$

3D CNN	Detector	GFA	top-1	top-5
ResNet-50	-	-	55.62	81.60
ResNeXt-101	-	-	57.43	81.46
I3D RGB	-	-	59.38	82.78
I3D Flow	-	-	56.65	80.79
I3D two-stream	-	-	61.44	83.60
ResNet-50	1024 dim	Type A	57.61	82.64
Fusion	-	-	63.15	84.57

Table 1. Performance of different models for verb recognition on the new train/val set.

3D CNN	Detector	GFA	top-1	top-5
ResNet-50	-	-	25.07	46.84
ResNeXt-101	-	-	25.68	46.52
I3D RGB	-	-	27.92	52.85
ResNet-50	1024 dim	Type A	31.79	56.80
ResNet-50	2048 dim	Type A	32.99	57.81
ResNeXt-101	1024 dim	Type A	30.79	56.69
I3D RGB	1024 dim	Type B	31.14	58.42
I3D RGB	2048 dim	Type B	34.13	60.36
Fusion	-	-	39.09	65.00

Table 2. Performance of different models for noun recognition on the new train/val set.

2.3. Action Re-weighting

The actions are determined by the pairs of verb and noun. The basic method of obtaining the action score is to calculate the multiplication of verb probability and noun probability. However, there are some verb-noun pairs that do not exist in reality, e.g. “open the knife”. Following the approach in [12], we consider the occurrence frequency of action in training set as its prior. The final action probability is re-weighted by the prior.

3. Experiments

3.1. Results

We train our model for verb and noun independently. To validate our models, we split the training data to the new training and validation set following [2].

For verb recognition, as shown in Table 1, we train five different models on the new training set and evaluate their top-1 and top-5 accuracy on the validation set. The two-stream I3D model (late fusion of I3D RGB and I3D flow) obtains the best performance, which can achieve 61.44% top-1 accuracy and 83.60% top-5 accuracy. Our ResNet-50 with GFA improves the top-1 accuracy by 1.99% than the baseline model, where GFA is type A with the norm and scale operation.

For noun recognition, as shown in Table 2, we experiment with eight models on the new train/val set. Due to

Model	data split	re-weighting	verb		noun		action	
			top-1	top-5	top-1	top-5	top-1	top-5
fused model	train/val	w/o	63.15	84.57	39.09	65.00	27.68	48.07
fused model	train/val	w	63.15	84.57	39.09	65.00	28.98	49.78
fused model	trainval/test-s1	w	69.80	90.95	52.27	76.71	41.37	63.59
fused model	trainval/test-s2	w	59.68	82.69	34.14	62.38	25.06	45.95

Table 3. Performance of the fused model on the train/val and trainval/test set.

the large margin improvement of the performance of noun recognition, we try more combinations of 3D CNN, detectors, and GFA. ResNet-50 with 2048-dim detection features and GFA-A results in 7.92% improvement of top-1 accuracy. Besides, I3D RGB model with 2048-dim detection features and GFA-B achieves the highest top-1 accuracy at 34.13%.

For action recognition, as shown in Table 3, we calculate the final action top-1 and top-5 accuracy of our fused model on train/val split in two ways. The re-weighting strategy improves the top-1 accuracy by 1.30% and top-5 accuracy by 1.71%.

For the final submission, we train the above models on the whole training data. Our model ensemble achieves the best performance on both seen(s1) and unseen(s2) test set. The final results are shown in Table 3.

3.2. Implementation

In all experiments, the inputs for 3D CNN are 64-frame video clips. The clips are randomly scaled and cropped to 224×224 . For verb recognition, we randomly sample 64 frames per video segment at the training time and uniformly sample frames for testing. As for noun recognition, we randomly sample the continuous 64 frames with a temporal stride of 2 for training and sample 64 frames around the center of the video segment for testing. The window size of the context frames for detection models is 12 seconds and the sample rate of frames is 2 fps. We choose the top-10 bounding boxes of the context frames to extract object features. We adopt the stochastic gradient descent (SGD) with momentum 0.9 and weight decay 0.0001 to optimize the model for 50 epochs. The overall learning rate is initialized to be 0.01(verb) / 0.003(noun), then dropped ten times at the 30th epoch.

4. Conclusion

In this paper, we report the method details for the Epic-Kitchens action recognition task. We modify the LFB model to stabilize the training process and strengthen the interaction of different activations. Our model achieves the state-of-the-art on both seen and unseen test data.

References

- [1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, 2018.
- [2] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. Object level visual reasoning in videos. In *ECCV*, 2018.
- [3] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.
- [4] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018.
- [5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
- [6] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*, 2018.
- [7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. In *ICCV*, 2017.
- [8] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 2016.
- [9] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.
- [10] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017.
- [11] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *T-PAMI*, 2015.
- [12] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *CVPR*, 2019.
- [13] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

Temporal Binding Network (TBN)

EPIC-Kitchens Action Recognition Challenge Entry

Evangelos Kazakos
University of Bristol

Arsha Nagrani
University of Oxford

Andrew Zisserman
University of Oxford

Dima Damen
University of Bristol

Abstract

We propose a method to fuse multiple modalities (RGB, Flow and Audio) for egocentric (i.e. first-person) action recognition, through a novel architecture for temporal-binding. Our temporal binding network (TBN) fuses modalities within a range of temporal offsets, along with mid-level fusion and sparse temporal sampling of fused representations. TBN is trained end-to-end, outperforming individual modalities as well as late-fusion of modalities.

Our method achieves state of the art results on both the seen and unseen test sets of EPIC-Kitchens.

1. Introduction

In this work, we explore audio as a prime modality to provide complementary information to visual modalities (appearance and motion) in egocentric action recognition. While audio has been explored in video understanding in general [1, 2, 8, 9, 11], the egocentric domain in particular offers rich sounds resulting from the interactions between hands and objects, as well as the close proximity of the wearable microphone to the undergoing action. Audio is a prime discriminator for some actions (e.g. ‘wash’, ‘fry’) as well as objects within actions (e.g. ‘put plate’ vs ‘put bag’). At times, the temporal progression (or change) of sounds can separate visually ambiguous actions (e.g. ‘open tap’ vs ‘close tap’). Audio can also capture actions that are out of the wearable camera’s field of view, but audible (e.g. ‘eat’ is heard but not seen). Conversely, other actions are *soundless* (e.g. ‘wipe hands’) and the wearable sensor might capture irrelevant sounds, such as talking or music playing in the background. The opportunities and challenges of incorporating audio in egocentric action recognition allow us to explore new multi-sensory fusion approaches, particularly related to the potential *temporal asynchrony* between the action’s appearance and the discriminative audio signal – the main focus of our work.

In Fig. 1, we show an example of ‘breaking an egg’ from the EPIC-Kitchens dataset. The distinct sound of cracking the egg, the motion of separating the egg

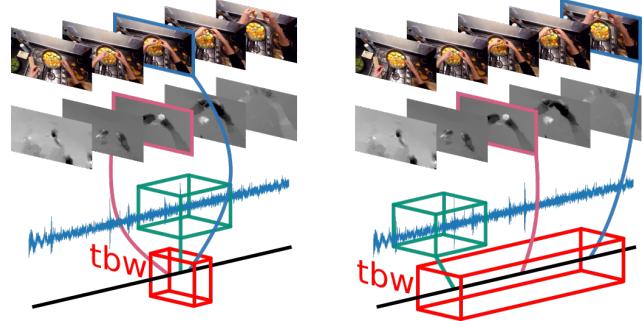


Figure 1: As the width of the temporal binding window increases (left to right), modalities (appearance, motion and audio) are fused with varying temporal shifts.

and the change in appearance of the egg occur at different frames/temporal positions within the video. Approaches that fuse modalities with synchronised input would thus be limited in their ability to learn such actions. In this work, we explore fusing inputs within a temporal window, which we refer to as the Temporal Binding Window (TBW) (Fig 1), allowing the model to train using asynchronous inputs from the various modalities. Evidence in neuroscience and behavioural sciences points at the presence of such a TBW for multiple sensory modalities in humans [10, 12, 13].

To the best of our knowledge, the only work that has explored fusion for action recognition using three modalities (appearance, motion and audio) is [16], employing late-fusion of predictions. Tested on UCF101, the work shows audio to be the least informative modality for third person action recognition (16% accuracy for audio compared to 80% and 78% for spatial and motion). A similar conclusion was made for other third-person datasets (AVA [5]). In this work, we show audio to be a competitive modality for egocentric AR on EPIC-Kitchens, achieving comparable performance to appearance. We also report state-of-the-art performance using mid-level fusion within a temporal binding window.

2. The Temporal Binding Network

Our goal is to find the optimal way to fuse multiple modality inputs while modelling temporal progression through sampling. We first explain the general notion of temporal binding of multiple modalities in Sec 2.1, then detail our architecture in Sec 2.2.

2.1. Multimodal Temporal Binding

Consider a sequence of samples from one modality in a video stream,

$$m_i = (m_{i1}, m_{i2}, \dots, m_{iT/r_i}) \quad (1)$$

where T is the video's duration and r_i is the modality's framerate (or frequency of sampling). Input samples are first passed through unimodal feature extraction functions f_i . To account for varying representation sizes and frame-rates, most multi-modal architectures apply pooling functions G to each modality in the form of average pooling or other temporal pooling functions, before attempting multimodal fusion. Given a pair of modalities m_1 and m_2 , the final class predictions for a video are hence obtained as follows:

$$y = h(G(f_1(m_1)), G(f_2(m_2))) \quad (2)$$

where f_1 and f_2 are unimodal feature extraction functions, G is a temporal aggregation function, h is the multimodal fusion function and y is the output label for the video. In such architectures (e.g. TSN [14]), modalities are temporally aggregated for a prediction before different modalities are fused; this is typically referred to as ‘late fusion’.

Conversely, multimodal fusion can be performed at *each* time step as in [4]. One way to do this would be to synchronise modalities and perform a prediction at *each* timestep. For modalities with matching framerates, synchronised multi-modal samples can be selected as (m_{1j}, m_{2j}) , and fused according to the following equation:

$$y = h(G(f_{sync}(m_{1j}, m_{2j}))) \quad (3)$$

where f_{sync} is a multimodal feature extractor that produces a representation for each time step j , and G then performs temporal aggregation over all time steps. When frame rates vary, and more importantly so do representation sizes, only approximate synchronisation can be attempted,

$$y = h(G(f_{sync}(m_{1j}, m_{2k}))) : k = \lceil \frac{jr_2}{r_1} \rceil \quad (4)$$

We refer to this approach as ‘synchronous fusion’ where synchronisation is achieved or approximated.

In this work, however, we propose fusing modalities within temporal windows. Here modalities are fused within a range of temporal offsets, with all offsets constrained to

lie within a finite time window, which we henceforth refer to as a temporal binding window (TBW). Interestingly, as the number of modalities increases, say from two to three modalities, the TBW representation allows fusion of modalities each with different temporal offsets, yet within the same binding window $\pm b$:

$$\begin{aligned} y = h(G(f_{tbw}(m_{1j}, m_{2k}, m_{3l})) : k \in [\lceil \frac{jr_2}{r_1} - b \rceil, \lceil \frac{jr_2}{r_1} + b \rceil] \\ : l \in [\lceil \frac{jr_3}{r_1} - b \rceil, \lceil \frac{jr_3}{r_1} + b \rceil] \end{aligned} \quad (5)$$

Sampling within a temporal window allows fusing modalities with various temporal shifts, *up to* the temporal window width $\pm b$. This is different from proposals that fuse inputs over predefined temporal differences (e.g. [7]).

2.2. TBN with Sparse Temporal Sampling

First, the action video is divided into K segments of equal width, to allow for sparse temporal sampling and thus for modelling the temporal progression of the action, as with previous works [14, 17]. Within each segment, we select a random sample of the first modality $\forall k \in K : m_{1k}$. This ensures the temporal progression of the action is captured by sparse temporal sampling of this modality, while random sampling within the segment offers further data for training.

The sampled m_{1k} is then used as the centre of a TBW of width $\pm b$. The other modalities are selected randomly from within each TBW (Eq. 5). Note that the K temporal binding windows could be overlapping. In total, the input to our architecture in both training and testing is $K \times M$ samples from M modalities.

A ConvNet (per modality) is used to extract *mid-level* features, which are then fused through *concatenating* the modality features and feeding them to a fully-connected layer, making multi-modal predictions per TBW. We back-propagate all the way to the inputs of the ConvNets. The convolutional weights for each modality are shared over the K segments. Additionally, the mid-level fusion weights and class predictions are also shared across the segments.

3. Experiments

RGB and Flow: We use the publicly available RGB and computed optical flow with the dataset [3].

Audio Processing: We extract 1.28s of audio from the untrimmed video, convert it to single-channel, and resample it to 24kHz. We then convert it to a log-spectrogram representation using an STFT of window length 10ms, hop length 5ms and 256 frequency bands. This results in a 2D spectrogram matrix of size 256×256 , after which we compute the logarithm.

Training details: We use Inception with Batch Normalisation (BN-Inception) [6] as a base architecture, and fuse

		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	Audio	43.56	22.35	14.21	79.66	43.68	27.82	32.28	19.10	07.27	25.33	18.16	06.17
	TBN Single Model	64.75	46.03	34.80	90.70	71.34	56.65	55.67	43.65	22.07	45.55	42.30	21.31
	TBN Ensemble	66.10	47.89	36.66	91.28	72.80	58.62	60.74	44.90	24.02	46.82	43.89	22.92
S2	Audio	35.43	11.98	06.45	69.20	29.49	16.18	22.46	09.41	04.59	18.02	09.79	04.19
	TBN Single Model	52.69	27.86	19.06	79.93	53.78	36.54	31.44	21.48	12.00	28.21	23.53	12.69
	TBN Ensemble	54.46	30.39	21.99	81.23	55.69	40.60	32.57	21.68	09.83	27.60	25.58	13.53

Table 1: We show how audio-only is a strong modality for EPIC-Kitchens then report single model and ensemble results for seen (S1) and unseen (S2) test splits.

the modalities after the average pooling layer. We train the networks using SGD with momentum, using a batch size of 128, a dropout of 0.5, a momentum of 0.9, and a learning rate of 0.01. Note that our network is trained end-to-end for all modalities and TBWs. We train with $K = 3$ segments over the $M = 3$ modalities.

TBN Ensemble: We report results of an ensemble of 5 TBNs, where each one is trained with different TBW widths.

Adding prior knowledge: Following [15], we also compute the probability of an action as the product of the softmax and the class a prior. We found this to only be beneficial for the unseen test set S2, and thus report TBN-Ensemble using this prior for S2 only.

Leaderboard results are shown in Table 1. We also include the audio stream results to signify its importance in EPIC-Kitchens. At the closing of the challenge, TBN is ranked second on the leaderboard for S1 and third for S2. Table 1 shows the reported results on all metrics.

Challenge entry: As two authors are prime contributors to EPIC-Kitchens collection and running the challenge, we are not officially competing in the challenge. However, we wish to note that we did not use any of the test set annotations in optimising our results.

4. Conclusion

We have shown that the TBN architecture is able to flexibly combine the RGB, flow and audio modalities to achieve an across the board performance improvement, compared to individual modalities. Our results are highly competitive on the final leaderboard.

References

- [1] R. Arandjelovic and A. Zisserman. Look, listen and learn. In *ICCV*, 2017. 1
- [2] Y. Aytar, C. Vondrick, and A. Torralba. See, hear, and read: Deep aligned representations. *CoRR*, abs/1706.00932, 2017. 1
- [3] D. Damen, H. Doughty, G. Maria 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. 2
- [4] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 2
- [5] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman. A better baseline for ava. In *ActivityNet Workshop at CVPR*, 2018. 1
- [6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015. 2
- [7] W. Lin, Y. Mi, J. Wu, K. Lu, and H. Xiong. Action recognition with coarse-to-fine deep feature integration and asynchronous fusion. *AAAI*, 2018. 2
- [8] A. Nagrani, S. Albanie, and A. Zisserman. Seeing voices and hearing faces: Cross-modal biometric matching. In *CVPR*, 2018. 1
- [9] A. Owens and A. A. Efros. Audio-visual scene analysis with self-supervised multisensory features. In *ECCV*, 2018. 1
- [10] C. Parise, C. Spence, and M. O. Ernst. When correlation implies causation in multisensory integration. *Current Biology*, 22(1):46 – 49, 2012. 1
- [11] A. Senocak, T.-H. Oh, J. Kim, M.-H. Yang, and I. So Kweon. Learning to localize sound source in visual scenes. In *CVPR*, 2018. 1
- [12] R. A. Stevenson, M. M. Wilson, A. R. Powers, and M. T. Wallace. The effects of visual training on multisensory temporal processing. *Experimental Brain Research*, 225(4):479–489, 2013. 1
- [13] M. T. Wallace and R. A. Stevenson. The construct of the multisensory temporal binding window and its dysregulation in developmental disabilities. *Neuropsychologia*, 64:105 – 123, 2014. 1
- [14] 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. 2
- [15] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krähenbühl, and R. Girshick. Long-Term Feature Banks for Detailed Video Understanding. In *CVPR*, 2019. 3
- [16] Z. Wu, Y.-G. Jiang, X. Wang, H. Ye, and X. Xue. Multi-stream multi-class fusion of deep networks for video classification. In *ACM International Conference on Multimedia*, 2016. 1
- [17] B. Zhou, A. Andonian, A. Oliva, and A. Torralba. Temporal relational reasoning in videos. In *The European Conference on Computer Vision (ECCV)*, 2018. 2

FBK-HUPBA Submission to the EPIC-Kitchens 2019 Action Recognition Challenge

Swathikiran Sudhakaran¹, Sergio Escalera^{2,3}, Oswald Lanz¹

¹Fondazione Bruno Kessler, Trento, Italy

²Computer Vision Center, Barcelona, Spain

³Universitat de Barcelona, Barcelona, Spain

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

Abstract

In this report we describe the technical details of our submission to the EPIC-Kitchens 2019 action recognition challenge. To participate in the challenge we have developed a number of CNN-LSTA [3] and HF-TSN [2] variants, and submitted predictions from an ensemble compiled out of these two model families. Our submission, visible on the public leaderboard with team name FBK-HUPBA, achieved a top-1 action recognition accuracy of 35.54% on S1 setting, and 20.25% on S2 setting.

1. Introduction

Action recognition from videos is one of the most important and ever growing research areas in computer vision. The applications of action recognition range from video surveillance to robotics, human-computer interaction, video indexing and retrieval, *etc*. The availability of graphics processing units (GPUs) and large scale datasets have resulted in the development of several data-driven techniques for action recognition via deep learning. EPIC-Kitchens dataset [1] consists of egocentric videos. Recognition of actions classes in this dataset is challenged by the need for a fine-grained discrimination of small objects and their manipulation.

For our participation to the challenge we considered two different approaches with complementary feature encoding perspective for classifying action categories:

- CNN-LSTA [3]: late (and shallow) aggregation of frame level features with a variant of LSTM;
- HF-Nets [2]: early (and deep) aggregation of frame level features using a temporal gating mechanism.

Fig. 1 shows block diagrams of the two different approaches, both of them developed by the FBK-HUPBA

team. For a detailed presentation of the two baseline methods we refer the reader to the original papers [3, 2].

To participate in the challenge we have developed variants of both CNN-LSTA and HF-TSN baselines. We have changed backbone CNNs, enriched the aggregation scheme of LSTA, implemented a structured prediction, and differentiated training strategies. We finally compiled an ensemble out of this pool of trained models. Our submission visible on the public leaderboard was obtained by averaging classification scores from ensemble members.

2. CNN-LSTA and variants

Our first family of models is CNN-RNN structured. The RNN is a Long Short-Term Attention (LSTA) recurrent unit [3]. In brief, LSTA extends LSTM with built-in attention and a revised output gating. Attention is introduced to promote discriminative features in the memory updating. This is done by applying a spatial weight map to the input. Output pooling provides more flexibility in localizing and propagating the active memory components.

We have modified CNN-LSTA baseline as follows:

- Backbone: we used ResNet-34, ResNet-50, InceptionV3;
- Pre-training: we utilized pretrained models on ImageNet and Kinetics;
- Aggregation: we used LSTA internal memory as aggregated descriptor for classification as in [3], but we also aggregated the sequence of output states using GRU and concatenated its final memory state with that of the LSTA for classification.

For the variant with Gated Recurrent Unit (GRU), the output state of LSTA during each time step is spatial average pooled and applied to two GRUs. The output states of

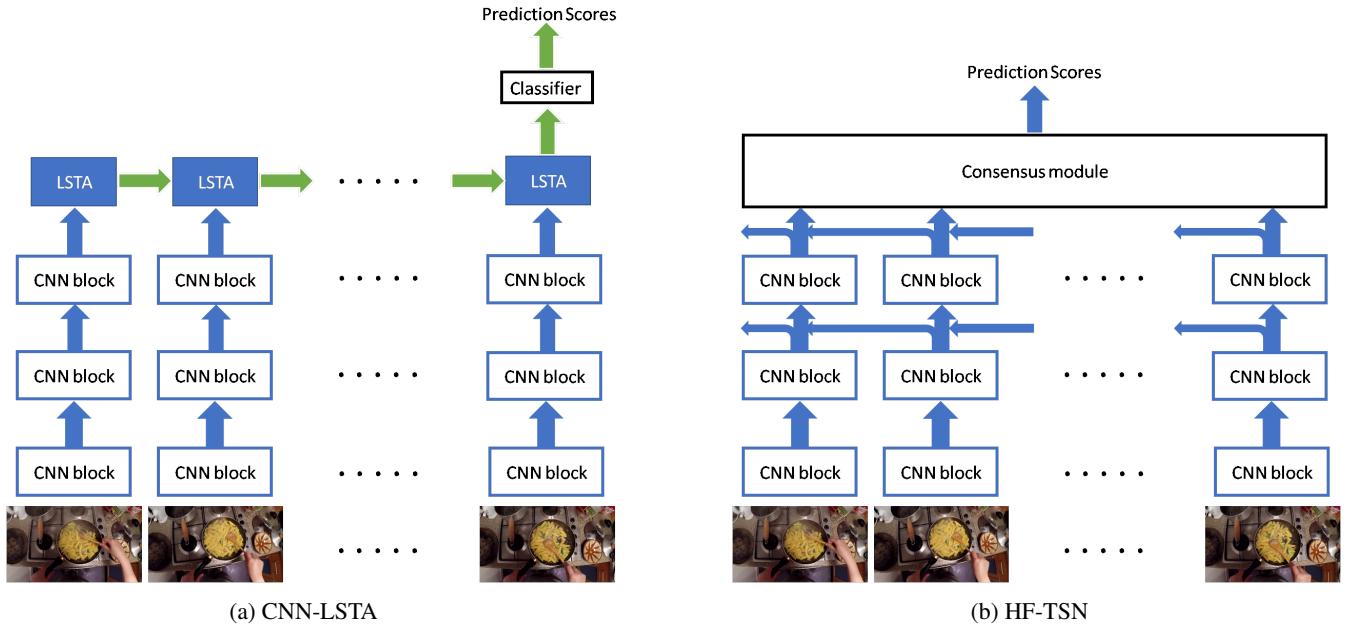


Figure 1: Block diagram illustrating the two model families used for generating the action recognition scores. The first model in Fig. 1a uses a Long Short-Term Attention (LSTA) module to aggregate the frame level features obtained from a Convolutional Neural Network (CNN) backbone. This is equivalent to late fusion of the frame level features. In the second method, Fig. 1b, features from adjacent frames are combined as the inputs move across the CNN layers, followed by a late fusion of the features obtained at the final layer of the CNN. Thus, the two considered approaches provide complementary ways to aggregate frame level features.

the GRUs after encoding all the video frames are then concatenated to predict the `verb`, `noun` and `action` classes. The scores generated from GRU and LSTA are then averaged to obtain the corresponding class scores. Structured prediction is detailed in Sec. 4.

3. HF-TSN and variants

Our second pool are TSN models with hierarchical feature aggregation [2]. In HF-TSN, features from adjacent frames of a video interact with each other as the features are being passed along the layers of a CNN. The interactions are learned and comprises of either differencing or averaging operation, or a mixture of them, via a convolutional layer. The features corresponding to each spatio-temporal receptive field, obtained at the final layer of the CNN, are applied to a linear layer and averaged to obtain the action class score. The consensus module in Fig. 1b represents the linear layer followed by averaging operation.

We have modified HF-TSN baseline as follows:

- Backbone: we used ResNet-50 and BNInception.

For the model with ResNet-50, HF blocks are applied at the input of each of the ResNet-50 blocks. Thus a total of 16 HF blocks are present in this variant as opposed to the 10 present in the model with BNInception.

4. Structured prediction

The labels provided with the dataset are in the form of `verb` and `noun` pairs. An action is defined by the combination of such `verb-noun` pairs. So the network should be able to either correctly predict both the `verb` and `noun` classes in order to combine them into an `action` class, or directly predict the `action` class from which `verb` and `noun` classes can be derived. We trained all the networks as a multi-task classification problem predicting `verb`, `noun` and `action` classes. We generated `action` classes from the combination of `verb` and `noun` labels present in the dataset. It is important to note that not all combinations of `verb-noun` pairs are valid, such as, `take-fridge`, `open-carrot`, `cut-salt` are unfeasible.

In order to model such inter-dependencies among the `verb` and `noun` classes, we apply the `action` prediction scores as an instance-specific bias term to the `verb` and `noun` classifiers. For this, the `action` scores are applied through two linear layers each to map to the number of `verb` and `noun` classes. The result is then applied to the output of the corresponding classifier (`verb` and `noun`). This allows the network to learn the dependencies between the `verb` and `noun` classes and prevent it from making unfeasible predictions consisting of implausible `verb` and

noun combinations. The drawback of this approach is that we are bound to predict action classes observed during training.

5. Cross-modal fusion

For LSTA model, we also implement a two stream model with cross-modal fusion. We follow the approach proposed in [3] for the two stream implementation.

The LSTA model with ResNet-34 CNN is used as the appearance stream. For the motion stream, we first trained a ResNet-34 CNN pre-trained on ImageNet for predicting verb classes followed by a separate training stage for predicting verb, noun and action classes. A stack of optical flow images corresponding to 5 consecutive frames is used as the input to the network. The first convolutional layer of the network is modified to accept an input image with 10 channels and the weights are initialized by averaging the weights from the three channels of the original network.

Once the appearance and motion stream networks are trained separately, we combine them using cross-modal fusion and fine-tune the parameters. In order to perform cross-modal fusion, we first add a Convolutional Long Short-Term Memory (ConvLSTM) layer, with a hidden size of 512, after the `conv5_3` layer of the motion stream. Then the outputs corresponding to each of the frames from the `conv5_3` layer of the appearance stream are combined using a 3D convolution layer, which is applied as bias to the gates of the ConvLSTM layer. Similarly, the output from the `conv5_3` layer of the motion stream is applied as bias to the gates of the LSTA layer present in the appearance stream. Finally, the classification scores from the two individual streams are averaged to obtain the final prediction score of the video.

6. Training details

In this section we provide details on the training protocol. We did not use a held-out validation set for hyperparameter search or model validation.

6.1. CNN-LSTA variants

We used the same training strategy presented in LSTA [3], *i.e.* the networks are trained in two stages. In the first stage, the classification layers and LSTA layer (and the GRUs in the case of variant 3) are trained for 200 epochs starting with a learning rate of 0.001 which is decayed by a factor of 0.1 after 25, 75 and 150 epochs. During stage 2, the `conv5_x` layer in the case of ResNet family of CNNs or `Mixed_7x` layers in the case of InceptionV3, are trained in addition to the layers trained during stage 1. Stage 2 training is done for 150 epochs with an initial learning rate of 0.0001 which is decayed by a factor of 0.1 after 25 and 75 epochs. A dropout

of 0.7 is used to avoid overfitting. ADAM algorithm is used for the optimization of the parameters with a batch size of 32 during training. 20 frames selected uniformly across time are used as the input during both training and evaluation stages. We use random scaling and horizontal flipping as data augmentation techniques during training and during evaluation, we average the scores obtained from five crops (four corner crops and the center crop) and their horizontally flipped versions. In all the models, LSTA and GRU with a memory size of 512 is used. The dimension of the input to the ResNet models is 224×224 and for InceptionV3 is 299×299 .

6.2. HF-TSN variants

The models are trained for 120 epochs with an initial learning rate of 0.01 that is decayed by a factor of 0.1 after 50 and 100 epochs. We used a batch size of 32 and dropout of 0.5 to prevent overfitting. Stochastic Gradient Descent (SGD) is used as the optimization algorithm. Spatial scaling and random horizontal flipping with temporal jittering is used as data augmentation techniques. During evaluation, 10 image crops are generated from each frame using cropping and horizontal flipping and their average of scores is used for predicting the action class of the video. 16 frames are sampled from each video during training and inference. The input image dimension is set as 224×224 .

6.3. Two-stream variants

For the flow stream, the network is trained for 700 epochs, for verb classification, with an initial learning rate of 0.01 which is reduced by 0.5 after 75, 150, 250 and 500 epochs. This acts as a pre-training for the network. After this, we train the network for action classification with the same structured prediction technique explained in 4. We also apply spatial attention to the features at the output of the `conv5_3` layer. We follow the idea proposed in [4] for applying spatial attention to the motion features. During this stage, the network is trained 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 in both stages.

For the two stream model, the networks are finetuned for 100 epochs with a learning rate of 0.01 using ADAM algorithm. Learning rate is reduced by a factor of 0.99 after each epoch. We finetune the classification layers, LSTA, ConvLSTM and `conv5_x` layers of the two networks in this stage.

7. Results

The recognition accuracy obtained for each of the selected models and their ensemble are listed in Tab. 1. Since no validation set is provided with the dataset, we choose models for ensembling based on their design variability.

	Method	Backbone	Top-1 Accuracy (%)			Top-5 Accuracy (%)			Precision (%)			Recall (%)		
			Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
S1	LSTA	Res-34	58.25	38.93	30.16	86.57	62.96	50.16	44.09	36.30	16.54	37.32	36.52	19.00
		Res-50	57.81	37.84	29.54	86.14	63.63	49.82	52.76	34.77	16.35	33.94	34.46	18.05
		Res-50 [†]	57.69	39.36	29.79	86.77	64.46	50.52	50.83	36.49	17.54	33.68	35.70	17.38
		IncV3	57.28	39.32	29.35	86.43	64.32	50.18	54.77	36.08	14.51	34.29	35.64	16.65
	LSTA-GRU	Res-50*	57.30	37.59	29.17	85.88	62.97	49.24	49.32	34.79	16.81	34.84	34.33	18.40
		Res-34**	60.61	40.84	32.04	87.71	65.93	52.75	53.62	37.29	18.74	36.75	37.30	19.76
		Res-34***	61.31	40.93	32.14	87.47	65.28	52.60	50.93	38.23	19.59	37.90	37.47	20.36
	LSTA-2S	Res-34	62.12	40.41	32.60	87.95	64.47	52.85	52.70	39.66	15.95	36.34	36.88	18.61
	HF-TSN	BNInc	57.57	39.90	28.09	87.83	65.37	48.63	49.12	35.83	11.38	39.37	37.04	13.84
		Res-50	56.69	40.70	29.38	86.47	63.91	49.36	41.88	37.91	10.70	37.86	38.52	13.58
	Ensemble		63.34	44.75	35.54	89.01	69.88	57.18	63.21	42.26	19.76	37.77	41.28	21.19
S2	LSTA	Res-34	45.51	23.46	15.88	75.25	43.16	30.01	26.19	17.58	8.44	20.80	19.67	11.29
		Res-50	44.38	22.53	15.98	74.29	43.02	30.42	23.36	17.69	7.31	17.39	17.92	10.29
		Res-50 [†]	43.53	22.98	16.25	74.70	44.66	30.01	22.05	15.70	7.81	15.73	17.62	10.83
		IncV3	44.66	23.76	17.31	75.35	47.97	32.64	24.69	17.80	7.70	16.10	19.38	11.19
	LSTA-GRU	Res-50*	43.94	22.16	15.94	73.61	42.47	29.70	23.20	17.84	8.24	17.04	17.71	10.27
		Res-34**	45.37	23.49	16.59	74.74	45.24	31.17	30.04	16.05	7.51	16.38	17.93	10.23
		Res-34***	44.90	22.60	16.25	74.80	44.62	31.14	32.62	16.45	7.87	17.99	19.41	10.53
	LSTA-2S	Res-34	48.89	24.27	18.71	77.88	46.06	33.77	27.12	20.12	9.29	22.59	18.91	12.91
	HF-TSN	BNInc	42.40	25.23	16.93	75.76	48.96	33.32	24.25	20.48	6.29	15.77	21.96	10.05
		Res-50	45.48	24.55	17.38	75.32	46.91	33.32	29.44	22.94	7.44	19.11	21.05	10.68
	Ensemble		49.37	27.11	20.25	77.50	51.96	37.56	31.09	21.06	9.18	18.73	21.88	14.23

Table 1: Comparison of recognition accuracies with state-of-the-art in EPIC-KITCHENS dataset. [†]: Kinetics pre-trained; *: finetuned layers- GRU; **: finetuned layers- GRU+LSTA; ***- finetuned layers- GRU+LSTA+Conv5_3

Each selected model has been submitted for evaluation on the test server. Model ensembling is done by averaging the prediction scores obtained from individual models. We participated to the challenge with the ensemble.

The best performance obtained for S1 using RGB frames is by the LSTA model with GRUs encoding the output state of LSTA. The model resulted in a recognition accuracy of 32.14%. Using the cross-modal fusion technique explained in Sec. 5, the recognition accuracy improved by 2% (30.16 vs 32.60). By combining the LSTA-2S and HF-TSN-BNInception models, an improvement of 1% is obtained. With an ensemble of all the models, the action recognition accuracy is further improved by 2%.

In S2 setting, the best performance using RGB frames as input was obtained by HF-TSN model with ResNet-50 backbone (17.38%). A gain of about 3% is obtained using cross-modal fusion over the LSTA model. A gain of 2% is obtained from an ensemble of LSTA-2S and HF-TSN-BNInception. The ensemble of all the models resulted in an accuracy of 20.25%. This proves that the selection of models based on the difference in training settings and temporal encoding techniques was beneficial.

8. Conclusions

We described the details of the two model families and their variants we ensembled for our submission to the action

recognition task of the EPIC-Kitchens CVPR 2019 challenge. The recognition accuracy obtained shows that the two model families perform complementary temporal encoding of features. With an ensemble of the proposed methods, our entry to the challenge achieved the score of 35.54% on S1 setting, and 20.25% on S2 setting.

References

- [1] D. Damen, H. Doughty, G. Maria 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. [1](#)
- [2] S. Sudhakaran, S. Escalera, and O. Lanz. Hierarchical Feature Aggregation Networks for Video Action Recognition. *arXiv preprint arXiv:1905.12462*, 2019. [1, 2](#)
- [3] S. Sudhakaran, S. Escalera, and O. Lanz. LSTA: Long Short-Term Attention for Egocentric Action Recognition. In *Proc. CVPR*, 2019. [1, 3](#)
- [4] S. Sudhakaran and O. Lanz. Attention is All We Need: Nailing Down Object-centric Attention for Egocentric Activity Recognition. In *Proc. BMVC*, 2018. [3](#)

Team NTU-CML-MiRA EPIC-Kitchens Challenge 2019 Technical Report

Zhe-Yu Liu Ya-Liang Chang Chih-Hung Liang Yun-Hsuan Liu Ke-Jyun Wang
Winston Hsu
National Taiwan University, Taipei, Taiwan
{zhe2325138, yaliangchang, r06922057}@cmlab.csie.ntu.edu.tw,
{tmpss1131, rr123789456tw}@gmail.com, whsu@ntu.edu.tw

Abstract

This report describes our submission to EPIC-Kitchens challenge 2019 for the action recognition and anticipation tracks. We experiment both early and late fusion strategies on RGB and optical flow modalities and tried to use hand detection and joint positions as auxiliary modalities to enhance the results. In addition, we find that different backbones have their own advantages and use InceptionV1 with I3D and Resnet50 with Temporal Shift Module to predict the verb and noun classes, respectively. Our method finally achieves top1 accuracy of 61.65%, 43.63%, and 30.55% respectively in verb, noun, and action prediction in the seen kitchen set.

1. Introduction

EPIC-Kitchens [?] is a large egocentric video dataset containing annotations of daily activities in the kitchen.

In this work, we investigate the pros and cons of different modalities, backbones, and fusion strategies for better video action recognition results.

2. Data Preparation

2.1. Modality

The challenge organizers provide RGB video frames and optical flow computed by TV-L1 algorithm [?], and our experiments are based on the two modalities. Other potentially helpful modalities we have tried but not included in our final submission will be discussed in section 9.

2.2. Validation Splitting

The validation splitting scheme follows [?]. We use videos from participant id 5, 6, and 7 as our **unseen** validation set, which consists of 9.2% of training data in terms of video number. The **seen** validation set is randomly sampled from the remaining training data and is composed of around 10% of the training videos.

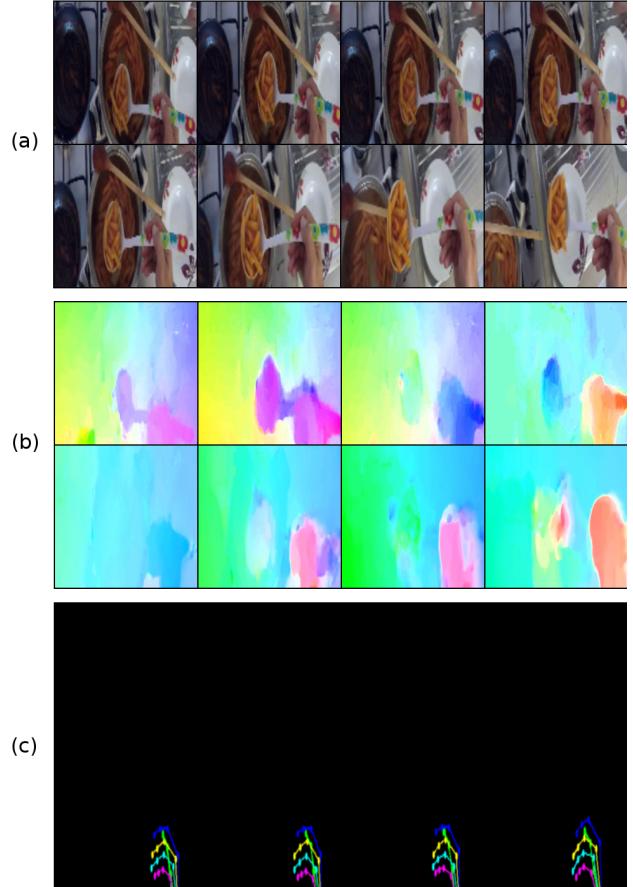


Figure 1. Examples of different modalities. (a) RGB frames. (b) Optical flows. (c) Hand poses. Note that the RGB frames and optical flows are given by the EpicKitchen dataset, while the hand poses are derived by the process in Section 9.1. The hand detection model is not fine-tuned so hand poses are missing in some frames.

3. Backbones and Performance Comparison

At the beginning of the contest, we tried different kinds of models as our prediction backbones and found their performance differs when it comes to the prediction of verb

classes or noun classes.

InceptionV1 with I3D: Inflated 3D Convolution (I3D) proposed by Carreira *et al.* [?] is proven to be a powerful architecture in the action recognition task. We apply the inflation technique on top of the Inception model [?] with weights pre-trained on Kinetics dataset [?] as one of our backbones.

Resnet50 with TSM: Temporal Shift Module (TSM), proposed by Lin *et al.* [?], shifts the feature map along the temporal dimension for a portion of channels to acquire temporal receptive fields by only 2D convolution layers. Because of its simplicity and effectiveness, we also apply TSM on top of Resnet50 [?] with weights pre-trained on Kinetics to compare the performance with other backbones.

Resnet50 with TBN: Besides the models mentioned above, we also tried to insert numerous temporal bilinear modules designed by Li *et al.* [?] with Bottleneck Wide TB setting into Resblocks of Resnet18 and Resnet50. As the Wide TB blocks shift the temporal dimension and perform element-wise multiplication, the model can make use of the temporal information from adjacent frames. However, as the TB block needs much more parameters even with bottleneck layers, we can not apply it on each Resblocks in Resnet50. Eventually we only add 2 TB blocks to the Resblocks in *res3* of Resnet50 according to [?].

Performance comparison: In our experiments, all backbones are supervised with the noun or verb classes separately. The result in table 1 and 2 demonstrate that InceptionV1 I3D performs best for verb prediction, and Resnet50 with TSM achieve the best accuracy in terms of noun prediction. Since the performance gaps between the best and second-best models are large, we decide to use different models for the classification of verb and noun.

Top1 Acc. (%)	Seen Verb	Seen Noun
InceptionV1 I3D	47.71	23.15
Resnet50 TSM	40.75	32.78
Resnet50 TBN	-	25.09

Table 1. Top1 accuracy of each backbone trained with RGB modality on seen validation set.

Top1 Acc. (%)	Unseen Verb	Unseen Noun
InceptionV1 I3D	40.82	11.70
Resnet50 TSM	35.14	18.37
Resnet50 TBN	-	9.42

Table 2. Top1 accuracy of each backbone trained with RGB modality on unseen validation set.

4. Fusion

In order to fully utilize the complementary information among different modalities, we conducted experiments on

the following fusion strategies. Note that due to the lack of time, we only tried RGB and optical flow modalities.

Input Fusion: in this manner, we concatenate RGB and flow to form 5-channel input videos and change the in-channel number of the first convolution layer to adapt 5-channel videos. The model weights are initialized from the optical flow pre-trained one on the Kitchen dataset except for the first convolution.

Late Fusion: late fusion is to simply average the prediction of models trained with different modalities.

Table 3 provides detailed comparison. Note that using optical flow actually hurt the performance of noun prediction, therefore the fusion results for noun classification are not shown.

In the end, we choose the late fusion strategy for our submission because it is more stable and has a larger performance gain in the seen validation set.

Top1 Acc. (%)	Seen Verb	Unseen Verb
RGB	47.71	40.82
Flow	49.7	47.87
Input Fusion	50.56	49.12
Late Fusion	53.68	47.68

Table 3. Verb top1 accuracy of InceptionV1 I3D using modality RGB, optical, and their fusion with different strategies.

5. Training Details

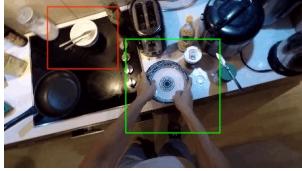
Data Sampling: In the training stage, the numbers of frames are empirically set to 6 and 12 for noun and verb models, respectively, in 6 fps. The starting frame of each clip is randomly sampled in a range such that the end frame does not exceed the action period.

Data Augmentation: A good data augmentation expands the space of training data to prevent severe overfitting. First, we resize the shorter part of the video to 256 pixels wide and apply the multi-scale cropping technique described in [?], which is composed of the corner cropping and scale-jittering techniques. Cropped frames will be resized to 244×244 before they are fed into our models.

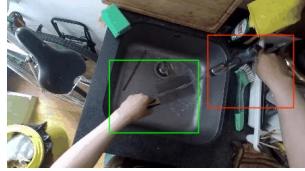
Hyper parameters: We use Amsgrad optimizer with learning rate 0.0003, weight decay 0, and decline the learning rate by 0.6 times per 5 epochs. Since we have model weights pre-trained on Kinetics except for the last classification later, we freeze all the weights but the last layer for the first 3 epochs to achieve a more steady training.

6. Inference

Sampling Scheme: we sample different crops of testing videos and use the average of prediction over the crops to reduce inference bias.



(a) Visually similar classes



(b) Wrong attention classes

Figure 2. Some examples of easily confused nouns.

For spatial sampling, we resize the shorter side of the videos to 256 pixels wide at the beginning. Then, three squares with width 256 are cropped in the position of the left side, right side, and center. On the part of temporal sampling, 4 clips with different starting frames are sampled, and their starting frames are distributed uniformly in the range such that the end frame does not exceed the action period.

Ensemble: for the final submission version, we independently split the other three validation sets following the same procedure in section 2.2 and train our models on all of them. The final prediction is fused using bagging.

7. Fail Cases Investigation

We study the easily confused classes and sort out some common properties about these failed instances.

7.1. Confusion of Noun

Visually similar classes: certain instances cannot be discriminated by RGB or flow visual features, but require depth information. For example, *plate* being falsely recognized as the class *bowl* in 2% of *plate* examples using our TSM model (see Fig. 2 (a)), which has the second highest frequency over all confused classes.

Action irrelevant objects: images with various objects need hand gesture features to avoid the action irrelevant object being chosen as the target instance. Fig. 2 (a) shows the action *taking the plate* and it is not related to *bowl*.

Wrong attention: some action may relate to more than one items as hands approach or pass by multiple items. In this situation, our model may attend to the wrong place. The classes *knife* and *tap* exemplify this (see Fig. 2 (b)), where the latter is misclassified as the former in near 2% of cases.

7.2. Confusion of Verb

Time reversed classes have similar visual content but are reversed from each other. One classic instance here is *put*, which is misclassified as *taking* by InceptionV1 I3D RGB model in 10% of *put* examples, while *taking* being falsely classified as *put* in 20% of *taking* examples.

Vague meaning classes states that action is easily misclassified as the common action, especially for classes with less training data. For example, the class *wash* is misclassified as *put* in 8% of *wash* examples.

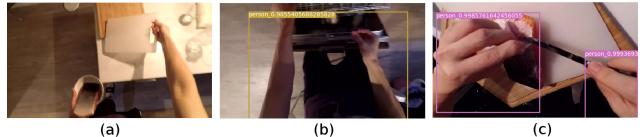


Figure 3. YoloV3 hand detection failed cases. (a) Fail to detect bounding boxes. (b) Merge two hands into one bounding box. (c) Expected hand bounding boxes.

8. Action Anticipation Track

Apart from action recognition track, we also apply the same approach in Epic-Kitchens Action Anticipation Challenge without modification and get 31.15%, 16.84%, and 5.72% top1 accuracies respectively in the verb, noun, and action prediction in the seen test set. Our temporal sampling range is set to $[t - 5, t - 1]$ in seconds, where t is the starting time of the target action.

9. Future Work

9.1. Hand Bounding Box and Pose

Since the Epic-Kitchens dataset is mostly composed of actions by hands, we expect that hand poses could be a crucial modality. We use human detection from YoloV3 [?] for hand detection and Openpose [?] for hand pose detection given the bounding boxes from YoloV3. We connect the key points following Openpose and make it an RGB image as an input modality as shown in Fig. 1. Due to the time limit, both models are used without fine-tuning, so the performance is not ideal (the model may fail to detect hands in some frames or include two hands in the same bounding box, see Fig 3). Improving the first-person view hand detection is a potential future work to increase the robustness for the hand pose modality.

9.2. Object Detection

Inspired by [?] that giving focus on the regions of interesting objects can improve object awareness for higher noun prediction, our recognition task could also benefit from objects detection results. Among several object detection methods including region proposal based ones such as Fast-RCNN [?] and Faster-RCNN [?], non-region proposal based like SSD [?], and end-to-end learning based like YOLOv3 [?], we choose YOLOv3 for the sake of the model complexity and the number of trainable parameters.

Although we did not have the detection model converge in the Epic-Kitchens dataset, we plan to employ the detection model with our recognition model by the following two approaches in the future: 1. Extract feature maps from the three output layers of YOLOv3 and rescale them such that it can be concatenated with the feature maps of the middle layers in our recognition models. 2. Place the fine-tuned YOLOv3 in front of our recognition model to ex-

tract bounding boxes as auxiliary input channels for object-wisely semantic information, and train the whole architecture in an end-to-end manner.

10. Conclusion

In this challenge, we exploit advantages and disadvantages among several neural network backbones and fusion strategies of multiple modalities. Our method gets competitive results in the action recognition and anticipation tracks. Although we do not have time to bring the hand, pose, and object box modalities into play, according to the analysis in section 7, there should be room for improvement when these modalities are properly utilized. Therefore, employing a more delicate fusion method could be a promising future work.

Recognizing Manipulation Actions from State-Transformations

Nachwa Abouabkr

Univ. Grenoble Alpes, Grenoble INP,
CNRS, Inria, LIG
nachwa.abouabkr@inria.fr

James L. Crowley

Univ. Grenoble Alpes, Grenoble INP,
CNRS, Inria, LIG
james.crowley@inria.fr

Remi Ronfard

Univ. Grenoble Alpes, Inria,
Grenoble INP, CNRS, LJK
remi.ronfard@inria.fr

Abstract

Manipulation actions transform objects from an initial state into a final state. In this paper, we report on the use of object state transitions as a mean for recognizing manipulation actions. Our method is inspired by the intuition that object states are visually more apparent than actions thus provide information that is complementary to spatio-temporal action recognition. We start by defining a state transition matrix that maps action verbs into a pre-state and a post-state. We extract keyframes at regular intervals from the video sequence and use these to recognize objects and object states. Change in object state are then used to predict action verbs. We report results on the EPIC kitchen action recognition challenge.

1. Introduction

Most current approaches to action recognition interpret a frame sequence as a spatio-temporal signal. However, extending a 2D convolutional network by adding a 3rd temporal dimension to the receptive field results in a substantial increase in the number of parameters that must be learned, greatly increasing the computational cost and the requirements for training data. An alternative approach is to decompose recognition into a static recognition phase using a 2D kernel followed by either a 1D temporal kernel [19], or a Recurrent Neural network [7]. Researchers have also explored the use of two-stream networks in which one stream is used to analyze image appearance from RGB images and the other represents motion from optical flow maps [18, 14, 11]. such approaches provide spatio-temporal analysis while avoiding the very large increase in learned parameters.

An alternative to learning spatio-temporal models for action recognition from video is to recognize changes in properties of objects from a sequence of frames [13, 3]. Baradel et al. [3] proposed a convolutional model that is trained to predict both object classes and action classes in two branches. This model is followed by an object relation

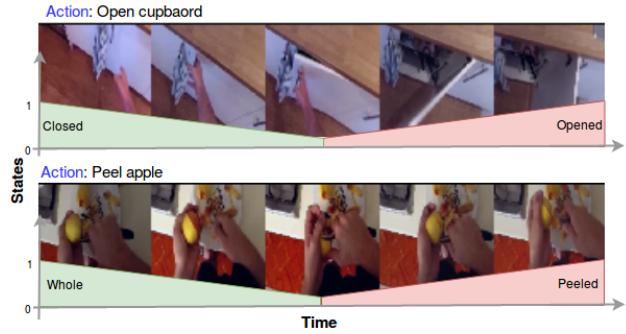


Figure 1. Changes in object states over time for action recognition. Two sample sequences from the EPIC kitchen dataset.

network that learns to reason over object interactions.

Our approach is inspired by the human ability to recognize changes in situation using a limited number of static observations. Human associate observations with background knowledge in a form of previously seen episodes or past experience [9, 4]. Thus a change in an object's state allows a human to form hypotheses about how the object was changed. This ability allows a human subject to interpret a complex scene from static images and make hypotheses about unseen actions that may have occurred and could explain changes to the scene. For example, we can understand which action is shown in Figure 1 with 5 keyframes or less from the video clip. Inferring the associated actions in frame sequences is a relatively effortless task for a human, while it remains challenging for machines [16]. We have investigated whether such an approach can be used to infer unseen actions from a set of frames which are chronologically ordered and contains semantic relations between objects. Such inference would complement hypotheses from spatio-temporal action recognition.

A manipulation action transforms an object from a pre-existing state (pre-state) into a new state (post-state). Thus we can say that the action *causes* a change in the state of the corresponding object. Alayrac et al. [2] have investigated automatic discovery of both object states and actions from videos. They treat this problem as a discriminative cluster-

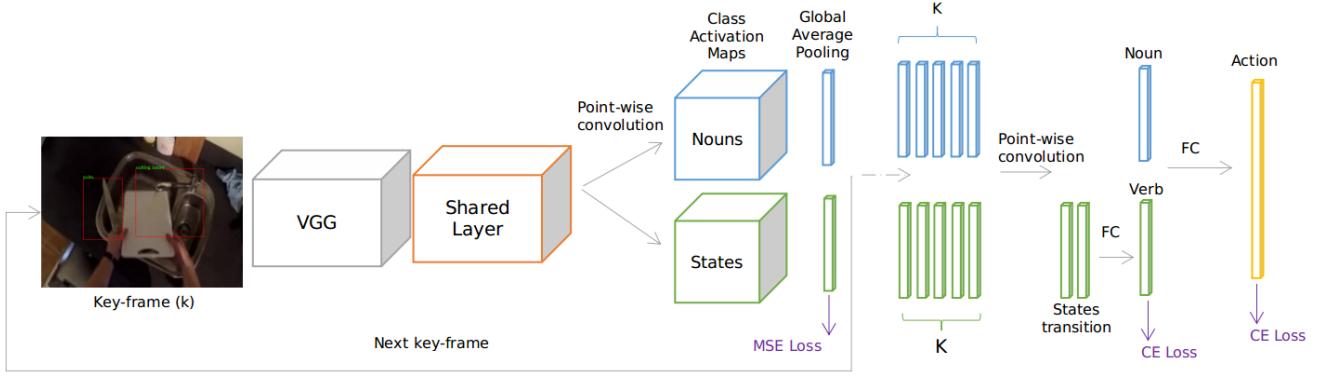


Figure 2. Proposed architecture of learning action recognition as state transformations.

ing problem by exploiting the ordering of the frames. Their work is promising, even though it has been evaluated on only a small number of action classes. A related work [8], studies visual changes of objects state between first and last frames.

In this paper, we investigate the feasibility of recognizing object types and object states from a small number of frames and then use changes in object states to predict actions. Our intuition is that 2D object types and states are easier to recognize than spatio-temporal action verb.

2. Manipulation action as state transformation

An action, as defined in the Cambridge dictionary¹, is the effect something has on another thing. Many manipulation actions can be expressed as triple in which a subject imparts a change to an object. That is, a manipulation action $a_i \in A$ can be expressed as: the subject that performs the action, the verb $v_i \in V$ which describes the effect of the action, and the object $n_i \in N$ the effect is applied to. For egocentric data such as EPIC kitchen the subject assumed to be the person.

The action recognition problem can be formulated with one class for each possible combination of these elements. For example, *person cuts tomato* and *person cuts cucumber* can be considered as two different classes as in [17]. Some recent datasets have provided a decomposition of an action into a verb and one or more objects $a = (v, (n_1, \dots, n_n))$ [10, 5, 12]. This makes it possible to study the task of action recognition as a composition of several sub-tasks (e.g. object detection and action verb recognition).

2.1. State-changing actions

We are concerned with recognizing manipulation actions that change the state of objects $s_i \in S$. The state change can

appear in the object’s shape, its appearance (color), or its location. Examples of object states include: closed, opened, full, empty, whole, and cut.

We define a state transition function F that transforms the corresponding object from a pre-state s_i into a post-state s_j . In some cases, this state transition can be defined directly from the type of action verb v_i . We observe that sometimes a single verb is not enough to distinguish an action. For example, the verb *remove* can mean open in *remove lid* and can mean peel in *remove the skin of the garlic*. Therefore, the state transition must take into account both action verbs and nouns.

Since the state changes happen as we move through time, the transition function F returns a real value of each state depending on the frame position in the video sequence. As in Figure 1 the object starts in its initial state that gradually fades out and the post-state starts to appear as we advance in the video. In our initial experiments we have assumed that the state changing frame is the mid-frame of the video sequence. Therefore, we define the action transition mapping function $F(v, n)$, which takes the action’s verb v and a set of objects (nouns) n and returns a continuous value of objects’ states for each frame depending on the frame position in the video. For example, the action *open fridge* changes the fridge state from opened to closed.

2.2. Architecture

In previous work [1], we investigated detection and location of object types as well as object states from images. In this paper, we extend this work to learn changes in object state from keyframes. The architecture of our model is shown in Figure 2. Given a video segment, we first split it into k sub-segments of equal length and sample a random keyframe from each sub-segment. For each keyframe, we learn two concept classes (object types and object states) separately. Then, from the selected sequence

¹Cambridge University Press. (2019). Cambridge online dictionary, Cambridge Dictionary online. Retrieved at April 3, 2019

	Seen kitchens subset (S1)				Unseen kitchens subset (S2)			
	Acc T1	Acc T5	Precision	Recall	Acc T1	Acc T5	Precision	Recall
Action								
Our model(RGB)	19.76	36.98	9.83	10.23	9.08	19.46	3.68	4.77
2SCNN[14](RGB)	13.67	33.25	6.66	5.47	6.79	20.42	3.39	3.01
TSN[18](RGB)	19.86	41.89	9.96	8.81	10.11	25.33	4.77	5.67
Verb								
Our model(RGB)	47.41	81.33	31.20	20.43	34.35	69.24	15.09	11.00
2SCNN[14](RGB)	40.44	83.04	33.74	15.9	33.12	73.23	16.06	9.44
TSN[18](RGB)	45.68	85.56	61.64	23.81	34.89	74.56	19.48	11.22
Noun								
Our model(RGB)	28.31	53.77	21.21	22.48	17.48	37.56	10.71	12.55
2SCNN[14](RGB)	30.46	57.05	28.23	23.23	17.58	40.46	11.97	12.53
TSN[18](RGB)	36.8	64.19	34.32	31.62	21.82	45.34	14.67	17.24

Table 1. Results on the EPIC kitchen dataset (Seen and Unseen subsets). Highest values are in bold. Results of baseline methods (2SCNN and TSN) are reported by [5].

of k keyframes, we extract two channels using a point-wise convolution from which we construct the state transition matrix (pre-state, post-state). For object types (nouns), we use a point-wise convolution to extract a vector of nouns that appear in the video segment. Action verbs are then learned from the state transition matrix. In the end, the action classes are learned directly from the set of object types and action verbs.

3. Experiment

EPIC Kitchen dataset. We have investigated state transformations using action labels using the egocentric videos of people cooking and cleaning in the EPIC Kitchen dataset. In this dataset, an action label is composed of a tuple of $a_i = (\text{verb } v_i, \text{ noun } n_i)$ extracted from a narrated text given for each video action segment.

The EPIC verb represents the action verb while the EPIC noun is the action object. As the EPIC Kitchen dataset is an egocentric dataset which suggests one subject in the scene, the action subject is always the cook’s hands. We group each action verb depending on the type of effect they cause into 3 different groups: those that change the object’s shape, color appearance, or location. This study leaves some non-state-changing verbs (like the verb *check*) out of those groups as it does not change any object states. As a result we define 49 state transitions and 31 different states.

Network Architecture. As shown in Figure 2, we use a similar setting as in [1] for each keyframe. We start by extracting features using a VGG16 network with batch normalization [15] pre-trained on the ImageNet dataset [6].

VGG features provide the input to a shared² 3×3 convolutional layer. We separate the learning of object attributes into two branches: one for object types and the other for object states. Each attribute is learned with an independent loss. VGG features are frozen during the training process for object types and states.

For each keyframe, one noun vector and one state vector are extracted using Global Average Pooling over corresponding Class Activation Maps. Afterwards, we perform a point-wise convolution to extract one noun vector and the states transition matrix over keyframes. Verbs are learned directly from the state transition matrix using a fully-connected (FC) layer. Both action attributes (verb, nouns) are fused using at a late stage a FC layer for action classification. All hidden layers use the ReLU (rectified linear unit) activation function. A frame can have one or more states and/or nouns. Therefore, we treat nouns and states as multi-label classification problems that are learned with a Mean Square Error (MSE). On the other hand, verbs and actions are learned with a Cross Entropy (CE) function.

Training. We use EPIC Kitchen video segments for training our model. A clip is a collection of k randomly sampled keyframes from k equal length sub-segments, and it represents the corresponding action video segment. This strategy has been used in multiple works with similar problems [18, 3]. We divide the EPIC videos in 80% for training and 20% for validation. Our validation set has only samples from many-shot actions and all samples of few-shot actions are in the our training split.

²shared over both attributes (object types and states)

	<i>take</i>	<i>put</i>	<i>open</i>	<i>close</i>	<i>wash</i>	<i>cut</i>	<i>mix</i>	<i>pour</i>	<i>peel</i>	Avg
Precision (%)	56.7	59.3	58.8	39.8	80.1	74.7	68.9	39.1	37.7	57.23
Recall (%)	48.2	45.0	62.9	57.1	67.7	60.7	50.2	40.3	53.5	53.96

Table 2. Model performance on validation set on state-changing verbs.

EPIC challenge evaluation. For evaluation, we aggregate the results of 10 clips as in [3]. We report the same evaluation metrics provided by the EPIC challenge [5]. Provided metrics include class-agnostic and class-aware metrics; Top-1 and Top-5 micro-accuracy in addition to precision and recall over only many shot classes (i.e. classes with more than 100 samples).

Implementation details. For learning, we used MSE loss to learn nouns and states during per-frame learning. Object nouns in the Actions of EPIC dataset are used to define our object classes. Each action of EPIC dataset is a tuple of a verb and a noun. The noun is chosen to be the first noun occurring in the narration sentence. Because sentences and frames can contain multiple objects, we train to detect all nouns in the sentence and treat this training step as a multi-label recognition problem for each frame. Because object state changes gradually, the state is represented as a continuous number estimated using MSE.

In training, we used the Adam optimizer with a learning rate of $1e-3$ that decreases following the Reduce on Plateau scheduling method. The implementation code is available³ and was written using Pytorch.

4. Discussion

Comparison with baselines. We report the results of our model in Table 1 on EPIC Kitchen dataset for action recognition task. As the test sets are not publicly available yet, we compared our results to two baseline techniques, 2SCNN model [14] and TSN model [18], as reported in [5].

In our model, we only use RGB channels. Our model has 20M parameters and only 5M trainable parameters which is significantly lower than both baseline techniques i.e. for each input modality: 2SCNN model [14] uses 170M trainable parameters and TSN model [18] has 11M trainable parameters. Our model outperforms 2SCNN model [14] in most of reported metrics and provides recognition of verbs and actions that is comparable to TSN reported results[18].

State-changing Actions. In order to evaluate our model on state-changing actions, we report results of our validation set in Table 2. The model is trained to learn state

³Code is available at https://github.com/Nachwa/object_states

changes and shows better performance on state-changing verbs than on verbs that are not state changes.

Our results show some confusion between semantically similar verbs like (e.g. insert and put, or put and move) and verbs that have visually similar states (e.g. wash and fill - where fill examples refers to filling water from a tap). Our model is not designed to detect actions that do not result in a change in object state (e.g. move and walk).

5. Conclusion

In this paper, we investigated a method for recognition of manipulation actions as changes of state of objects in keyframes. We demonstrate that this can provide reasonably accurate recognition of manipulation actions. We reported results of our model on the challenge of EPIC kitchen dataset and compare these to two baseline techniques. For the action recognition task, our model outperforms one of the baseline techniques using 34 times less training parameters, and achieved comparable results with the other.

References

- [1] Nachwa Aboubakr, Remi Ronfard, and James Crowley. Recognition and localization of food in cooking videos. In *ACM International Conference Proceeding Series*, 2018. 2, 3
- [2] Jean-Baptiste Alayrac, Ivan Laptev, Josef Sivic, and Simon Lacoste-Julien. Joint discovery of object states and manipulation actions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2127–2136, 2017. 1
- [3] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. Object level visual reasoning in videos. In *ECCV*, 2018. 1, 3, 4
- [4] Samy Blusseau, A Carboni, A Maiche, Jean-Michel Morel, and R Grompone von Gioi. A psychophysical evaluation of the a contrario detection theory. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 1091–1095. IEEE, 2014. 1
- [5] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018. 2, 3, 4
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image

- database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3
- [7] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015. 1
- [8] Alireza Fathi and James M Rehg. Modeling actions through state changes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2586, 2013. 2
- [9] François Fleuret, Ting Li, Charles Dubout, Emma K Wampler, Steven Yantis, and Donald Geman. Comparing machines and humans on a visual categorization test. *Proceedings of the National Academy of Sciences*, 108(43):17621–17625, 2011. 1
- [10] Xiaofeng Gao, Ran Gong, Tianmin Shu, Xu Xie, Shu Wang, and Song-Chun Zhu. Vrkitchen: an interactive 3d virtual environment for task-oriented learning. *arXiv preprint arXiv:1903.05757*, 2019. 2
- [11] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid. Joint learning of object and action detectors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4163–4172, 2017. 1
- [12] Yin Li, Miao Liu, and James M Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 619–635, 2018. 2
- [13] Chih-Yao Ma, Asim Kadav, Iain Melvin, Zsolt Kira, Ghassan AlRegib, and Hans Peter Graf. Attend and interact: Higher-order object interactions for video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6790–6800, 2018. 1
- [14] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 1, 3, 4
- [15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICRL)*, 2015. 3
- [16] Sebastian Stabinger, Antonio Rodríguez-Sánchez, and Justus Piater. 25 years of cnns: Can we compare to human abstraction capabilities? In *International Conference on Artificial Neural Networks*, pages 380–387. Springer, 2016. 1
- [17] Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 729–738. ACM, 2013. 2
- [18] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016. 1, 3, 4
- [19] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatio-temporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018. 1

EPIC-Kitchens Egocentric Action Anticipation and Recognition Challenges 2019

Team DMI-UNICT Technical Report

Antonino Furnari
University of Catania
furnari@dmi.unict.it

Giovanni Maria Farinella
University of Catania
gfarinella@dmi.unict.it

Abstract

This technical report describes Team DMI-UNICT approach to the EPIC-Kitchens Egocentric Action Anticipation and Egocentric Action Recognition Challenges 2019. We developed Rolling-Unrolling LSTMs (RU-LSTMs), an architecture able to anticipate actions at multiple temporal scales using two LSTMs to 1) summarize the past, and 2) formulate predictions about the future. The input video is processed considering three complimentary modalities: appearance (RGB), motion (optical flow) and objects (object-based features). Modality-specific predictions are fused using a novel Modality ATTention (MATT) mechanism which learns to weigh modalities in an adaptive fashion. The proposed approach can anticipate actions at multiple anticipation times τ_a ranging from 0.25s to 2s. For the challenge, we consider the anticipations obtained at $\tau_a = 1$ s. We also adapted the proposed approach to tackle the egocentric action recognition task, reporting results for the EPIC-Kitchens egocentric action recognition challenge as well. Code and models will be released at the project page: <http://iplab.dmi.unict.it/rulstm>.

1. Introduction

We observe that anticipation methods need to perform two sub-tasks: 1) summarizing what has happened in the past, up to the point of prediction, and 2) making hypotheses on what will happen next, depending on what has already been observed. While action anticipation methods generally attempt to perform these tasks jointly [2, 4, 11], we propose to disentangle the two responsibilities by using two separate LSTMs. The first LSTM (“Rolling” LSTM) continuously summarizes the semantic content of the streaming video. The second LSTM (“Unrolling” LSTM) anticipates future actions conditioned on the outputs of the rolling LSTM. The proposed learning architecture is pre-trained using a novel “sequence completion” pre-training technique, which encourages the disentanglement of the two responsibilities.

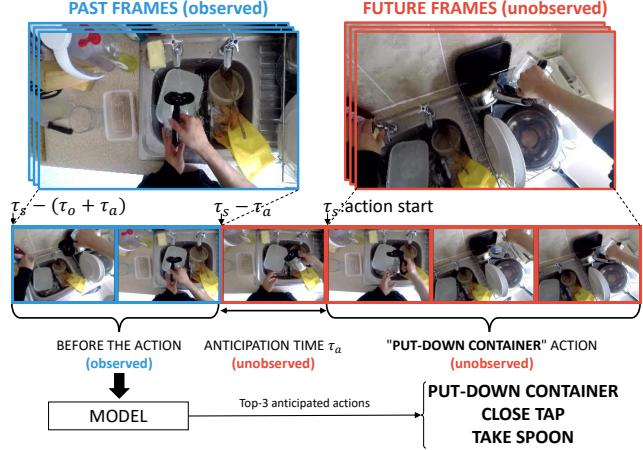


Figure 1. Egocentric action anticipation as defined in [1].

To leverage multimodal information, our model includes an RGB, an optical flow and an object branch which process the video independently. The predictions made by the three branches are fused through a new “modality attention” mechanism which adaptively chooses how to combine predictions related to the different modalities based on the observed sample.

2. Proposed Approach

The proposed Rolling-Unrolling LSTM architecture used for the challenge is described in detail in [3]. The reader is referred to [3] for further details and discussion.

Processing Strategy As defined in [1] and illustrated in Figure 1, egocentric action anticipation is defined as the task of anticipating an action (e.g., “put-down container”) by observing a video segment of length τ_o preceding the action by τ_a seconds. Here τ_a is referred to as the “anticipation time”, τ_o is the “observation time”, and the starting time of the action is denoted by τ_s . Our approach generalizes this definition by enabling predictions at multiple anticipation times. Figure 2 illustrates the processing strategy adopted by the proposed method. The video is processed

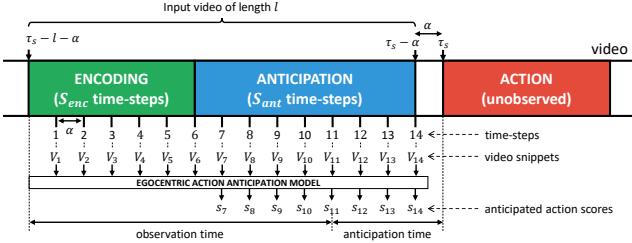


Figure 2. Video processing scheme of the proposed method with $S_{enc} = 6$ and $S_{ant} = 8$.

in an on-line fashion, with a short video snippet V_t consumed every α seconds, where t indexes the current time-step. Specifically, an action occurring at time τ_s is anticipated by processing a video segment of length l starting at time $\tau_s - l - \alpha$ and ending at time $\tau_s - \alpha$. The input video ends at time $\tau_s - \alpha$ as our method aims at anticipating actions *at least* α seconds before they occur. The processing is performed in two stages: an “encoding” stage, which is carried out for S_{enc} time-steps, and an “anticipation” stage, which is carried out for S_{ant} time-steps. In the encoding stage, the model summarizes the semantic content of the S_{enc} input video snippets without producing any prediction, whereas in the anticipation stage the model continues to encode the semantics of the S_{ant} input video snippets and outputs S_{ant} action scores s_t which can be used to perform action anticipation. This scheme effectively allows to formulate S_{ant} predictions for a single action at multiple anticipation times. In our experiments, we set $\alpha = 0.25s$, $S_{enc} = 6$ and $S_{ant} = 8$. In these settings, the model analyzes video segments of length $l = \alpha(S_{enc} + S_{ant}) = 3.5s$ and outputs 8 predictions at the following anticipation times: $\tau_a \in \{2s, 1.75s, 1.5s, 1.25s, 1s, 0.75s, 0.5s, 0.25s\}$. For the challenge, we report the anticipations predicted at time-step $t = 11$. This way, our model anticipates actions with an effective observation time equal to $\tau_o = \alpha \cdot t = 2.75s$ and an anticipation time equal to $\tau_a = \alpha(S_{ant} + S_{enc} + 1 - t) = 1s$.

Rolling-Unrolling LSTM The proposed method uses two separate LSTMs to encode past observations and formulate predictions about the future. Following previous literature [10], we include multiple identical branches which analyze the video according to different modalities. Specifically, at each time-step t , the input video snippet V_t is represented using different modality-specific representation functions $\varphi_1, \dots, \varphi_M$ depending on learnable parameters $\theta^{\varphi_1}, \dots, \theta^{\varphi_M}$. This process allows to obtain the modality-specific feature vectors $f_{1,t} = \varphi_1(V_t), \dots, f_{M,t} = \varphi_M(V_t)$, where M is the total number of modalities (i.e., the total number of branches in our architecture), and $f_{m,t}$ is the feature vector computed at time-step t for the modality m . The feature vector $f_{m,t}$ is fed to the m^{th} branch of the architecture. While our model can easily incorpo-

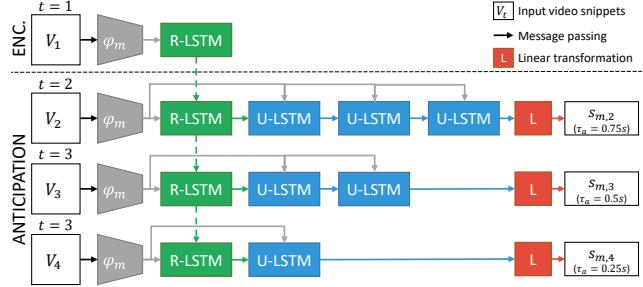


Figure 3. Example of RU modality-specific branch with $S_{enc} = 1$ and $S_{ant} = 3$.

rate different modalities, in this work we consider $M = 3$ modalities, i.e., RGB frames (spatial branch), optical flow (motion branch) and object-based features (object branch).

Figure 3 illustrates the processing taking place in a single branch m of the proposed RU-LSTM model. For illustration purposes only, the figure shows an example in which $S_{enc} = 1$ and $S_{ant} = 3$. At time step t , the feature vector $f_{m,t}$ is fed to the Rolling LSTM (R-LSTM), which encodes its semantic content recursively, as follows:

$$(h_{m,t}^R, c_{m,t}^R) = LSTM_{\theta_m^R}(f_{m,t}, h_{m,t-1}^R, c_{m,t-1}^R) \quad (1)$$

where $LSTM_{\theta_m^R}$ denotes the R-LSTM of branch m , depending on the learnable parameters θ_m^R , whereas $h_{m,t}^R$ and $c_{m,t}^R$ are the hidden and cell states computed at time t in the modality m . The initial hidden and cell states of the R-LSTM are initialized with zeros: $h_{m,0}^R = \mathbf{0}$, $c_{m,0}^R = \mathbf{0}$.

In the anticipation stage, at time step t , the Unrolling LSTM (U-LSTM) is used to make predictions about the future. The U-LSTM takes over the hidden and cell vectors of the R-LSTM at the current time-step (i.e., $h_{m,t}^R$ and $c_{m,t}^R$) and iterates over the representation of the current video snippet $f_{m,t}$ for a number of times n_t equal to the number of time-steps required to reach the beginning of the action, i.e., $n_t = S_{ant} + S_{enc} - t + 1$. Hidden and cell states of the U-LSTM are computed as follows at the j^{th} iteration:

$$(h_{m,j}^U, c_{m,j}^U) = LSTM_{\theta_m^U}(f_{m,t}, h_{m,j-1}^U, c_{m,j-1}^U) \quad (2)$$

where $LSTM_{\theta_m^U}$ is the U-LSTM of branch m , depending on the learnable parameters θ_m^U , and $h_{m,t}^U$, $c_{m,t}^U$ are the hidden and cell states computed at iteration j for the modality m . The initial hidden and cell states of the U-LSTM are initialized from the current hidden and cell states computed by the R-LSTM: $h_{m,0}^U = h_{m,0}^R$, $c_{m,0}^U = c_{m,0}^R$. Note that the input $f_{m,t}$ of the U-LSTM does not depend on j (see eq. (2)) because it is fixed during the “unrolling” procedure. The main rationale of “unrolling” the U-LSTM for a different number of times at each time-step is to encourage it to differentiate predictions at different anticipation times.

Modality-specific action scores $s_{m,t}$ are computed at time-step t by processing the last hidden vector of the U-

LSTM with a linear transformation with learnable parameters θ_m^W and θ_m^b : $s_{m,t} = \theta_m^W h_{m,n_t}^U + \theta_m^b$.

Sequence Completion Pre-training (SCP) The two LSTMs composing the RU architecture are designed to address two specific sub-tasks: the R-LSTM is responsible for encoding past observations and summarizing what has happened up to a given time-step, whereas the U-LSTM focuses on anticipating future actions conditioned on the hidden and cell vectors of the R-LSTM. To encourage the two LSTMs to specialize on the two different sub-tasks, we propose to train the architecture using a novel Sequence Completion Pre-training (SCP) procedure. During SCP, the connections of the U-LSTM are modified to allow it to process future representations, rather than iterating on the current one. In practice, the U-LSTM hidden and cell states are computed as follows during SCP:

$$(h_{m,j}^U, c_{m,j}^U) = LSTM_{\theta_m^U}(f_{m,t+j-1}, h_{m,j-1}^U, c_{m,j-1}^U) \quad (3)$$

where the input representations $f_{m,t+j-1}$ are sampled from future time-steps $t+j-1$. The main goal of pre-training the RU with SCP is to allow the R-LSTM to focus on summarizing past representations without trying to anticipate the future.

Modality ATTention (MATT) Coherently with past work on egocentric action anticipation [1], we found it sub-optimal to fuse multi-modal predictions with classic approaches such as early and late fusion. This is probably due to the fact that, when anticipating egocentric actions, one modality might be more useful than another (e.g., appearance over motion), depending on the processed sample. We introduce a Modality ATTention (MATT) module which computes a set of attention scores indicating the relative importance of each modality for the final prediction. At a given time-step t , such scores are obtained by processing the concatenation of the hidden and cell vectors of the R-LSTM networks belonging to all branches $m = 1, \dots, M$ with a deep neural network D depending on the learnable parameters θ^{MATT} :

$$\lambda_t = D_{\theta^{MATT}}(\oplus_{m=1}^M (h_{m,t}^R \oplus c_{m,t}^R)) \quad (4)$$

where \oplus denotes the concatenation operator and $\oplus_{m=1}^M (h_{m,t}^R \oplus c_{m,t}^R)$ is the concatenation of the hidden and cell vectors produced by the R-LSTM at time-step t across all modalities. Late fusion weights can be obtained normalizing the score vector λ_t with the softmax function in order to make sure that fusion weights sum to one: $w_{m,t} = \frac{\exp(\lambda_{t,m})}{\sum_k \exp(\lambda_{t,k})}$, where $\lambda_{t,m}$ is the m^{th} component of the score vector λ_t . The final set of fusion weights is obtained at time-step t by merging the modality-specific predictions produced by the different branches with a linear combination as follows: $s_t = \sum_m w_{m,t} \cdot s_{m,t}$. Figure 4

illustrates an example of a complete RU with two modalities and the MATT fusion mechanism. For illustration purposes, the figure shows only three anticipation steps.

3. Implementation and Training Details

Branches and Representation Functions We instantiate the proposed architecture with 3 branches: a spatial branch which processes RGB frames, a motion branch which processes optical flow, and an object branch which processes object-based features. Our architecture analyzes video snippets of 5 frames $V_t = \{I_{t,1}, I_{t,2}, \dots, I_{t,5}\}$, where $I_{t,i}$ is the i^{th} frame of the video snippet. The representation function φ_1 of the spatial branch computes the feature vector $f_{1,t}$ by extracting features from the last frame $I_{t,5}$ of the video snippet using a Batch Normalized Inception CNN [8] trained for action recognition. The representation function φ_2 of the motion branch extracts optical flow from the 5 frames of the current video snippet as proposed in [12]. The optical flow is fed to a Batch Normalized Inception CNN trained for action recognition to obtain the feature vector $f_{2,t}$. Note that φ_1 and φ_2 allow to obtain “action-centric” representations of the input frame which can be used by the R-LSTM to summarize what has happened in the past. The representation function φ_3 of the object branch extracts objects from the last frame $I_{t,5}$ of the input snippet V_t using an object detector. A fixed-length representation $f_{3,t}$ is obtained by accumulating the confidence scores of all bounding boxes predicted for each object class. Specifically, let $b_{t,i}$ be the i^{th} bounding box detected in image $I_{t,5}$, let $b_{t,i}^c$ be its class and let $b_{t,i}^s$ be its detection confidence score. The j^{th} component of the output representation vector $f_{3,t}$ is obtained by summing the confidence scores of all detected objects of class j , i.e., $f_{3,t,j} = \sum_i [b_{t,i}^c = j] b_{t,i}^s$, where $[\cdot]$ denotes the Iverson bracket. This representation only encodes the presence of an object in the scene, discarding its position in the frame. We found this holistic representation to be sufficient in the case of egocentric action anticipation. Differently from φ_1 and φ_2 , φ_3 produces object-centric features, which carry information on what objects are present in the scene and hence could be interacted next.

Architectural Details of RU-LSTM and MATT We use a Batch Normalized Inception CNN [8] (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 [13]. Specifically, we used the pre-computed optical flows provided along with EPIC-KITCHENS (see <http://epic-kitchens.github.io/>). 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 fea-

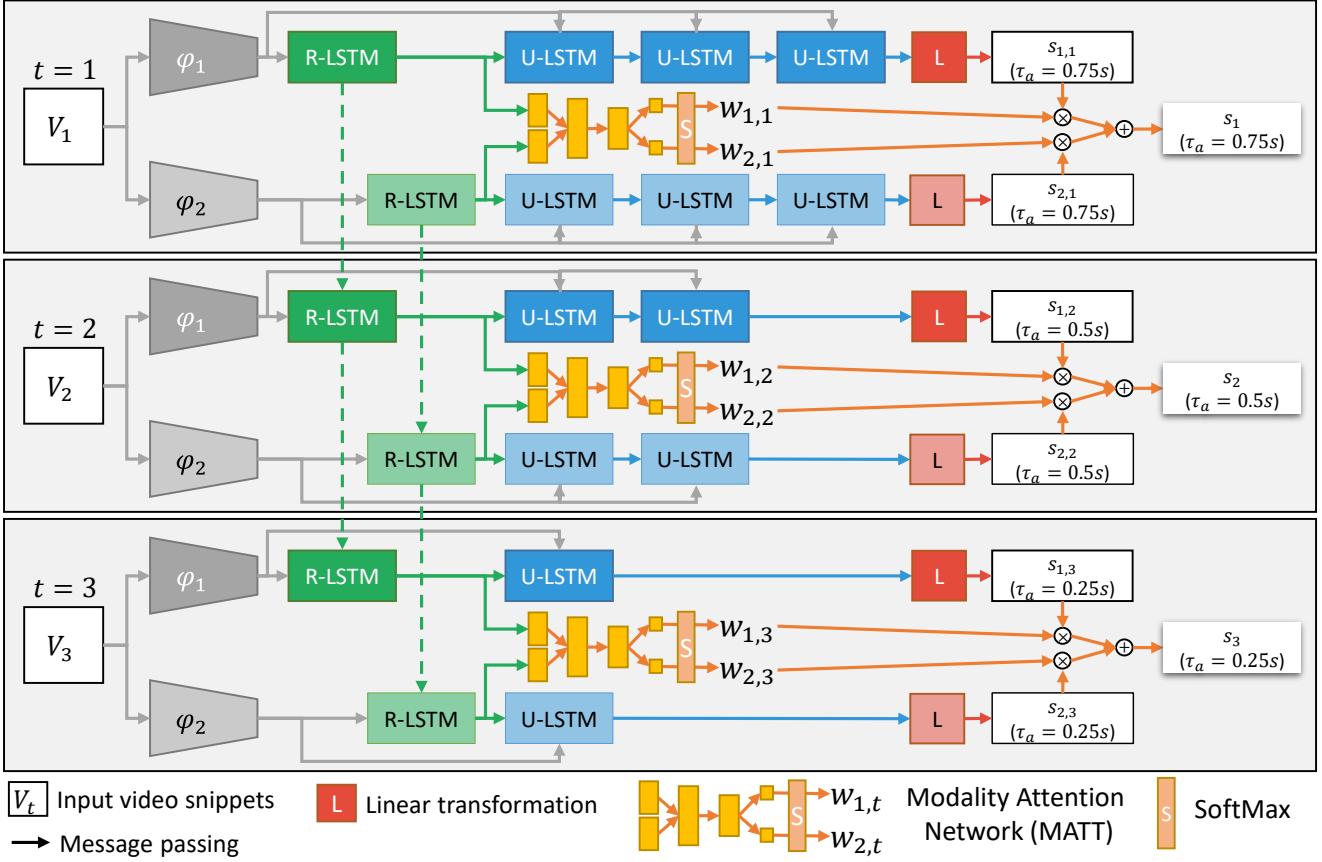


Figure 4. Example of the complete RU architecture with two modalities and the Modality ATTention mechanism (MATT).

ture vector regardless the 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 [5] with a ResNet-101 backbone [7], as implemented in [6]. Both the Rolling LSTM (R-LSTM) and the Un-rolling 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. The ReLU activation function are used within the attention network.

Architectural Details 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 [12]. Specifically, we set the number of segments to 3 and train the TSN models with Stochastic Gradient Descent (SGD) using standard 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 to train each CNN on a single Titan X. For all other parameters, we use the values recommended in [12]. We train the object detector to recognize the 352 object classes of the EPIC-KITCHENS dataset. The object detector has been trained on two Tesla V100 for 504531 iterations, with an initial learning rate of 0.005, which has been decreased by a factor of 10 after 336354 and 454077 steps. More details are available at our project web page (<http://iplab.dmi.unict.it/rulstm>). This training procedure allows to

	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	Single	33.04	22.78	14.39	79.55	50.95	33.73	25.50	24.12	07.37	15.73	19.81	07.66
	Ensemble	31.13	22.93	15.25	78.03	51.05	35.13	22.58	24.26	08.41	17.71	20.05	08.05
S2	Single	27.01	15.19	08.16	69.55	34.38	21.10	13.69	09.87	03.64	09.21	11.97	04.83
	Ensemble	26.63	15.47	09.12	68.11	35.27	21.88	16.58	09.93	03.16	11.08	11.70	04.55

Table 1. EPIC-Kitchens egocentric action anticipation results.

	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	Single	56.93	45.03	33.06	85.68	67.12	55.32	50.42	39.84	18.91	38.82	38.11	19.12
	Ensemble	58.99	45.00	35.14	86.70	69.08	57.62	52.23	40.06	19.40	42.12	39.32	20.28
S2	Single	43.67	26.77	19.49	73.30	48.28	37.15	23.40	20.82	09.72	18.41	21.59	13.33
	Ensemble	47.35	28.64	21.37	73.75	51.01	39.47	26.88	22.09	10.53	22.12	23.31	13.98

Table 2. EPIC-Kitchens egocentric action recognition results.

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 with MATT and the model is further fine-tuned for 100 epochs using cross entropy and the same learning parameters.

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 *action recognition*, we choose the epoch obtaining the best average Top-1 action accuracy across observation rates. The proposed RU-LSTM architecture has been implemented using the PyTorch library [9]. The code is available at <http://iplab.dmi.unict.it/rulstm>.

Ensemble The egocentric action anticipation scores submitted for the challenge have been obtained by an ensemble of two methods using respectively a BNInception [8] and a ResNet-101 [7] backbone. The two models have been trained independently and the results have been fused by late fusion.

4. Adaptation for the Task of Egocentric Action Recognition

We tested our approach on the task of egocentric action recognition as well. To this aim, we adapted our model to process 8 frames, evenly sampled from the input video. The network is trained to recognize the action as soon as possible applying cross entropy to the predictions obtained in the 8 time-steps. For the egocentric action recognition challenge, we consider the output obtained at the 8th time-step, i.e. when 100% of the video has been observed. We train four branches: a spatial branch processing RGB frames, a motion branch processing optical flow, an object branch processing object-centric representation, and an audio branch processing spectrograms. Branch-specific predictions are fused by late fusion (i.e., MATT is not used in this case).

Audio Processing The audio branch processes spectrograms extracted from the audio track of the input video. Specifically, we computed spectrograms over temporal windows of 1 second. Spectrograms are extracted at 30 *fps*, which allows to obtain overlap within the audio content of each temporal window. To compute spectrograms, we use the *librosa.feature.melspectrogram* function from the *librosa* library (<https://librosa.github.io/librosa/generated/librosa.feature.melspectrogram.html>), specifying 128 Mel bands and $\frac{s_r}{128}$ samples between successive frames (“hop_length” parameter), where s_r is the sampling rate of the audio track. This allows to obtain spectrograms of size 128×128 , which are encoded as grayscale images. The spectrograms are processed by a BNInception network truncated to the penultimate inception block, which is trained for egocentric action recognition with TSN.

Ensemble For the challenge, we trained an ensemble of models, including a spatial branch based on BNInception, a

spatial branch based on ResNet-101, a motion branch based on BNInception, a motion branch based on ResNet-101, an audio branch based on BNInception and the object branch.

5. Results

Table 1 reports the results of the proposed approach on the EPIC-Kitchens egocentric action anticipation challenge. While we participate in the challenge using our Ensemble model, the performances of the single model using a BNInception backbone are reported for reference.

Table 2 reports the results of the proposed approach on the EPIC-Kitchens egocentric action recognition challenge. Similarly to the anticipation results, the proposed ensemble model is compared with respect to a single model based on a BNInception backbone using a spatial branch, a motion branch and an object branch.

References

- [1] 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. [1](#), [3](#)
- [2] 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. [1](#)
- [3] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. *International Conference on Computer Vision*, 2019. [1](#)
- [4] Y. Gao, Z. Yang, and R. Nevatia. RED: Reinforced encoder-decoder networks for action anticipation. *British Machine Vision Conference*, 2017. [1](#)
- [5] Ross Girshick. Fast R-CNN. In *International Conference on Computer Vision*, pages 1440–1448, 2015. [4](#)
- [6] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. <https://github.com/facebookresearch/detectron>, 2018. [4](#)
- [7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, pages 770–778, 2016. [4](#), [5](#)
- [8] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on International Conference on Machine Learning*, pages 448–456, 2015. [3](#), [5](#)
- [9] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. [5](#)
- [10] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. [2](#)
- [11] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating visual representations from unlabeled video. In *Computer Vision and Pattern Recognition*, pages 98–106, 2016. [1](#)
- [12] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36, 2016. [3](#), [4](#)
- [13] 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. [3](#)

RML-Ryerson University Submission to EPIC-Kitchen Action Anticipation Challenge (ActivityNet 2019)

Nour Eldin Elmadany, Yifeng He, Ling Guan

Ryerson Multimedia Laboratory

Department of Electrical and Computer Engineering, Ryerson University, Toronto.

noureldin.elmadany@ryerson.ca

Abstract

In this paper, we introduce our submission for task c ego-centric activity understanding (EPIC-KITCHENS) Action Anticipation for Activitynet Challenge 2019. [3]. The proposed model is based on Multi-Fiber Network [2] and non-local neural network [8]. The presented model is based only on the RGB cue and the final submission achieves 13.20 % for Seen Kitchens (S1) and 8.5 % for Unseen Kitchens (S2).

1. Introduction

Activity Recognition in videos drawn researchers' attention in recent years. The research community introduced several third-person benchmark datasets including ActivityNet [4] and Kinetics [1]. Other researchers focused on first-person action recognition. However, the size of the first-person datasets are relatively smaller than the third-person datasets. Recently, EPIC-Kitchens dataset was introduced as a large scale first-person action recognition dataset. This dataset features the interactions between a first-person and kitchen-wares and appliances. It shows the multi-task learning including washing a few dishes. To recognize actions, verbs, and objects in videos, recent techniques were developed based on Deep ConvNets and raised the bar of state-of-the-art results. To address the challenge, we presented the Non-Local Multi-Fiber Network (NL-MFN) achieving 13.20 % for Seen Kitchens (S1) and 8.5 % for Unseen Kitchens (S2).

2. The Non-Local Multi-fiber Network

2.1. Multi-Fiber Network

The proposed network is based on Multi-Fiber Network [2] which aims to reduce the computational cost of the 3D Deep ConvNets. It slices the network into an ensemble of lightweight networks so called fibers. Multi-Fiber Network

is based the ResNets [5] and targets reducing the computational cost, by slicing the residual unit into N parallel isolated paths named fibers. Further, features are routed and multiplexed using a multiplexer. The multiplexer aims to learn the interactions across different fibers and re-distribute the features again among different fibers as illustrated in figure 1. Before each layer, batch normalization and ReLU non-linearity are used. The whole network is shown in figure 2(a).

2.2. Non-Local Neural Network

Non-Local Neural Network [8] extracts the long-term temporal information which has demonstrated the effectiveness of non-local operation in different applications. The non-local module is defined as follows:

$$y_i = \sum f(x_i, x_j)g(x_j) \quad (1)$$

where i and j are the index of an output position in time or space. x is the input tensor representing video and y is the output representation. f represents the relation between the positions i and j . Here $f(x_i, x_j)$ is gaussian and is computed as follows:

$$f(x_i, x_j) = e^{x_i^T x_j} \quad (2)$$

where $x_i^T x_j$ is dot-product similarity.

2.3. The Proposed Network

According to [2], Multi-Fiber Network achieved the state-of-the-art in video recognition with less computational cost which we also found empirically. However, Multi-Fiber Network focuses only on local neighborhood. Non-local module [8] is adopted to lend the network the ability of capturing long term temporal information needed for action anticipation. In Non-Local Multi-Fiber Network (NL-MFN), non-local module is plugged in after the four multi-fiber modules as shown in figure 2(b).

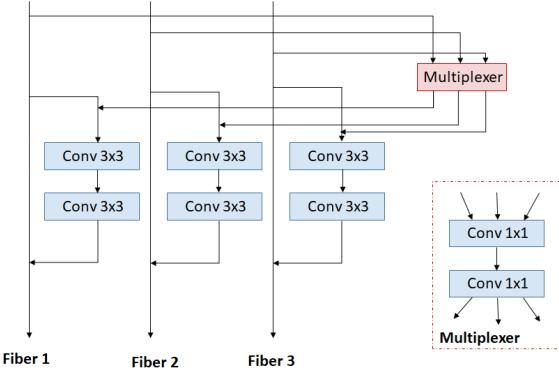


Figure 1. The architecture of the multi-fiber module. The multiplexer gathers the feature maps and re-distribute the feature maps using 2 1x1 convolution layer as in [2]

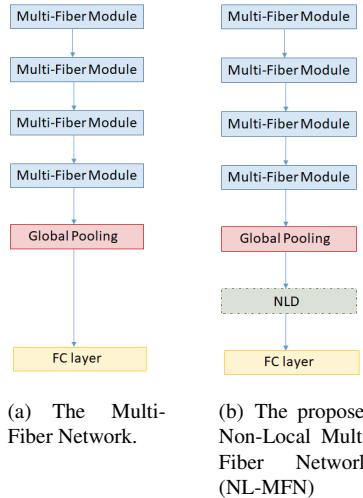


Figure 2. The network architecture used for video action anticipation.

3. Experiments on Action Anticipation Challenge

Action Anticipation is the ability of foreseeing the upcoming action. Given a video segment $A_i = [t_{si} - t_o - t_a, t_{ei} - t_a]$ where t_{si} and t_{ei} are the start and the end time of segment, t_a is the anticipation time and equals to 1 sec, and t_o is the observation time, the action class C_a is anticipated. Following [3], the video should be anticipated by only observing the preceding action time by $\tau_a = 1$ sec. As in [3], we feed the network with video segments preceding annotated actions and train it to predict the labels. According to [3], the dataset is imbalanced dataset where it includes many, few, and zero shots classes. So, focal loss [6] was adopted as a loss function which is defined as follows:

$$FL(p_t) = -(1 - p_t)^\gamma \log(p_t) \quad (3)$$

where γ is set empirically to 2 as in [6]. As the task is to anticipate the action, however the annotation is in terms of verbs and nouns. Multi-task loss is adopted for training the network where three losses are set for training including verbs, nouns, and actions. The dataset includes 125 verbs and 352 nouns. Moreover, the number of actions is 2513 and is computed from the unique verb and noun annotation tuples of the training dataset only.

3.1. Experiments

In our submission, RGB modality is used only. First, a Multi-Fiber Network is trained by fine tuning a pre-trained model on Kinetics [1] and UCF101 [7] using Pytorch. The training using a single Nvidia Titan Xp with a batch size of 8 for 30 epochs with initial learning rate of 0.01 and is dropped by $\frac{1}{10}$ every 10 epochs. The network weights are used as initialization for the NL-MFN except for non-local module and the fully connected layers. The NL-MFN is further finetuned for 25 epochs with a learning rate of 0.01 and is dropped by $\frac{1}{10}$ every 10 epochs. The results are shown in Table 1 as appeared on the leaderboard securing the second place.

Acknowledgment

We would like to thank NVIDIA for supporting our Research with TITAN Xp GPU.

References

- [1] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. [1](#), [2](#)
- [2] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Multi-fiber networks for video recognition. In *European Conference on Computer Vision (ECCV)*, 2018. [1](#), [2](#)
- [3] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The dataset. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV*, pages 753–771, 2018. [1](#), [2](#)
- [4] Bernard Ghanem, Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015. [1](#)
- [5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. [1](#)
- [6] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceed-*

Table 1. Experiments on EPIC-Kitchens Action Anticipations.

Dataset	Verb (Top 1)	Noun (Top 1)	Action (Top 1)	Verb (Top 5)	Noun (Top 5)	Action (Top 5)	Verb (Prec.)	Noun (Prec.)	Action (Prec.)	Verb (Rec.)	Noun (Rec.)	Action (Rec.)
Seen Kitchens (S1)	34.40%	23.36%	13.20%	79.07%	45.57%	31.08%	26.36%	21.81%	5.28%	19.47%	20.01%	5.20%
Unseen Kitchens (S2)	27.89%	15.53%	8.50%	70.47%	34.28%	20.38%	17.77%	12.32%	3.28%	9.35%	12.11%	3.84%

ings of the IEEE international conference on computer vision, pages 2980–2988, 2017. 2

- [7] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. 2
- [8] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *CVPR*, 2018. 1

Leveraging the Present to Anticipate the Future in Videos

Antoine Miech^{1,2} Ivan Laptev^{1,2} Josef Sivic^{1,2,3} Heng Wang⁴ Lorenzo Torresani⁴ Du Tran⁴
¹Inria ²École Normale Supérieure ³CIIRC ⁴Facebook AI

Abstract

Anticipating actions before they are executed is crucial for a wide range of practical applications including autonomous driving and robotics. While most prior work in this area requires partial observation of executed actions, in the paper we focus on anticipating actions seconds before they start. Our proposed approach is the fusion of a purely anticipatory model with a complementary model constrained to reason about the present. In particular, the latter predicts present action and scene attributes, and reasons about how they evolve over time. By doing so, we aim at modeling action anticipation at a more conceptual level than directly predicting future actions. Our model outperforms previously reported methods on the EPIC-KITCHENS. Note that this challenge technical report is an extract from [17].

1. Introduction

Automatic video understanding has improved significantly over the last few years. Such advances have manifested in disparate video understanding tasks, including action recognition [2, 5, 7, 12, 22, 24], temporal action localization [21, 23, 28, 29], video search [8], video summarization [18] and video categorization [16]. In this work, we focus on the problem of anticipating future actions in videos as illustrated in Figure 1.

A significant amount of prior work [2, 5, 7, 12, 13, 22, 24, 25, 27] in automatic video understanding has focused on the task of action recognition. The goal of action recognition is to recognize what action is being performed in a given video. While accurate recognition is crucial for a wide range of practical applications such as video categorization or automatic video filtering, certain settings do not allow for complete and even partial observation of action before it happens. For instance, an autonomous car should be able to recognize the intent of a pedestrian to cross the road much before the action is actually initiated in order to avoid an accident. In practical applications where we seek to act before an action gets executed, being able to anticipate the future given the present is critical.

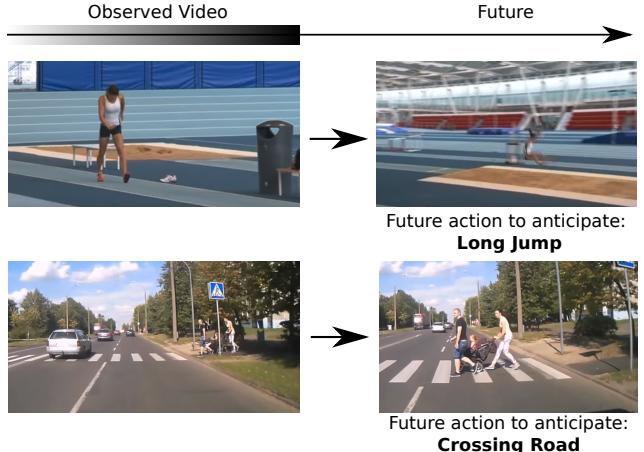


Figure 1: **Action anticipation.** Examples of action anticipation in which the goal is to anticipate future actions in videos seconds before they are performed.

Anticipating the future, especially long-term, is a challenging task because the future is not deterministic: several outcomes are possible given the current observation. To reduce uncertainty, most work in this field [1, 9, 10, 14, 19, 20] requires partially observed execution of actions. In this paper, we address the task of action anticipation even when no partial observation of the action is available. While prior work [4, 11, 15, 26] has addressed this same task, in this work, we specifically focus on better leveraging recognition models to improve future action prediction. We propose a fusion of two approaches: one directly anticipates the future while the other first recognizes the present and then anticipates the future, given the present.

1.1. Contributions

The contributions of our work are: **(i)** We propose a new framework for the task of anticipating human actions several seconds before they are performed. Our model is decomposed into two complementary models. The first, named the predictive model, anticipates action directly from the visual inputs. The second one, the transitional model, is first constrained to predict what is happening in the observed time interval and then leverages this prediction to anticipate the future actions. **(ii)** We present extensive ex-

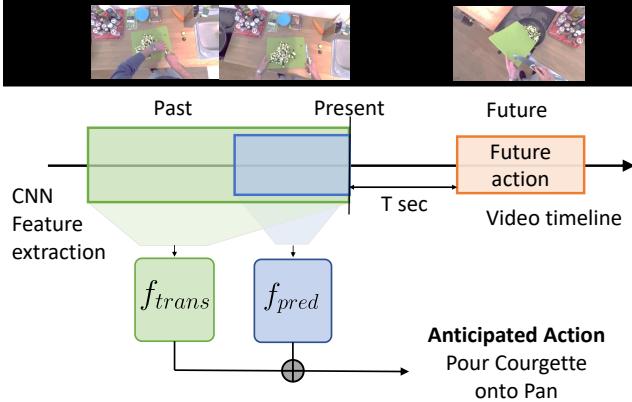


Figure 2: **Overview of our approach.** Our task is to predict an action T seconds before it starts to be performed. Our model is a combination of two complementary modules: the predictive model and the transitional model. While the predictive model directly anticipates the future action, the transitional model is first constrained to output what is currently happening. Then, it uses this information to anticipate future actions.

periments on the EPIC-KITCHENS [3] dataset with state-of-the-art results.

2. Action Anticipation Model

Our goal is to anticipate an action T seconds before it starts. More formally, let V denote a video. Then we indicate with $V_{a:b}$ the segment of V starting at time a and ending at time b , and with y_c the label of the action that starts at time c . We would like to find a function f such that $f(V_{0:t})$ predicts y_{t+T} . The main idea behind our model is that we decompose f as a weighted average of two functions, a predictive model f_{pred} and a transitional model f_{trans} :

$$f = \alpha f_{pred} + (1 - \alpha) f_{trans}, \quad \alpha \in [0, 1], \quad (1)$$

where α is a dataset dependent hyper-parameter chosen by validation. The first function f_{pred} is trained to predict the future action directly from the observed segment. On the other hand, f_{trans} is first *constrained* to compute high-level properties of the observed segment (e.g., attributes or the action performed in the present). Then, in a second stage, f_{trans} uses this information to anticipate the future action. In the next subsections we explain how to learn f_{pred} and f_{trans} . Figure 2 presents an overview of the proposed model.

2.1. Predictive model f_{pred}

The goal of the predictive model f_{pred} is to directly anticipate future action from the visual input. As opposed

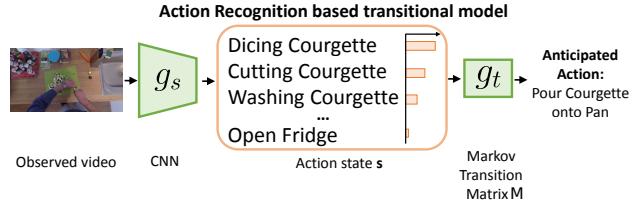


Figure 3: **Illustration of the transitional model.** Our Action Recognition (AR) based transitional model learns to prediction future actions based on the predictions of an action recognition classifier applied on current/present frames (clips).

to f_{trans} , f_{pred} is not subject to any specific constraint. Suppose that we are provided with a training video V with action labels $y_{t_0+T}, \dots, y_{t_n+T}$. For each label y_{t_i+T} , we want to minimize the loss:

$$l(f_{pred}(V_{s(t_i):t_i}), y_{t_i+T}), \quad (2)$$

where $s(t_i) = \max(0, t_i - t_{pred})$, l is the cross entropy loss, t_{pred} is a dataset dependent hyper-parameter, also chosen by validation, that represents the maximum temporal interval of a video f_{pred} has access to. This hyper-parameter is essential because looking too much in the past may add irrelevant information that degrades prediction performance. This loss is then summed up over all videos from the training dataset. In this work, f_{pred} is a linear model which takes as input a video descriptor which we describe in section 3.3.

2.2. Transitional model f_{trans}

The transitional model f_{trans} splits the prediction into two stages: g_s and g_t . The first stage g_s aims at recognizing a current state s , describing the observed video segment. The state s can represent an action or a latent action-attribute. The second stage g_t takes as input the current state s , and anticipates the next action given the current state s . g_s can be thought of as a complex function extracting high-level information from the observed video segment, while g_t is a simple (in fact, linear) function operating on the state s and modeling the correlation between the present state and the future action. We will next explain in detail how we define the current state s and how we model the transition function g_t .

Transitional Model based on Action Recognition. Real-world videos often consist of a sequence of elementary actions performed by a person in order to reach a final goal such as *Preparing coffee*, *Changing car tire* or *Assembling a chair*. Many datasets come with a training set where each video has been annotated with action labels and segment boundaries for all occurring actions (e.g EPIC-KITCHENS,

Breakfast). When this is available we can use action labels instead of predefined visual attributes for state \mathbf{s} . The intuition behind our claim is the fact that the anticipation of the next action significantly depends on the present being performed action. In other words, we make a Markov assumption on the sequence of performed actions. More formally, suppose we are provided with an ordered sequence of action annotations $(a_0, \dots, a_N) \in \{1, \dots, K\}^N$ for a given video, where a_n defines the action class performed in video segment V_n . We propose to model $P(a_{n+1} = i|V_n)$ as follows:

$$P(a_{n+1} = i|V_n) = \sum_{j=1}^K P(a_{n+1} = i|a_n = j)P(a_n = j|V_n) \quad (3)$$

$\forall n \in \{0, \dots, N-1\}$, $i \in \{1, \dots, K\}$. This reformulation decomposes the computation of $P(a_{n+1} = i|V_n)$ in terms of two factors: 1) an action recognition model $g_s(V_n)$ that predicts $P(a_n = j|V_n)$, i.e., the action being performed in the present; 2) a transition matrix M that captures the statistical correlation between the present and the future action, i.e., such that $M_{ij} \approx P(a_{n+1} = i|a_n = j)$. In this scenario, g_t takes as input the probability scores of each action given by g_s to anticipate the next action in a probabilistic manner:

$$g_t(\mathbf{s}) = M\mathbf{s}, \quad (4)$$

$$P(a_{n+1} = i) = \sum_{j=1}^K M_{i,j} s_j = [g_t(\mathbf{s})]_i. \quad (5)$$

In practice, we compute M by estimating the conditional probabilities between present and future actions from the sequences of action annotations in the training set. Figure 3 illustrates this model.

3. Experiments

In this section, we evaluate our approach on the EPIC-KITCHENS dataset.

3.1. EPIC-KITCHENS Datasets

EPIC-KITCHENS [3] is a large-scale cooking video dataset containing 39,594 accurate temporal segment action annotations. Each video is composed of a sequence of temporal segment annotations. Three different tasks are proposed together with the dataset: object detection, action recognition and *action anticipation*. The action anticipation task is to predict an action one second before it has started. The dataset contains three different splits: the training set, the seen kitchens test set (S1) composed of videos from kitchens also appearing in the training set and finally the unseen kitchens test set (S2) with kitchens that are not appearing in the training set. A publicly available challenge is also

Model	Pretrain	Fine-tune	Action	Verb	Noun
ResNet-50	Imagenet	No	3.4	24.5	7.4
R(2+1)D-18	Kinetics	No	5.2	27.2	10.3
R(2+1)D-18	Kinetics	EK-Anticip.	5.0	24.6	9.7
R(2+1)D-18	Kinetics	EK-Recogn.	6.0	27.6	11.6

Table 1: **Effects of pre-training.** Action anticipation top-1 per clip accuracy on EPIC-KITCHENS with different models and pre-training datasets.

organized to keep track of the best performing approach on this anticipation task. Because of this public challenge, the labels of S1 and S2 test sets are not available. Thus, most of our results are reported on our validation set composed of the following kitchens: P03, P14, P23 and P30. We also report results evaluated by the challenge organizers on the held-out test set. Unless specified otherwise, for comparison purposes, we report experiments with $T = 1$ sec.

3.2. Action label space

As opposed to [3] where the action labels are predicted by first predicting the verbs and nouns separately, we directly predict the cartesian action label space by considering the combination of (verb, action) that only appear at training (roughly 3k actions).

3.3. Video Representation

In this subsection we discuss how we represent the observed video segment V to perform action prediction. Our overall strategy is to split the video into clips, extract clips representation and perform pooling over these clips. Given an input video segment V , we uniformly split it into small clips $V = [V_1, \dots, V_N]$ where each clip V_i , $i \in [1, N]$ is short enough (e.g. 8 or 16 frames) that it can be fed into a pretrained video CNN C . From the penultimate layer of the CNN we extract an L_2 -normalized one-dimensional representation $C(V_i)$ for each clip V_i . Then we perform a temporal aggregation $Agg([C(V_1), \dots, C(V_N)])$ of the extracted features in order to get a one-dimensional video representation for V . In our experiments, C is the R(2+1)D network of 18-layers from Tran *et al.* [24]. We perform a simple max pooling to aggregate features from all clips, but more sophisticated temporal aggregation techniques [16] can also be used in our model.

Leveraging the present for pretraining. In previous work [3, 26] the video representation was learned by fine-tuning a pretrained video CNN on the task of action anticipation. Instead, we propose to finetune the CNN representation on the task of action recognition on the target dataset. More specifically, instead of training the CNN on video clips sampled before action starts, we train it on clips

	Action		Verb		Noun	
	A@1	A@5	A@1	A@5	A@1	A@5
Transitional	5.1	17.1	25.2	72.0	12.1	33.2
Predictive	6.3	17.3	27.4	73.1	11.9	31.5
Predictive + Transitional	6.7	19.1	27.3	73.5	12.9	34.6
Transitional (with GT)	16.1	29.4	29.3	63.3	30.7	44.4
Action recognition	12.1	30.0	39.3	80.0	23.1	49.3

Table 2: **Transitional and predictive model ablation.** Transitional model and predictive model ablation study on our EPIC-KITCHENS validation set with $T = 1 \text{ sec}$. Grey rows should be interpreted as accuracies upper bounds.

sampled in the action segment interval. This is motivated by the fact that the task of action recognition is “easier” than action anticipation and thus it may lead to better feature learning. Table 1 reports accuracies on the EPIC-KITCHENS validation set obtained with our predictive model applied to different CNN representations. These results illustrate the benefit of fine-tuning the CNN on action recognition, instead of action anticipation as done in prior work [3, 26]. The Table provide also numbers for two additional baselines corresponding to 1) using the CNN pretrained on Kinetics without finetuning and 2) extracting features from a ResNet-50 2D CNN pretrained on Imagenet. It can be noted that the best accuracies for actions, verbs and nouns are obtained with the CNN finetuned on the action recognition task of EPIC-KITCHENS. Based on these results, in the rest of the work, we use CNN features computed from a R(2+1)D-18 first pretrained on Kinetics [2] and then finetuned for action recognition on the target dataset.

3.4. Ablation study

In order to understand the benefits of the different components in our model, we evaluate the predictive model separately from the transitional model. Table 2 summarizes the results achieved on the validation set of EPIC-KITCHENS [3]. Interestingly, combining the predictive model with the transitional models yields further accuracy gains. This suggests that the predictions are complementary.

We also show in grey, an accuracy upper bound achieved when directly recognizing the future frame as opposed to predicting from the past one (row Action recognition). The grey row Transitional (AR with GT) experiments shows the accuracy achieved when the transitional model is provided the groundtruth label of the last observed action. The improvement when using the groundtruth label is significant. This suggests that a large cause of missing performance is weak action recognition models and that better action recognition will produce stronger results for prediction.

3.5. Comparison to the state-of-the-art

We compare our approach to the state-of-the-art on the EPIC-KITCHENS dataset. Table 3 reports results obtained from the EPIC-KITCHENS unseen kitchens action anticipation challenge submission server. Note that our EPIC-KITCHENS submission is done under the anonymous nickname of *masterchef* and is reported by the row **Ours (Predictive + Transitional)** in this paper. On both datasets, our method outperforms all previously reported results under almost all metrics. Note that our best submitted model on the EPIC-KITCHENS challenge is simple and does not make use of any ensembling nor optical flow input.

4. Conclusion

We have described a new model for future action anticipation. The main motivating idea for our method is to model action anticipation as a fusion of two complementary modules. The predictive approach is a purely anticipatory model. It aims at directly predicting future action given the present. On the other hand, the transitional model is first constrained to recognize what is currently seen and then uses this output to anticipate future actions.

Acknowledgment. The project was partially supported by the Louis Vuitton - ENS Chair on Artificial Intelligence, the ERC grant LEAP (No. 336845), the CIFAR Learning in Machines&Brains program, and the European Regional Development Fund under the project IMPACT (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000468).

References

- [1] M. S. Aliakbarian, F. S. Saleh, M. Salzmann, B. Fernando, L. Petersson, and L. Andersson. Encouraging lstms to anticipate actions very early. In *ICCV*, 2017. [1](#)
- [2] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. [1, 4](#)
- [3] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018. [2, 3, 4, 5](#)
- [4] Y. A. Farha, A. Richard, and J. Gall. When will you do what?-anticipating temporal occurrences of activities. In *CVPR*, 2018. [1](#)
- [5] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. [1](#)
- [6] A. Furnari and G. M. Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. *arXiv preprint arXiv:1905.09035*, 2019. [5](#)
- [7] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017. [1](#)

	Action				Verb				Noun			
	A@1	A@5	P	R	A@1	A@5	P	R	A@1	A@5	P	R
Damen <i>et al.</i> (TSN Fusion) [3]	1.7	9.1	1.0	0.9	25.4	68.3	13.0	5.7	9.8	27.2	5.1	5.6
Damen <i>et al.</i> (TSN Flow) [3]	1.8	8.2	1.1	0.9	25.6	67.6	10.8	6.3	8.4	24.6	5.0	4.7
Damen <i>et al.</i> (TSN RGB) [3]	2.4	9.6	0.9	1.2	25.3	68.3	7.6	6.1	10.4	29.5	8.8	6.7
DMI-UNICIT [6]	9.1	21.9	3.2	4.6	26.6	68.1	16.6	11.1	15.5	35.3	9.9	11.7
Ours (Predictive)	6.1	18.0	1.6	2.9	27.5	71.1	12.3	8.4	10.8	30.6	8.6	8.7
Ours (Predictive + Transitional)	7.2	19.3	2.2	3.4	28.4	70.0	11.6	7.8	12.4	32.2	8.4	9.9

Table 3: EPIC-KITCHENS results on hold-out unseen test set S2. The official ranking is based on the action top 1 accuracy score (A@1). A@1: top-1 accuracy, A@5: top-5 accuracy, P: precision, R: recall. Challenge website details: <https://competitions.codalab.org/competitions/20071>. Note that our best model was submitted under the anonymous nickname **masterchef**.

- [8] L. A. Hendricks, O. Wang, E. Shechtman, J. Sivic, T. Darrell, and B. Russell. Localizing moments in video with natural language. *ICCV*, 2017. 1
- [9] M. Hoai and F. De la Torre. Max-margin early event detectors. 2014. 1
- [10] Y. Kong, Z. Tao, and Y. Fu. Deep sequential context networks for action prediction. In *CVPR*, 2017. 1
- [11] T. Lan, T.-C. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *ECCV*, 2014. 1
- [12] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 1
- [13] J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. In *CVPR*, 2011. 1
- [14] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *CVPR*, 2016. 1
- [15] T. Mahmud, M. Hasan, and A. K. Roy-Chowdhury. Joint prediction of activity labels and starting times in untrimmed videos. In *ICCV*, 2017. 1
- [16] A. Miech, I. Laptev, and J. Sivic. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017. 1, 3
- [17] A. Miech, I. Laptev, J. Sivic, H. Wang, L. Torresani, and D. Tran. Leveraging the present to anticipate the future in videos. In *CVPR Precognition Workshop*, 2019. 1
- [18] B. A. Plummer, M. Brown, and S. Lazebnik. Enhancing video summarization via vision-language embedding. In *CVPR*, 2017. 1
- [19] M. S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *ICCV*, 2011. 1
- [20] Y. Shi, B. Fernando, and R. Hartley. Action anticipation with rbf kernelized feature mapping rnn. In *ECCV*, 2018. 1
- [21] Z. Shou, H. Gao, L. Zhang, K. Miyazawa, and S.-F. Chang. Autoloc: Weaklysupervised temporal action localization in untrimmed videos. In *ECCV*, 2018. 1
- [22] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *ICLR*, pages 568–576, 2014. 1
- [23] K. K. Singh and Y. J. Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *ICCV*, 2017. 1
- [24] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018. 1, 3
- [25] G. Varol, I. Laptev, and C. Schmid. Long-term Temporal Convolutions for Action Recognition. *PAMI*, 2017. 1
- [26] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating visual representations from unlabeled video. In *CVPR*, 2016. 1, 3, 4
- [27] H. Wang and C. Schmid. Action Recognition with Improved Trajectories. In *ICCV*, 2013. 1
- [28] L. Wang, Y. Xiong, D. Lin, and L. Van Gool. Untrimmednets for weakly supervised action recognition and detection. In *CVPR*, 2017. 1
- [29] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin. Temporal action detection with structured segment networks. 2017. 1

Attending to Long(er)-Term Memory for Anticipation

Yaser Souris, Tridivraj Bhattacharyya, Juergen Gall
University of Bonn
`{souri@iai., s6trbhat@, gall@iai.} uni-bonn.de`

Luca Minciullo
Toyota Motor Europe
`luca.minciullo@toyota-europe.com`

Abstract

The proposed baseline for action anticipation from the EPIC-Kitchens [3] paper looks at the last 1 second of the observable past to anticipate the action in 1 second ahead. We suggest that looking much longer in the past might be helpful. In this work we propose a method that looking at the recent observable past (last 1 observable second) attends to much longer term past (last 12 seconds) and gathers information from this longer term memory to perform anticipation. Our preliminary results suggest that this method is promising and it is able to achieve the best reported verb anticipation performance on both EPIC-Kitchens test sets.

1. Introduction

The proposed baseline for action anticipation from the EPIC-Kitchens [3] paper looks at the last 1 second of the observable past to anticipate the action at 1 second ahead. Given the nature of the dataset, we hypothesize that looking into much longer past might be helpful. Usually there are information available in much longer past regarding the general activity that the participant is performing that are helpful for anticipation.

To achieve this and mitigate GPU memory problems we don't perform end-to-end training from raw pixel values or fine-tune the backbone network. We extracted frame-wise features using a pretrained network from the videos and use the extracted features as input to our network. Our network uses a WaveNet [7] style network to do temporal modeling of the input features and uses a multi-headed attention [8] mechanism for attention. Similar to the proposed baseline [3] we anticipate verb and nouns independently.

1.1. Dataset Splits

We have split the training set into *train* and *validation* sets. In the following we refer to the whole training set (containing 272 videos) as the *trainval* set.

Similar to [1] we split the *trainval* set based on person ids. This means that the created validation set will in spirit

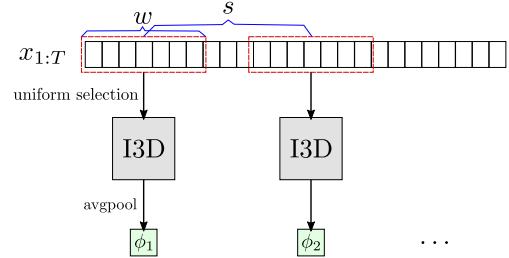


Figure 1. Feature extraction pipeline.

be similar to the unseen test set (S2). If the person id of a video is smaller than or equal to 25 it is considered in the *train* set, otherwise it is considered in the *validation* set. This means that the *train* set will have 200 videos, while the *validation* set has 72 videos.

2. Method

For long-term, untrimmed video processing we use a pretrained-network for feature extraction and don't perform end-to-end training or fine-tuning from raw pixel values. Following, we first describe our feature extraction pipeline, then we will describe our proposed network.

2.1. Feature Extraction

We extract frame-wise features from the EPIC-Kitchens videos. We do so regardless of whether a section of the video is annotated with an action in the EPIC-Kitchens dataset or not. In other words, we extract features even from the "background" section of the videos. An overview of the feature extraction pipeline is shown in Figure 1. This method of feature extraction has shown to be quite effective in action segmentation both fully supervised [4] and weakly supervised [6].

Features are extracted at specific frequency ν_ϕ . The I3D [2] network is pretrained on the RGB stream of the Kinetics-400 [5] dataset and we won't fine-tune it.

We note that Kinetics-400 videos have a frame-rate of

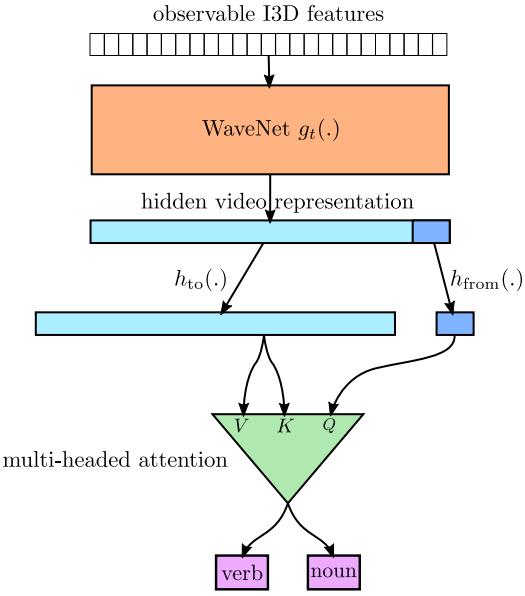


Figure 2. Network design.

$f_{kinetics} = 25$ while EPIC-Kitchens videos have a frame rate of $f_{epic} = 60$. From a window of size w we uniformly select 64 frames to create the input to the I3D network, where $w = 64 \times \frac{f_{epic}}{f_{kinetics}}$. The stride of the input window depends on the frequency of feature extraction and is equal to $s = \frac{f_{epic}}{\nu_\phi}$. This means that the higher the frequency of feature extraction, the smaller the stride of the input window. We do not rescale or crop the frames and perform spatial and temporal average pooling to arrive at a fixed sized feature vector.

2.2. Network Design

As input we gather the features of the last τ seconds of **observable** past (not including the anticipation window). We use a 7 layer WaveNet [7] with dilation factors 1, 2, 4, 8, 16, 32 & 64, hidden size of 64 and ReLU nonlinearity to perform temporal modeling ($g_t(\cdot)$) of the input features. The WaveNet has shown to be very effective for untrimmed video processing for action segmentation [4, 6]. We term the output of the WaveNet as the *hidden video representation*. It has the same temporal resolution as the input. Using two linear transformations h_{to} and h_{from} we create the two parts of the hidden representation that are going to be used as input to the multi-headed attention (MHA) [8] module. The *attend from part* is used as query (Q) in MHA and is the last 1 second fo the hidden video representations after the linear transformation h_{from} . The *attend to part* is used as both key (K) and value (V) in MHA and is the whole hidden video representation after linear transformation h_{to} . The MHA module produces a fixed size output that is used

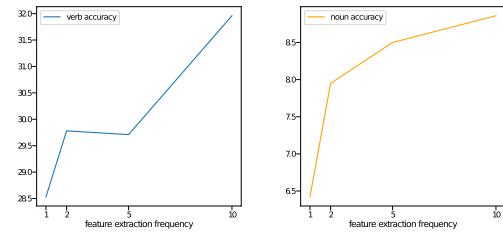


Figure 3. Results on the validation set. Higher frequency performs better.

to predict the verb and nouns from independently using two fully connected layers. Sum of the two cross entropy losses for verb and noun prediction is used as the training objective. In Figure 2 an overview of our method is shown.

2.3. Implementation Details

We use a batch size of 8 and optimize our networks for 60 epochs using the Adam optimizer. The learning rate is set to 3×10^{-4} and is lowered by a factor of 10 after 50 epochs. Hidden sizes of WaveNet and all other hidden size dimensions are set to 64. The number of heads in MHA is 1 and increasing it did not help with getting better performance on the validation set.

3. Results

3.1. Results on Validation Set

We tried to find what is the best feature extraction frequency ν_ϕ using the validation set. As can be seen from Figure 3, higher feature extraction frequencies generally perform better.

3.2. Test-Set Results

We submit our method to the server for evaluation on the test set. We find that after finding the hyperparameters using the *validation* set it is important that you train your method on the whole *trainval* set as this results in a considerable performance improvement compared to only training on the *train* set. We also evaluate an ensemble of 5 differently initialized models. We find that this kind of ensemble only achieves slight improvement over a single model. The complete set of results on the test set can be seen in Table 1. Our method shows best performing verb anticipation performance on both test sets.

Acknowledgment We would like to thank Toyota Motor Europe for funding and supporting this work.

References

- [1] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. Object level visual reasoning in videos.

Training Set	Ensemble	Seen set (S1)			Unseen set (S2)		
		Verb accuracy	Noun accuracy	Action accuracy	Verb accuracy	Noun accuracy	Action accuracy
train	X	32.50	10.79	3.75	30.18	8.50	2.94
trainval	X	34.57	12.29	5.12	31.65	9.53	3.76
trainval	✓(5)	34.94	13.06	5.23	32.37	9.66	3.52

Table 1. Results on the test set.

In *ECCV*, 2018.

- [2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.
- [3] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.
- [4] Yazan Abu Farha and Juergen Gall. MS-TCN: Multi-Stage Temporal Convolutional Network for Action Segmentation. In *CVPR*, 2019.
- [5] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv*, 2017.
- [6] Yaser Souri, Mohsen Fayyaz, and Juergen Gall. Weakly Supervised Action Segmentation Using Mutual Consistency. In *arXiv*, 2019.
- [7] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, 2016.
- [8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.