EPIC-KITCHENS-55 - 2020 Challenges Report

Dima Damen, Evangelos Kazakos, Will Price, Jian Ma, Hazel Doughty
University of Bristol, UK

Antonino Furnari, Giovanni Maria Farinella
University of Catania, Italy

Abstract

This report summarises the EPIC-KITCHENS-55 2020 challenges, and their findings. It serves as an introduction to all technical reports that were submitted to the EPIC@CVPR2020 workshop, and an official announcement of the winners.

1. EPIC-KITCHENS-55

As the largest dataset in egocentric vision, EPIC-KITCHENS-55 continued to receive significant attention from the research community over the past year. EPIC-KITCHENS-55 has a number of unique features that distinguished its collection, including *non-scripted* and *untrimmed* nature of the footage captured in participants' *native environments*. More details on the dataset's collection and annotation pipeline are available in this year's PAMI extended version [3].

This report details the submissions and winners of the 2020 edition of the three challenges available on CodaLab: Action Recognition, Action Anticipation and object detection. For each challenge, submissions were limited per team to a maximum of 50 submissions in total, as well as a maximum daily limit of 1 submission. In Sec. 2, we detail the general statistics of dataset usage in its first year. The results for the Action Recognition , Action Anticipation and Object Detection in Video challenges are provided in Sec. 3, 4 and 5 respectively. The winners of the 2020 edition of these challenges are noted in Sec. 6.

Details of the 2019 challenges are available from the technical report [4].

2. Reception and User Statistics

Since its introduction, EPIC-KITCHENS-55 received significant attention with a total of 23K page views since April 2018, and 10K page views over the past year. Fig 1 shows page views of the dataset's website, based on country. The Action Recognition challenge received the largest number of participants (46 teams) and submissions (368 submissions). The Action Anticipation challenge has 20 participating teams, and received 244 submissions. The Object Detection in Video challenge has 16 participating teams and 182 submissions. Of these, 8 teams have declared their affiliation and submitted technical reports for the Action Recognition challenge compared to 5 in the Action Anticipation challenge and 4 in the Object Detection in Video challenge. This report includes details of these teams' submissions. A snapshot of the complete leaderboard, when the 2020 challenge concluded on the 30th of May, is available at http://epic-kitchens.github.io/2020#results.

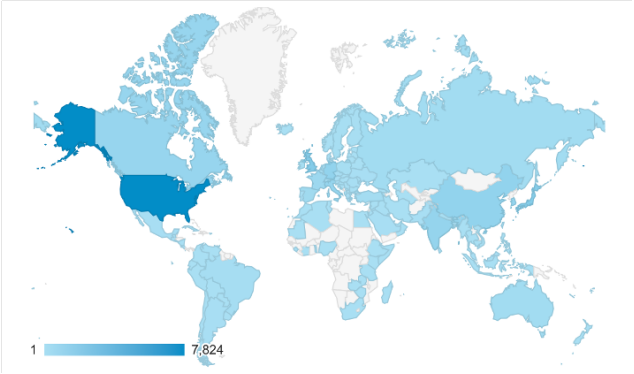


Figure 1: Heatmap of countries based on EPIC-KITCHENS-55 page view statistics.

3. Action Recognition Challenge

The Action Recognition challenge has been set similar to previous challenges [1, 10]. In both train and test sets, the start and end times of an action are given. Correct recognition of the action includes correctly recognising the 'verb' of the action as well as the main interacting 'noun'. For example, the action 'put plate in sink' would need to be recog-

June 22nd, 2020
Technical Report

	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
	1	**UTS-Baidu**	14	05/28/20	**70.41**	**52.85**	**42.57**	**90.78**	**76.62**	**63.55**	**60.44**	**47.11**	**24.94**	**45.82**	**50.02**	**26.93**
	2	**NUS-CVML**	18	05/29/20	**63.23**	**46.45**	**41.59**	**87.50**	**70.49**	**64.11**	**51.54**	**42.09**	**25.37**	**40.99**	**42.69**	**26.98**
		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**
	3	**SAIC-Cambridge**	34	05/27/20	**69.43**	**49.71**	**40.00**	**91.23**	**73.18**	**60.53**	**60.01**	**45.74**	**24.95**	**47.40**	**46.78**	**25.27**
S1	3	**FBK-HuPBA**	50	05/29/20	**68.68**	**49.35**	**40.00**	**90.97**	**72.45**	**60.23**	**60.63**	**45.45**	**21.82**	**47.19**	**45.84**	**24.34**
	4	GT-WISC-MPI	12	01/30/20	68.51	49.96	38.75	89.33	72.30	58.99	51.04	44.00	23.70	43.70	47.32	23.92
	5	G-Blend	14	05/28/20	66.67	48.48	37.12	88.90	71.36	56.21	51.86	41.26	20.97	44.33	44.92	21.48
	6	**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**
		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**
	7	CVG Lab Uni Bonn	12	05/27/20	62.86	43.44	34.53	89.64	69.24	56.73	52.82	38.81	19.21	44.72	39.50	21.80
	1	**UTS-Baidu**	14	05/28/20	**60.43**	**37.28**	**27.96**	**83.07**	**63.67**	**46.81**	**35.23**	**32.60**	**17.35**	**28.97**	**532.78**	**19.82**
	2	**GT-WISC-MPI**	12	01/30/20	**60.02**	**37.49**	**27.38**	**82.55**	**63.47**	**45.10**	**37.75**	**31.56**	**16.52**	**31.61**	**33.72**	**20.02**
	3	**NUS-CVML**	18	05/29/20	**54.56**	**33.46**	**26.97**	**80.40**	**60.98**	**46.43**	**33.60**	**30.54**	**14.99**	**25.28**	**28.39**	**17.97**
	4	N/A	14	05/28/20	58.25	36.74	26.56	81.29	60.33	43.63	33.66	27.09	15.71	28.09	32.34	18.89
S2	5	FBK-HuPBA	50	05/29/19	56.67	33.9	25.71	81.87	59.68	44.42	30.72	27.25	12.74	25.09	29.46	17.93
		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**
	6	SAIC-Cambridge	34	05/27/20	57.6	34.69	24.62	81.84	61.25	41.38	37.81	30.81	16.61	30.37	32.40	17.81
		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**
	7	**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**
	8	CVG Lab Uni Bonn	12	05/27/20	51.62	26.73	20.45	79.14	52.41	38.17	30.19	21.49	9.77	27.50	23.88	13.50

Table 1: Results of EPIC-KITCHENS-55 Action Recognition challenge - 1 June 2020

nised into the verb class 'put' and the noun class 'plate'.

Table 1 shows the top-8 entries on the challenge leaderboard for 2020. Methods are ranked based on top-1 action accuracy (noted by arrow), which was used to decide on the overall rank. The top-3 submissions are highlighted in bold. Shaded lines reflect the top-3 winners from 2019, to allow direct comparison. Compared to the top winner of last year, the improvement remains marginal - 1.20% increase in top-1 action accuracy on S1 and 2.76% on S2. However, 5 more teams outperform the second-ranked method from 2019. Moreover, the performance of the same method across both S1 (seen kitchens) and S2 (unseen kitchens) subsets is comparable.

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. Similar to their proposed winning solution in last year's challenge, the authors utilise the active object bounding-box annotations to train for the challenge. The new architecture has two separate branches, one for verbs and one for nouns, where the top-K Region of Interest (RoI) features are max-pooled and integrated with both branches via local and global alignment respectively. The two branches interact with each other via a cross-stream gating mechanism.

NUS-CVML (Rank 2 - S1, Rank 3 - S2) The second-ranking entry utilises long-term temporal context from the untrimmed video and *coupling blocks* integrate recent observations to long-range future and past context. Both recent and long-range representations are multi-scale, and a Temporal Aggregation Block consisting of multiple coupling blocks, perform multi-granular temporal aggregation.

SAIC-Cambridge (Rank 3 - S1, Rank 6 - S2) uses a recently proposed spatio-temporal attention network [7] as the basic block of their solution. We refer the reader to [7] for details. To further improve performance, temporal information from neighbouring actions is utilised by learning a low rank factorisation of the action matrix. Finally, 4 modalities are trained and late fused: RGB, attended RGB, Flow, and Audio, inspired by [5].

FBK-HUPBA (Rank 3 - S1, Rank 5 - S2) designs an ensemble of variants of gate-shift modules (GSM) and EgoACO, an extension of long-short term attention (LSTA) models. The variants are instantiated by using different backbones and pre-trainings. While two-stream EgoACO is the best individual model, additional improvement was achieved using an ensemble.

GT-WISC-MPI (Rank 4 - S1, Rank 2 - S2) employs a 3D CNN where intermediate outputs of the network are used as input to a probabilistic attention module, which models attention using a linear mapping of the video features. An attention map is sampled from the predicted distribution and video features are pooled using this attention map.

G-Blend (Rank 5 - S1, Rank 4 - S2) is an audio-visual approach where RGB and audio are late-fused with concatenation. A channel-separated convolutional network (CSN) is used for video and a residual network is used for audio. Different losses are used for each modality and recalibrated using gradiend-blending, which estimates per-modality weights based on the amount of overfitting in each modality.

VPULab (Rank 39 - S1, Rank 39 - S2) this method filters-out ego-motion from the video. Camera motion is estimated for the whole sequence, and used to divide the video in

	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	**NUS_CVML**	18	05/29/20	**37.87**	**24.10**	**16.64**	**79.74**	**53.98**	**36.06**	**36.41**	**25.20**	**9.64**	**15.67**	**22.01**	**10.05**
S1	2	**VI-I2R**	28	05/23/20	**36.72**	**24.61**	**16.02**	**80.39**	**54.90**	**37.11**	**31.03**	**26.02**	**8.68**	**15.28**	**22.03**	**8.70**
S1	3	**Ego-OMG**	16	05/26/20	**32.20**	**24.90**	**16.02**	**77.42**	**50.24**	**34.53**	**14.92**	**23.25**	**4.03**	**15.48**	**19.16**	**5.36**
S1	4	UNIPD-UNICT	16	05/26/20	36.73	24.26	15.67	79.87	53.76	36.31	35.86	25.16	7.42	14.12	21.30	7.62
S1	5	GT-WISC-MPI	20	11/12/19	36.25	23.83	15.42	79.15	51.98	34.29	24.90	24.03	6.93	15.31	21.91	7.88
S1	6	**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**
S1		**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**
S1		**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**
S2	1	**Ego-OMG**	16	05/26/20	**27.42**	**17.65**	**11.81**	**68.59**	**37.93**	**23.76**	**13.36**	**15.19**	**4.52**	**10.99**	**14.34**	**5.65**
S2	2	**VI-I2R**	28	05/23/20	**28.71**	**17.21**	**10.11**	**71.77**	**40.49**	**23.46**	**12.54**	**15.94**	**4.28**	**9.24**	**14.21**	**5.97**
S2	3	**NUS_CVML**	18	05/29/20	**29.50**	**16.52**	**10.04**	**70.13**	**37.83**	**23.42**	**20.43**	**12.95**	**4.92**	**8.03**	**12.84**	**6.26**
S2	5	GT-WISC-MPI	20	11/12/19	29.87	16.80	9.94	71.77	38.96	23.69	15.96	12.02	4.40	9.65	13.51	5.18
S2	6	UNIPD-UNICT	16	05/26/20	28.51	16.59	9.32	71.66	37.97	23.28	13.15	13.26	4.72	7.86	13.77	5.07
S2	7	**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**
S2		**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**
S2		**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**

Table 2: Results on EPIC-KITCHENS-55 Action Anticipation challenge - 1 June 2020

chunks using K-Means. Camera motion is compensated for each chunk separately, and features from each chunk are late-fused to obtain final predictions. Preliminary results in the submitted report showed promise on the authors own split but the experiments did not materialise, with this method performing last on the public leaderboard. The authors wanted to highlight the importance of ego-motion by submitting this technical report.

4. Action Anticipation Challenge

The 2020 edition of the Action Anticipation challenge has been set similarly to the 2019 edition. The instructions given to the participants for the 2020 edition are summarised in Figure 2. For each annotated action segment A_i of temporal bounds $[t_{s_i}, t_{e_i}]$, the participates were asked to predict the action by observing a video segment preceding the start of the action by a fixed anticipation time $\tau_a = 1\ second$. The length of the observed segment τ_o (observation time) could be set arbitrarily by the participants. Submissions observing any visual content appearing after time $t_{s_i} - \tau_a$ as outlined in Figure 2 (right) were deemed to be invalid.

The submissions followed the same format as that of the recognition challenge, i.e., the participants provided recognition scores for verbs, nouns and actions. Results are reported considering both the S1 and S2 test sets. Table 2 shows the results achieved by the participants, along with the public leaderboard rankings. The top-3 submissions are highlighted in bold. Shaded lines reflect the top-3 ranked methods of last edition for direct comparison. The methods have been evaluated using the same metrics as the action recognition challenge. Interestingly, all submissions outperform the top ranked methods of last year's edition. Overall, the submissions have improved over the top scoring method of the 2019 challenges by $+6.87\%$, $+1.97\%$ and $+1.39\%$ for top-1 verb, noun and action on S1 and $+1.5\%$, $+2.12\%$, $+2.69\%$ on S2.

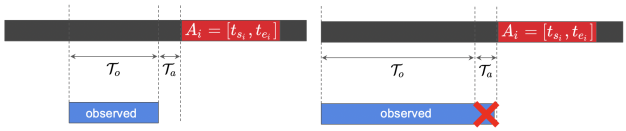


Figure 2: Expected (left) and rejected (right) action anticipation challenge instructions.

We next summarise the contributions of the participants based on their technical reports.

4.1. Technical Reports

Technical reports for the Action Anticipation challenge, in order of their overall rank on the public leaderboard, are:

NUS_CVML (Rank 1 - S1, Rank 3 - S2) The method is based on the analysis of long- and short-term features. Action anticipation is obtained by aggregating such features using computation modules based on non-local blocks - a coupling block to aggregate representations from long- and short-term past representations and a temporal aggregation block to combine the representations to perform anticipation. The proposed approach achieves good results leveraging image representations employed in previous approach, which suggests superior temporal reasoning abilities.

VI-I2R (Rank 2) The method extends Rolling-Unrolling LSTMs by improving the RGB and Flow representations using Temporal Relational Networks (TRN) instead of Temporal Segment Networks (TSM), including additional hand mask features, and incorporating a past action classification module. The final results are obtained considering an ensemble of different instances of the same method.

Ego-OMG (Rank 3 - S1, Rank 1 - S1) The method is based on Egocentric Object Manipulation Graphs (Ego-OMG), a representation proposed for activity modeling and future action prediction. The graph encodes contact and anticipated contact between hands and objects. Hand-object contact is anticipated using a contact anticipation network based on a

	Rank	Team	Submissions		Few Shot Classes (%)			Many Shot Classes (%)			All Classes (%)		
			Entries	Date	IoU >0.05	IoU >0.5	IoU >0.75	IoU >0.05	IoU >0.5	IoU >0.75	IoU >0.05	IoU >0.5 ▲	IoU >0.75
S1	1	**hutom**	51	05/30/20	**47.44**	**35.75**	**14.32**	**60.77**	**46.50**	**15.60**	**58.27**	**44.48**	**15.36**
	2	**DHARI**	27	05/29/20	**54.98**	**32.40**	**14.55**	**68.74**	**43.88**	**15.38**	**66.15**	**41.72**	**15.23**
	3	**FB AI**	69	04/01/20	**26.55**	**19.01**	**8.22**	**58.44**	**46.22**	**15.61**	**52.44**	**41.10**	**14.22**
	4	CVG Lab Uni Bonn	23	05/12/20	39.36	26.66	7.89	53.50	41.28	12.46	50.84	38.53	11.60
	5	VCL	61	05/18/20	33.23	23.16	5.00	50.78	37.91	9.79	47.48	35.13	8.89
	6	[2] (baseline)	-	09/03/18	30.63	20.28	2.75	49.55	37.39	9.82	45.99	34.18	8.49
S2	1	**FB AI**	69	04/01/20	**13.70**	**10.41**	**2.88**	**59.21**	**45.42**	**16.24**	**54.57**	**41.85**	**14.88**
	2	**hutom**	51	05/30/20	**29.81**	**20.87**	**8.09**	**58.66**	**43.42**	**13.00**	**55.72**	**41.12**	**12.50**
	3	**DHARI**	27	05/29/20	**35.75**	**22.31**	**7.33**	**67.92**	**41.92**	**14.29**	**64.64**	**39.93**	**13.58**
	4	CVG Lab Uni Bonn	23	05/12/20	25.34	21.54	7.81	52.18	38.24	11.41	49.45	36.54	11.04
	5	VCL	61	05/18/20	19.87	15.27	4.07	50.37	35.63	9.16	47.26	33.55	8.64
	6	[2] (baseline)	-	09/03/18	20.81	15.88	2.41	47.69	33.84	8.49	44.95	32.01	7.87

Table 3: Results of EPIC-KITCHENS-55 Object Detection in Video challenge - 1 June 2020

3D CNN. Graph convolutions and LSTMs are used to obtain the final prediction.

UNIPD-UNICT (Rank 3 - S1, Rank 6 - S2) The method employs a label smoothing technique which can be used to distill knowledge shared between verbs, nouns, and actions from one-hot labels through the introduction of "soft labels". The proposed approach includes verb-noun label smoothing, glove-based label smoothing and temporal label smoothing. The knowledge distillation technique is applied to the state-of-the-art Rolling-Unrolling LSTM and to an approach to anticipation based on multi-head attention.

GT-WISC-MPI (Rank 4 - S1, Rank 5 - S2) The method considers hand motion and interaction hotspots as features for egocentric action anticipation. The model comprises a backbone 3D CNN, an anticipation module to predict future actions, a motor attention module to anticipate hand trajectories, and an interaction hotspots module to predict interaction regions. The final results are obtained combining the network based on RGB frames with the object branch of Rolling-Unrolling LSTMs.

5. Object Detection in Video challenge

The Object Detection in Video challenge follows similar challenges in object detection [6, 9] where the goal is to localise and classify objects in an image. Unlike previous object recognition challenges, the annotation in EPIC-KITCHENS-55 focuses on 'active' objects where the same object is labelled multiple times while being manipulated. This introduces a temporal aspect to the problem, with dependencies between the annotations, different from object detection from individual images. Objects are labelled at $2fps$ throughout the duration of the action they appear in, as well as ± 2 seconds around the action's temporal bounds.

Since the annotations focus on 'active' objects, the images evaluated per class are restricted to those where the object has been annotated, as inactive object will appear in other images without annotation. Table 3 shows the results of methods submitted to the public leaderboard for both test sets (S1 and S2). Methods are ranked by performance on all classes with IoU > 0.5. The top three submissions are highlighted in bold. While last year's challenge did not see any submissions, likely due to the additional temporal aspect of the problem and computational resources required (see [2]), this year's challenge saw several works beating the Faster-RCNN [8] baseline by a large margin. The top-performing method outperforms the baseline, for all classes and IoU>0.5, by 10% in both S1 and S2. Similar improvement is reported for many-shot classes, however the improvement over few-shot classes varies for the various approaches, particularly for unseen environments (S2).

We next describe the individual contributions of each team based on their technical report.

5.1. Technical Reports

Technical reports for the Object Detection in Video challenge are:

hutom (Rank 1 - S1, Rank 2 - S2) introduces a semi-supervised learning method which uses a bidirectional tracker to generate pseudo labels for frames where the object is not annotated. By using these pseudo labels with a Fully Convolutional One-Stage Object Detection (FCOS) network the proposed method is able to be robust to the sparsity of the annotations.

FB AI (Rank 3 - S1, Rank 1 - S2) uses the current frame's features to warp features from relevant previous and future frames. The aggregation of these warped features with the current frame is then used with Cascade R-CNN to produce object detections. These long-range temporal cues allow the method to mitigate the negative effects caused by motion blur and object occlusions.

DHARI (Rank 2 - S1, Rank 3 - S2) propose the Global Region of Interest (RoI) Extractor to extract RoI features from all levels of a Feature Pyramid Network. A Hard IoU-imbalance Sampling strategy is also used to better sample incorrect bounding boxes as opposed to unlabelled objects. These techniques, and other training tricks such as class-balanced sampling, are used in combination with a Cascade R-CNN.

VCL (Rank 5 - S1, Rank 5 - S2) experiment with various ways to incorporate prior knowledge into existing multi-

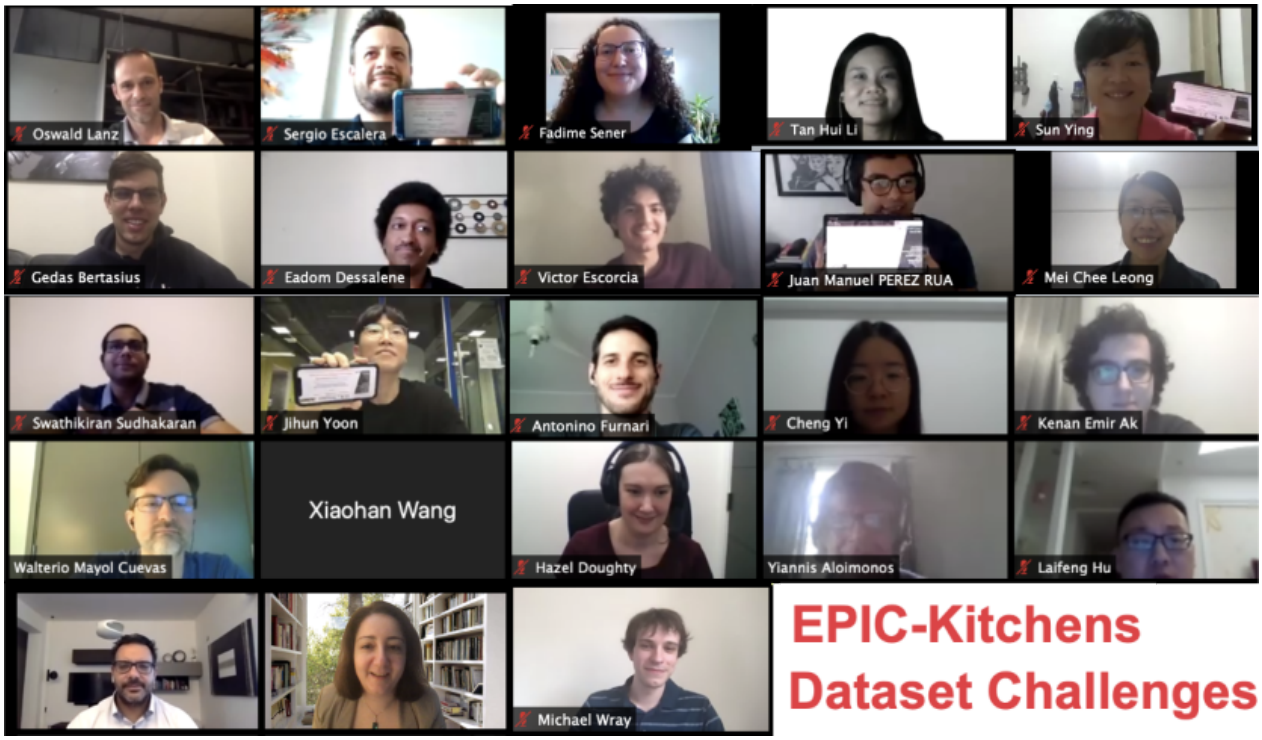


Figure 3: Organisers and team winners during EPIC@CVPR2020 virtual Workshop, 15 June 2020.

	S1	S2	Team	Member	Affiliations
Action Recognition	①	①	UTS-Baidu (wasun)	Xiaohan Wang	University of Technology Sydney, Baidu Research
				Yu Wu	University of Technology Sydney, Baidu Research
				Linchao Zhu	University of Technology Sydney
				Yi Yang	University of Technology Sydney
				Yueting Zhuang	Zhejiang University
	②	③	NUS-CVML (action-banks)	Fadime Sener	University of Bonn
				Dipika Singhania	National University of Singapore
				Angela Yao	National University of Singapore
	④	②	GT-WISC-MPI (aptx4869lm)	Miao Liu	Georgia Institute of Technology
				Yin Li	University of Wisconsin-Madison
				James M. Rehg	Georgia Institute of Technology
	③	⑤	FBK-HUPBA (sudhakran)	Swathikiran Sudhakaran	FBK, University of Trento
				Sergio Escalera	CVC, Universitat de Barcelona
				Oswald Lanz	FBK, University of Trento
	③	⑥	SAIC-Cambridge (tnet)	Juan-Manuel Perez-Rua	Samsung AI Centre, Cambridge
				Antoine Toisoul	Samsung AI Centre, Cambridge
				Brais Martinez	Samsung AI Centre, Cambridge
				Victor Escorcia	Samsung AI Centre, Cambridge
				Li Zhang	Samsung AI Centre, Cambridge
				Xiatian Zhu	Samsung AI Centre, Cambridge
				Tao Xiang	Samsung AI Centre Cambridge, Univ of Surrey
Action Anticipation	①	③	NUS-CVML (action-banks)	Fadime Sener	University of Bonn
				Dipika Singhania	National University of Singapore
				Angela Yao	National University of Singapore
	②	①	Ego-OMG (edessale)	Eadom Dessalene	University of Maryland, College Park
				Michael Maynord	University of Maryland, College Park
				Chinmaya Devaraj	University of Maryland, College Park
				Cornelia Fermuller	University of Maryland, College Park
				Yiannis Aloimonos	University of Maryland, College Park
	②	②	VI-I2R (chengyi)	Ying Sun	A*STAR, Singapore
				Yi Cheng	A*STAR, Singapore
				Mei Chee Leong	A*STAR, Singapore
				Hui Li Tan	A*STAR, Singapore
				Kenan E. Ak	A*STAR, Singapore
Object Detection in Video	①	②	hutom (killerchef)	Jihun Yoon	hutom
				Seungbum Hong	hutom
				Sanha Jeong	hutom
				Min-Kook Choi	hutom
	③	①	FB AI (gb7)	Gedas Bertasius	Facebook AI
				Lorenzo Torresani	Facebook AI
	②	③	DHARI (kide)	Kaide Li	ZheJiang Dahua Technology
				Bingyan Liao	ZheJiang Dahua Technology
				Laifeng Hu	ZheJiang Dahua Technology
				Yaonong Wang	ZheJiang Dahua Technology

Table 4: Top-3 Winners - 2020 EPIC-KITCHENS-55 challenges

stage object detection methods, such as Reasoning R-CNN. Using the frequency of object co-occurrences and the overlap of different objects as prior knowledge gives a modest improvement in results.

6. 2020 Challenge Winners

Accordingly, Table 4 details the winners of the 2020 EPIC challenges, announced as part of EPIC@CVPR2020 virtual workshop. A zoom capture of the 2020 challenges teams and winners is also in Fig 3.

References

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

[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. 4

[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. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2020. 1

[4] D. Damen, W. Price, E. Kazakos, A. Furnari, and G. M. Farinella. Epic-kitchens - 2019 challenges report. Technical report, 2019. 1

[5] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *ICCV*, 2019. 2

[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 *European conference on computer vision*, pages 740–755. Springer, 2014. 4

[7] J.-M. Perez-Rua, B. Martinez, X. Zhu, A. Toisoul, V. Escorcia, and T. Xiang. Knowing what, where and when to look: Efficient video action modeling with attention, 2020. 2

[8] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 4

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 4

[10] 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

Symbiotic Attention: UTS-Baidu Submission to the EPIC-Kitchens 2020 Action Recognition Challenge

Xiaohan Wang[1,2*], Yu Wu[1,2*], Linchao Zhu[1], Yi Yang[1], Yueting Zhuang[3]
{xiaohan.wang-3,yu.wu-3}@student.uts.edu.au;
{linchao.zhu,yi.yang}@uts.edu.au; yzhuang@zju.edu.cn
[1]The ReLER lab, University of Technology Sydney, [2]Baidu Research, [3]Zhejiang University

Abstract

In this report, we describe the technical details of our solution to the EPIC-Kitchens Action Recognition Challenge 2020. The EPIC-Kitchens dataset contains various small objects, intense motion blur, and occlusions. We tackle the egocentric action recognition task by suppressing background distractors and enhancing action-relevant interaction. First, we take candidate objects information to enable concentration on the occurring interactions. Second, we leverage a symbiotic attention mechanism with object-centric alignment to encourage the mutual interaction between the two branches and select the most action-relevant candidates for classification. Third, we incorporate multiple modality inputs, i.e.*, RGB frames and optical flows, to further improve the performance by a multi-modal fusion. Our model ranked the first on both the seen and unseen test set on EPIC-Kitchens Action Recognition Challenge 2020. The code for our model will be available at* https://github.com/wxh1996/SAP-EPIC

1. Introduction

Egocentric action recognition provides a uniquely naturalistic insight into how a person or an agent interacts with the world, which requires distinguishing the object that human is interacting with from various small distracting objects. In EPIC-Kitchens [4], due to the large action vocabulary, researchers [4, 14, 5] usually decouple actions into verbs and nouns, and then further train separate CNN models for the verb classification and noun classification, respectively. The verb branch focuses on classifying actions (verbs) that the actor is performing, e.g., put and open, while the noun branch is to identify the object which the actor is interacting with. The predictions from the two branches are usually directly merged without further interactions for action classification in previous works [4, 14, 5]. However, these works ignore the mutual relation between the standalone branches.

Recently, Wang *et al.* [13] introduced a novel Symbiotic Attention with Object-centric Alignment (SAOA) framework for egocentric video recognition. SAOA better exploits the benefits of the interactions among different sources, enabling mutual communication between the verb and noun branches via object detection features. Our solution to EPIC-Kitchens Action Recognition Challenge 2020 is based on the SAOA framework [13]. Our ensemble SAOA model was ranked fist on both the seen and the unseen test set.

The SAOA framework [13] introduces an object-centric feature alignment method to dynamically integrate location-aware information to the verb and the noun branches. SAOA extends symbiotic attention with privileged information (SAP) [12] by introducing the local-alignment method for the verb branch and evaluating more backbones and input modalities. The object-centric feature alignment encourages the meticulous reasoning between the actor and the environment. The object features and locations are extracted by an object detection model, providing finer local information that is beneficial to the attendance of an on-going action. The noun branch and the verb branch integrate location-aware information by two different approaches, *i.e.*, the global alignment and the local alignment. With the object-centric alignment, we obtain a set of candidate verb features and noun features. The symbiotic attention mechanism [13] is then introduced to enable mutual interactions between the two branches and select the most action-relevant features. It consists of two modules, *i.e.*, cross-stream gating mechanism and action-attended relation module. The SAOA method dynamically integrates three sources of information towards better action recognition.

Our final submission was obtained by an ensemble of the SAOA model [13], the SAP model [12] trained on both RGB and flow modalities. Our results demonstrate that the

*Equal contribution. This work was done when Xiaohan and Yu were interned at Baidu Research.

SAOA model [13] achieves the state-of-the-art performance in action recognition on the EPIC-Kitchens dataset.

2. Our Approach

As illustrated in Fig. 1, the SAOA model [13] includes three stages. First, the location-aware information is integrated into the feature from one branch by the object-centric alignment method. Second, the fused object-centric features are recalibrated by the other branch utilizing a cross-stream gating mechanism. After that, the normalized feature matrix is attended by the other branch to aggregate the most action-relevant information within an action-attended relation module. More details can be found in our paper [13].

2.1. Object-centric Feature Alignment

We decouple the action labels into verbs and nouns, and train two individual 3D CNNs as the backbones in our framework, with one for the verb feature extraction and the other for the noun feature extraction. The object-centric features are extracted by an object detection model, providing finer local information that is beneficial to the attendance of an on-going action. Specifically, we use a pre-trained detection model to provide detailed information of objects in the video. For each video, we use M sampled frames for detection inference. We keep top-K object features and corresponding proposals according to their confidence scores for each sampled frame. The output of the $RoIAlign$ layer of the detection model is regarded as the feature and location for each detected object. The noun branch and the verb branch integrate location-aware information by two different approaches.

Global alignment for noun classification. The noun features and the detection features are complementary to each other, and proper integration of these two features produces more accurate identification of the interacted object. In the global alignment, we concatenate each detection feature with the global noun feature followed by a nonlinear activation. The generated feature matrix incorporates both local relevant features and global contextual features, which restrain the features of irrelevant objects.

Local alignment for verb classification. The verb feature contains motion information, which is quite different from the appearance information in noun feature and object features. Thus, we integrate spatially-aligned verb features with object features. In this way, the most relevant verb features will be generated for better alignment with local object features. It eases the difficulties of the integration between verb features and local object features. For each object detection feature, we have a corresponding spatial detection location. We extract regional verb features from the verb branch by pooling from the spatial feature map with the given candidate spatial location. The regional motion feature is then combined with the corresponding detection feature. The final motion-object paired feature incorporates local detection features and location-aware motion features.

The fused object-centric feature matrix contains useful local details. However, due to the existence of inaccurate detection regions, there are a few disturbing background noises in the features. To address this problem, [13] utilized a cross-stream gating mechanism to enhance the interaction between the verb stream and the noun stream. Furthermore, [13] proposed an action-attended relation module to underline the action relevant information.

2.2. Cross-Stream Gating

Taking the noun classification as an example, for an input noun feature matrix, we use the global verb feature to generate gating weights for it. The output features are produced by re-scaling the noun feature matrix with the gating weights. After re-calibrating the object-centric noun feature by the verb feature, the action-unrelated noise can be suppressed. Moreover, the cross-stream gating mechanism enables mutual communication between the two branches, which adaptively exploits the correlations of verbs and nouns. The detailed formulation of cross-stream gating can be found in [13].

2.3. Action-attended Relation Module

The calibrated object-centric feature matrix contains the action-relevant information and implicit guidance about the spatio-temporal position of an on-going action. To make full use of the information, we consider uncovering the relationships among the features [13]. First, we assess the relevance between the global feature and location-aware object-centric features. Second, we sum the object-centric features weighted by the relevance coefficients. Specifically, we perform attention mechanism on the normalized object-centric noun features and the global verb feature. Through the interaction of global feature and object-centric features, our model selects the most action-relevant feature for classification.

2.4. Action Re-weighting

The actions are determined by the pairs of verb and noun. The primary method of obtaining the action score is to calculate the multiplication of verb and noun probability. However, there are thousands of combinations and most verb-noun pairs that do not exist in reality, *e.g.*, "open knife". In fact, there are only 149 action classes with more than 50 samples in the EPIC-Kitchens dataset [4]. Following the approach in [14], we re-weight the final action probability by the occurrence frequency of action in training set.

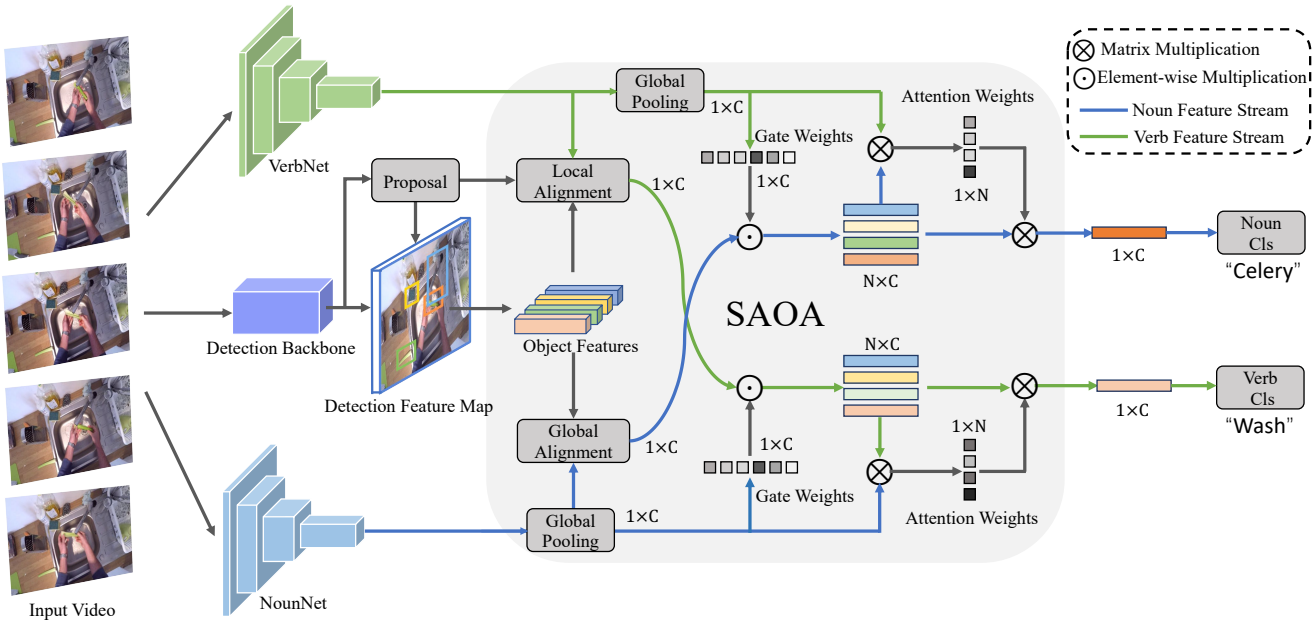


Figure 1. The SAOA framework. The framework consists of three feature extractors and one interaction module. The detection model generates a set of local object features and location proposals. This location-aware information is injected to the two branches by an object-centric alignment method. More details can be found in [13].

Table 1. The results on the EPIC-Kitchens validation set. "Obj" indicates the method leverages the information from the object detection model.

Method	Backbone	Input Type	Pre-training	Actions		Verbs		Nouns	
				top-1	top-5	top-1	top-5	top-1	top-5
SAP [12]	R-50	RGB+Obj	Kinetics	25.0	44.7	55.9	81.9	35.0	60.4
SAP [12]	R-50	Flow+Obj	Kinetics	24.5	43.1	56.1	81.4	33.6	58.7
SAP [12]	R-50	2-Stream	Kinetics	26.6	47.6	59.5	83.1	36.5	62.2
SAP [12]	R(2+1)D-34	RGB+Obj	IG-Kinetics	27.9	47.9	59.1	82.8	38.9	62.2
SAOA [13]	R-50	RGB+Obj	Kinetics	25.7	45.9	57.7	82.3	34.8	59.7
SAOA [13]	I3D	RGB+Obj	Kinetics+ImageNet	24.3	44.3	55.1	80.1	34.7	61.4
SAOA [13]	I3D	Flow+Obj	Kinetics+ImageNet	25.2	43.1	56.9	79.7	35.0	59.7
SAOA [13]	I3D	2-Stream	Kinetics+ImageNet	28.8	48.4	60.4	82.8	37.4	63.8
Ensemble	-	-	-	30.3	50.6	63.4	84.7	40.3	65.6

3. Experiments

3.1. Implementation Details

We followed [12, 13] to train our model. We train the framework in a two-stage optimization scheme. Specifically, we firstly pre-train the base models (VerbNet, NounNet, and the detector) individually. After that, we optimize the subsequent SAOA using extracted features from the base models. Damen *et al.* [3] train the recognition models on EPIC-Kitchens with a dropout layer. This strategy is not used in our models.

Backbone details. We adopt three typical 3D CNNs as our backbones, *i.e.*, ResNet50-3D [6], I3D [2], and R(2+1)D-34 [10].

For ResNet50-3D and I3D, we take the Kinetics [2] pre-trained weights to initialize the backbone. We then train the backbone models (VerbNet and NounNet) individually on the target dataset using 64-frame input clips. The targets for the VerbNet and NounNet are the verb label and noun label, respectively. The videos are decoded at 60 FPS. We adopt the stochastic gradient descent (SGD) with momentum 0.9 and weight decay 0.0001 to optimize the parameters for 35 epochs. The overall learning rate is initialized to 0.003, and then it is changed to 0.0003 in the last 5 epochs. The batch size is 32. During the first training stage, the input frame size is 224×224, and the input frame is randomly cropped from a random scaled video whose side is randomly sampled in [224, 288]. We sample 64 successive frames with stride=2 from each segment to constitute the input clip. The center index of the input clip is randomly chosen in the seg-

Table 2. Results on the leaderboard of EPIC-Kitchens Action Recognition Challenge.

	Method	Top-1 Accuracy			Top-5 Accuracy			Avg Class Precision			Avg Class Recall		
		Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
Seen	Baidu-UTS 2019 [11]	69.80	52.26	41.37	90.95	76.71	63.59	63.55	46.86	25.13	46.94	49.17	26.39
	TBN Single Model [7]	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 [7]	66.10	47.89	36.66	91.28	72.80	58.62	60.74	44.90	24.02	46.82	43.89	22.92
	SAP R-50(RGB)	63.22	48.34	34.76	86.10	71.48	55.91	36.98	41.94	14.60	31.56	45.24	15.94
	SAOA I3D(2-Stream)	67.58	47.79	37.68	89.21	71.83	59.25	57.79	42.13	19.62	42.65	44.75	20.72
	Ensemble w/o IG	70.13	52.49	41.78	90.97	76.71	63.92	60.20	47.38	25.00	45.40	49.57	25.84
	Ensemble	70.41	52.85	42.57	90.78	76.62	63.55	60.44	47.11	24.94	45.82	50.02	26.93
Unseen	Baidu-UTS 2019 [11]	59.68	34.14	25.06	82.69	62.38	45.95	37.20	29.14	15.44	29.81	30.48	18.67
	TBN Single Model [7]	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 [7]	54.46	30.39	20.97	81.23	55.69	39.40	32.57	21.68	10.96	27.60	25.58	13.31
	SAP R-50(RGB)	53.23	33.01	23.86	78.15	58.01	40.53	24.29	28.22	11.02	22.76	28.11	13.72
	SAOA I3D(2-Stream)	58.14	34.38	25.81	82.59	60.40	45.13	38.86	28.69	14.83	28.70	30.06	17.52
	Ensemble w/o IG	60.60	36.09	26.60	83.07	62.89	47.39	40.06	32.09	16.49	29.80	31.80	18.92
	Ensemble	60.43	37.28	27.96	83.06	63.67	46.81	35.23	32.60	17.35	28.97	32.78	19.82

ment during training. For the testing, we sample a center clip per segment. We resize the clip to the size of 256×256 and use a single center crop of 224×224.

For the R(2+1)D-34 backbone training, we use the weights pre-trained on IG-Kinetics-65M [5] as the initialization. The input frames are kept as 32 with stride=4 due to the large GPU memory cost of R(2+1)D-34. We train the model for 20 epochs. The learning rate is initialized to 0.0002 and then decayed by a factor of 0.1 after 9 and 18 epochs. The input size is 112×112 pixels randomly cropped from frames whose side is randomly sampled in [112, 144]. During the second-stage training and the final testing, the input size is 128×128 without cropping.

Detector details. Following [14, 13], we use the same Faster R-CNN to detect objects and extract object features. The detector is first pre-trained on Visual Genome [8] and then fine-tuned on the training split of the EPIC-Kitchens dataset. We use SGD optimizer to train the model with momentum 0.9 and weight decay 0.0001. We use a batch size of 12 and train the model on EPIC-Kitchens for 180k iterations for the trainval/test split. We use an initial learning rate of 0.005, which is decreased by a factor of 10 at iteration 140k and 160k. For the train/val split, we train the model for 150k iterations, and the learning rate decays at iteration 116k and 133k. Finally, our object features are extracted using *RoIAlign* from the detector's feature maps. For each video clip, we perform object detection on a set of frames that are sampled around the clip center within a fixed time duration. The time duration is set to 6 seconds for global alignment and 4 seconds for local alignment. The sample rate is at two frames per second. For each frame, we keep the top five features and proposals according to the confidence scores. Therefore, given a video clip, we obtain 60 detection features during global alignment. In local alignment, we obtain 40 detection features and corresponding locations.

SAOA details. We leverage the pre-trained backbone models and the detection models as the feature extractors. During the second-stage training, only the weights of SAOA are updated. We use SGD with momentum 0.9 and weight decay 0.0001 to optimize the parameters with batch-size of 32. For the model equipped with the I3D backbone, we train the model for 15 epochs. The learning rate is initialized to 0.001 and then reduced to 0.0001 in the last 5 epochs. For the models based on R-50, we train the model for 15 epochs, and the learning rate is set to a constant value 0.0001. Notably, since the detection features have different scales from the I3D features, the features from the I3D backbone need to be normalized before concatenation with detection features in the alignment modules. However, the feature from the R-50 backbone can be directly fed to the SAOA module without normalization. The main reason is the different network types between the detection backbones (based on residual block) and the I3D model (based on Inception block). Specifically, the features produced by the I3D backbone and detection model are l_2-normalized before concatenation. The combined feature is then multiplied by the l_2-norm of the I3D feature to scale the amplitude. A similar normalization strategy is introduced in [9]. During the training and testing of SAOA, we utilize the same temporal sampling strategy during the training and testing of the backbone. For each input video clip, we resize it to the size of 256. Then we feed the 64-frame clip to the network without spatial cropping. More training details of SAOA can be found in [13].

3.2. Results

We train three backbones and two models (SAP and SAOA) on the EPIC-Kitchens dataset. Following [1], we split the original training set of EPIC-Kitchens into the new train and validation set. The results of different models on the validation set are shown in Table 1. "2-Stream"

indicates the results obtained by fusing the predictions of the "RGB+Obj" model and the "Flow+Obj" model. Our 2-Stream SAOA based on the I3D backbone achieves the highest performance compared to other models without ensemble. This shows that our 2-Stream SAOA framework is capable of integrating benefits from both RGB and Flow input. Our SAOA R-50 (RGB) achieves higher verb top-1 accuracy than SAP R-50 (RGB) by 1.8%. This demonstrates the effectiveness of the local alignment method for the verb branch. "Ensemble" indicates the result obtained by fusing the predictions of the above model. The ensemble improves the top-1 action accuracy by 0.9% over the SAOA I3D (2-stream) model.

The results on the test seen set and unseen set are shown in Table 2. Our single model SAOA I3D (2-stream) outperforms the ensemble of TBN [7]. The best result is achieved by "Ensemble" that fuses the predictions of all models (trained on the entire training set) in Table 1. we also show the result of the ensemble ("Ensemble w/o IG"), which fuses the predictions of all other models except the SAP R(2+1)D-34 model. The SAP R(2+1)D-34 model is first pre-trained on the IG-Kinetics dataset and then fine-tuned on EPIC-Kitchens. We observe that pre-training on such a large-scale video dataset (in the format of verb-noun labeling) clearly boosts the noun classification on the unseen set. The ensemble ("Ensemble") is ranked first on both seen and unseen set in the EPIC-Kitchens Action Recognition Challenge 2020.

4. Conclusion

In this report, we described the model details of SAP and SAOA. We introduced the object features and locations to enable concentration on the occurring actions. Moreover, we utilize the symbiotic attention mechanism to discriminate interactions in the egocentric videos. We reported the results of the two models with different input modalities and backbones. Our method achieved state-of-the-art on the EPIC-Kitchens dataset.

References

[1] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. Object level visual reasoning in videos. In *ECCV*, 2018.

[2] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[3] Dima Damen, Hazel Doughty, Giovanni Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE T-PAMI*, 2020.

[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] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *CVPR*, 2019.

[6] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*, 2018.

[7] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *ICCV*, 2019.

[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] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *ECCV*, 2018.

[10] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018.

[11] Xiaohan Wang, Yu Wu, Linchao Zhu, and Yi Yang. Baiduuts submission to the epic-kitchens action recognition challenge 2019. In *CVPR-W*, 2019.

[12] Xiaohan Wang, Yu Wu, Linchao Zhu, and Yi Yang. Symbiotic attention with privileged information for egocentric action recognition. In *AAAI*, 2020.

[13] Xiaohan Wang, Linchao Zhu, Yu Wu, and Yi Yang. Symbiotic attention for egocentric action recognition with object-centric alignment. *In Submission, available at* https://drive.google.com/file/d/1v5Oq_QO5q0zlXdTXpsHY9t1hp7jYiXLC.

[14] 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.

EPIC-Kitchens Egocentric Action Recognition Challenge 2020
User "action_banks" Team "NUS_CVML" Technical Report

Fadime Sener[1,2], Dipika Singhania[2], Angela Yao[2]
[1]University of Bonn, Germany
[2]National University of Singapore
sener@cs.uni-bonn.de, dipika16@comp.nus.edu.sg, ayao@comp.nus.edu.sg

Abstract

In this technical report, we describe the Team "NUS_CVML" approach to the EPIC-Kitchens Egocentric Action Recognition Challenge 2020. Action recognition requires reasoning from current observations and the past and the future context in video and raises several fundamental questions. How should the observations and long-term context and their sequential relationships be modelled? What temporal extent of context needs to be processed? At what temporal scale should they be derived? We address these questions with a flexible multi-granular temporal aggregation framework. We show that it is possible to achieve competitive results on egocentric action recognition using simple techniques such as max-pooling and attention.

1. Introduction

In this report, we tackle long-term video understanding, specifically classification of trimmed segments in long videos to action classes. Motivated by the questions of temporal modelling, extent, and scaling, we propose a general framework for encoding long-term video. We split video streams into snippets of equal length and max-pool the frame features within the snippets. We then create ensembles of multi-scale feature representations that are aggregated bottom-up based on scaling and temporal extent. The model we used for this challenge is described in detail in [7], and we refer the reader to this paper for further details of our model, including ablation studies and evaluations on other datasets.

2. Representations

We begin by introducing the representations, which are inputs to the building blocks of our framework, see Fig. 1. We had two rationales when designing our network. First, the coupling blocks relate recent observations to long-range future and past context, since some actions directly determine what current actions can or cannot be. Second, to represent recent and long-term context at various granularities, we pool snippets over multiple scales.

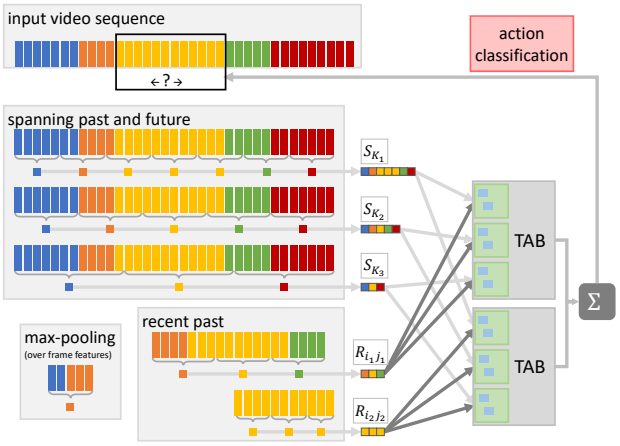


Figure 1. Model overview: In this example we use 3 scales for computing the "spanning context" snippet features $\mathbf{S}_{K_1}, \mathbf{S}_{K_2}, \mathbf{S}_{K_3}$, and 2 "recent" snippet features, $\mathbf{R}_{i_1,j_1}, \mathbf{R}_{i_2,j_2}$, by max-pooling over the frame features in each snippet. Each recent snippet is coupled with all the spanning snippets in our Temporal Aggregation Block (TAB). An ensemble of TAB outputs is used for action recognition. Best viewed in color.

2.1. Pooling

For a video of length T, we denote the feature representation of a single video frame indexed at time t as $f_t \in \mathbb{R}^D, 1 \leq t \leq T$. f_t can be derived from low-level features, such as I3D [2], or high-level abstractions, such as sub-activity labels derived from temporal action segmentation algorithms. To reduce computational load, we work at a snippet-level instead of at the frame level. We define a snippet feature $\mathbf{F}_{ij;K}$ as the concatenation of max-pooled features from K snippets, where snippets are partitioned consecutively from frames starting at i and ending at j:

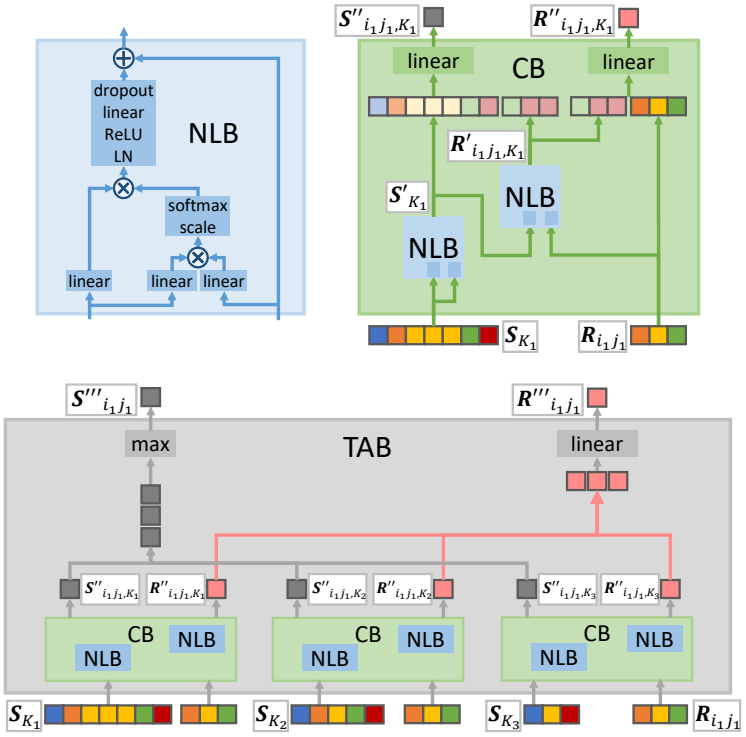


Figure 2. Overview of model components: Non-Local Blocks (NLB) compute interactions between two representations via attention (see Sec. 3.1). Two such NLBs are combined in a Coupling Block (CB) which calculates the attention-reweighted recent and spanning context representations (see Sec. 3.2). We couple each recent with all spanning representations via individual CBs and combine their outputs in a Temporal Aggregation Block (TAB) (see Sec. 3.3). The outputs of multiple such TABs are combined to perform classification through ensembles of multiple recents, see Fig. 1. Best viewed in color.

$$\mathbf{F}_{ij;K} = [F_{i,i+k}, F_{i+k+1,i+2k}, ..., F_{j-k+1,j}], \text{ where}$$
$$(F_{p,q})_d = \max_{p \le t \le q} \{f_t\}_d, \; 1 \le d \le D, \;\; k = (j-i)/K. \quad (1)$$

Here, $F_{p,q}$ indicates the maximum over each dimension d of the frame features in a given snippet between frames p and q, though it can be substituted with other alternatives.

2.2. Recent vs. Spanning Representations

Based on different start and end frames i and j and number of snippets K, we define two types of snippet features: *"recent"* features $\{\mathcal{R}\}$ from the action segment and *"spanning"* features $\{\mathcal{S}\}$ drawn from the longer past and future context around the action boundary. The recent snippets cover the action segment and several seconds around the action boundary to combat the annotation error. Spanning snippets refer to the longer context to make use of long-term sequence information for identifying the action. For *"recent"* snippets, the number of snippets is fixed to K_R. Recent snippet features $\mathcal{R}$ can be defined as a feature bank of snippet features with different start and end frames i, j, *i.e.*

$$\begin{aligned}\mathcal{R} &= \{\mathbf{F}_{i_1 j_1;K_R}, \mathbf{F}_{i_2 j_2;K_R}, ..., \mathbf{F}_{i_R j_R;K_R}\} \\ &= \{\mathbf{R}_{i_1 j_1}, \mathbf{R}_{i_2 j_2}, ..., \mathbf{R}_{i_R j_R}\},\end{aligned} \quad (2)$$

where $\mathbf{R}_{i,j} \in \mathbb{R}^{D \times K_R}$ is a shorthand to denote $\mathbf{F}_{ij;K_R}$, since the number of snippets K_R are fixed. In Fig. 1 we use two pairs of starting and ending points, (i_1, j_1) and (i_2, j_2), to compute the "recent" snippet features and represent each with $K_R = 3$ number of snippets (&).

For *"spanning"* snippets, start and end frames, s and e, are fixed to a long context window around action boundary, *i.e.* $s = i - c, e = j + c$. Spanning snippet features $\mathcal{S}$ are defined as a feature bank of snippet features with varying number of snippets K, *i.e.*

$$\begin{aligned}\mathcal{S} &= \{\mathbf{F}_{s\,e;K_1}, \mathbf{F}_{s\,e;K_2}, ..., \mathbf{F}_{s\,e;K_S}\} \\ &= \{\mathbf{S}_{K_1}, \mathbf{S}_{K_2}, ..., \mathbf{S}_{K_S}\},\end{aligned} \quad (3)$$

where $\mathbf{S}_K \in \mathbb{R}^{D \times K}$ is a shorthand for $\mathbf{F}_{e\,s;K}$. In Fig. 1 we use three scales to compute the "spanning context" snippet features with $K = \{7, 5, 3\}$ (, &).

Key to both types of representations is the ensemble of snippet features from multiple scales. We achieve this by

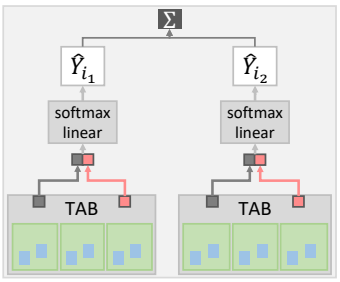


Figure 3. Prediction model for action recognition.

varying the number of snippets K for the spanning context. For recent, it is sufficient to keep the number of snippets K_R fixed, and vary only the start and end points i, j, due to redundancy between $\mathcal{R}$ and $\mathcal{S}$ for the snippets that overlap.

3. Framework

In Fig. 2 we present an overview of the components used in our framework, which we build in a bottom up manner, starting with the recent and spanning features $\mathcal{R}$ and $\mathcal{S}$, which are coupled with non-local blocks (NLB) (Sec. 3.1) within coupling blocks (CB) (Sec. 3.2). The outputs of the CBs from different scales are then aggregated inside temporal aggregation blocks (TAB) (Sec. 3.3). Outputs of different TABs can then be chained together for action recognition. (Sec. 4.1).

3.1. Non-Local Blocks (NLB)

We apply non-local operations to capture relationships amongst the spanning snippets and between spanning and recent snippets. Non-local blocks [10] are a flexible way to relate features independently from their temporal distance and thus capture long-range dependencies. We use the modified non-local block from [11], which adds layer normalization [1] and dropout [8] to the original one in [9]. Fig. 2 (left) visualizes the architecture of the block, the operation of which we denote as $\text{NLB}(\cdot,\cdot)$.

3.2. Coupling Block (CB)

Based on the NLB, we define attention-reweighted spanning and recent outputs as:

$$\mathbf{S}'_K = NLB(\mathbf{S}_K, \mathbf{S}_K) \quad \text{and} \quad \mathbf{R}'_{ij,K} = NLB(\mathbf{S}'_K, \mathbf{R}_{ij}). \tag{4}$$

The coupling is done by concatenating $\mathbf{R}'_{ij,K}$ with either $\mathbf{R}_{ij}$ or $\mathbf{S}'_K$ and passed through linear layers. This results in the fixed-length representations $\mathbf{R}''_{ij,K}$ and $\mathbf{S}''_{ij,K}$, where i is the starting point of the recent snippet, j is ending point of the recent snippet and K is the scale of the spanning snippet.

3.3. Temporal Aggregation Block (TAB)

The final representation for recent and spanning context is computed by aggregating outputs from multiple CBs. For the same recent, we concatenate $\mathbf{R}''_{ij,K_1}, ..., \mathbf{R}''_{ij,K_S}$ for all spanning scales and pass the concatenation through a linear layer to compute $\mathbf{R}'''_{ij}$. The final spanning context representation $\mathbf{S}'''_{ij}$ is a max over all $\mathbf{S}''_{ij,K_1}, ..., \mathbf{S}''_{ij,K_S}$. We empirically find that taking the max outperforms other alternatives like linear layers and/or concatenation for the spanning context.

TAB outputs, by varying recent starting points $\{i\}$ and ending points $\{j\}$ and scales of spanning snippets $\{K\}$, are multi-granular video representations that aggregate and encode both the recent and long-term context. We name these **temporal aggregate representations**. Fig. 1 shows an example with 2 recent starting points and 3 spanning scales.

3.4. Prediction Model

For action recognition task temporal aggregate representations can be used directly with a classification layer (linear + softmax). A cross-entropy loss based on ground truth labels Y can be applied to the predictions $\hat{Y}_{i,j}$, where Y is the action label for recognition, see Fig. 3.

Our final loss formulation is the sum of the cross entropy's over the actions:

$$\mathcal{L}_{\text{cl}} = -\sum_{r=1}^{R}\sum_{n=1}^{N_Y} Y_n \log(\hat{Y}_{i_r})_n, \tag{5}$$

where i_r is one of the R recent starting points, and N_Y is the total number of actions. During inference, the predicted scores are summed for a final prediction, *i.e.* $\hat{Y} = \max_n(\sum_{r=1}^{R} \hat{Y}_{i_r})_n$.

3.5. Implementation Details

We train our model using the Adam optimizer [6] with batch size 10, learning rate 10^{-4} and dropout rate 0.3. We train for 25 epochs and decrease the learning rate by a factor of 10 every 10^{th} epoch. We use 512 dimensions for all non-classification linear layers.

4. Results

Features We use the appearance (RGB), motion (optical flow) and object-based features provided by Furnari and Farinella [5]. They independently train the spatial and motion CNNs using the TSN [9] framework for action recognition on EPIC-Kitchens. They also train object detectors to recognize the 352 object classes of the EPIC-Kitchens dataset. The feature dimensions are 1024, 1024 and 352 for appearance, motion and object features respectively.

Parameters: The spanning scales $\{K\}$, recent scale K_R and recent starting points $\{i\}$ are given in Table 1. In our

# classes	# segments	$\{i, j\}$(in seconds)	spanning scope (s)	K_R	$\{K\}$
2513	39.6K	$\{(i, j), (i-1, j+1), (i-2, j+2), (i-3, j+3)\}$	$s = i - 6, e = j + 6$	5	$\{2, 3, 5\}$

Table 1. Dataset details and our respective model parameters.

	Top-1 Accuracy%			Top-5 Accuracy%			Precision (%)			Recall (%)		
	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
S1	66.56	49.60	**41.59**	90.10	77.03	64.11	59.43	45.62	25.37	41.65	46.25	26.98
S2	54.56	33.46	**26.97**	80.40	60.98	46.43	33.60	30.54	14.99	25.28	28.39	17.97

Table 2. Action recognition on EPIC tests sets, seen (S1) and unseen (S2)

work, we predict the action classes directly rather than predicting the verbs and nouns independently [3]. Directly predicting actions is shown to outperform the latter [4]. We use a validation set provided by Furnari and Farinella [5] for selecting EPIC-Kitchens parameters.

4.1. Recognition on EPIC-Kitchens

We train our model separately for each feature modality (appearance, motion and object) with the same parameters in Table 1; during inference we apply a late fusion of the predictions from the different modalities by average voting. We report our results for hold-out test data on EPIC-Kitchens Egocentric Action Recognition Challenge (2020) in Table 2 for seen kitchens (S1) with the same environments as in the training data and unseen kitchens (S2) of held out environments.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] 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.

[3] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[4] Antonino Furnari, Sebastiano Battiato, and Giovanni Maria Farinella. Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[5] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *International Conference on Computer Vision (ICCV)*, 2019.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Fadime Sener, Dipika Singhania, and Angela Yao. Temporal aggregate representations for long term video understanding. *arXiv preprint arXiv:2006.00830*, 2020.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[9] 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.

[10] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7794–7803, 2018.

[11] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krähenbühl, and Ross Girshick. Long-Term Feature Banks for Detailed Video Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Symbiotic Attention: UTS-Baidu Submission to the EPIC-Kitchens 2020 Action Recognition Challenge

Xiaohan Wang[1,2*], Yu Wu[1,2*], Linchao Zhu[1], Yi Yang[1], Yueting Zhuang[3]
{xiaohan.wang-3,yu.wu-3}@student.uts.edu.au;
{linchao.zhu,yi.yang}@uts.edu.au; yzhuang@zju.edu.cn
[1]The ReLER lab, University of Technology Sydney, [2]Baidu Research, [3]Zhejiang University

Abstract

In this report, we describe the technical details of our solution to the EPIC-Kitchens Action Recognition Challenge 2020. The EPIC-Kitchens dataset contains various small objects, intense motion blur, and occlusions. We tackle the egocentric action recognition task by suppressing background distractors and enhancing action-relevant interaction. First, we take candidate objects information to enable concentration on the occurring interactions. Second, we leverage a symbiotic attention mechanism with object-centric alignment to encourage the mutual interaction between the two branches and select the most action-relevant candidates for classification. Third, we incorporate multiple modality inputs, i.e.*, RGB frames and optical flows, to further improve the performance by a multi-modal fusion. Our model ranked the first on both the seen and unseen test set on EPIC-Kitchens Action Recognition Challenge 2020. The code for our model will be available at* https://github.com/wxh1996/SAP-EPIC

1. Introduction

Egocentric action recognition provides a uniquely naturalistic insight into how a person or an agent interacts with the world, which requires distinguishing the object that human is interacting with from various small distracting objects. In EPIC-Kitchens [4], due to the large action vocabulary, researchers [4, 14, 5] usually decouple actions into verbs and nouns, and then further train separate CNN models for the verb classification and noun classification, respectively. The verb branch focuses on classifying actions (verbs) that the actor is performing, e.g., put and open, while the noun branch is to identify the object which the actor is interacting with. The predictions from the two branches are usually directly merged without further interactions for action classification in previous works [4, 14, 5]. However, these works ignore the mutual relation between the standalone branches.

Recently, Wang *et al.* [13] introduced a novel Symbiotic Attention with Object-centric Alignment (SAOA) framework for egocentric video recognition. SAOA better exploits the benefits of the interactions among different sources, enabling mutual communication between the verb and noun branches via object detection features. Our solution to EPIC-Kitchens Action Recognition Challenge 2020 is based on the SAOA framework [13]. Our ensemble SAOA model was ranked fist on both the seen and the unseen test set.

The SAOA framework [13] introduces an object-centric feature alignment method to dynamically integrate location-aware information to the verb and the noun branches. SAOA extends symbiotic attention with privileged information (SAP) [12] by introducing the local-alignment method for the verb branch and evaluating more backbones and input modalities. The object-centric feature alignment encourages the meticulous reasoning between the actor and the environment. The object features and locations are extracted by an object detection model, providing finer local information that is beneficial to the attendance of an on-going action. The noun branch and the verb branch integrate location-aware information by two different approaches, *i.e.*, the global alignment and the local alignment. With the object-centric alignment, we obtain a set of candidate verb features and noun features. The symbiotic attention mechanism [13] is then introduced to enable mutual interactions between the two branches and select the most action-relevant features. It consists of two modules, *i.e.*, cross-stream gating mechanism and action-attended relation module. The SAOA method dynamically integrates three sources of information towards better action recognition.

Our final submission was obtained by an ensemble of the SAOA model [13], the SAP model [12] trained on both RGB and flow modalities. Our results demonstrate that the

*Equal contribution. This work was done when Xiaohan and Yu were interned at Baidu Research.

SAOA model [13] achieves the state-of-the-art performance in action recognition on the EPIC-Kitchens dataset.

2. Our Approach

As illustrated in Fig. 1, the SAOA model [13] includes three stages. First, the location-aware information is integrated into the feature from one branch by the object-centric alignment method. Second, the fused object-centric features are recalibrated by the other branch utilizing a cross-stream gating mechanism. After that, the normalized feature matrix is attended by the other branch to aggregate the most action-relevant information within an action-attended relation module. More details can be found in our paper [13].

2.1. Object-centric Feature Alignment

We decouple the action labels into verbs and nouns, and train two individual 3D CNNs as the backbones in our framework, with one for the verb feature extraction and the other for the noun feature extraction. The object-centric features are extracted by an object detection model, providing finer local information that is beneficial to the attendance of an on-going action. Specifically, we use a pre-trained detection model to provide detailed information of objects in the video. For each video, we use M sampled frames for detection inference. We keep top-K object features and corresponding proposals according to their confidence scores for each sampled frame. The output of the $RoIAlign$ layer of the detection model is regarded as the feature and location for each detected object. The noun branch and the verb branch integrate location-aware information by two different approaches.

Global alignment for noun classification. The noun features and the detection features are complementary to each other, and proper integration of these two features produces more accurate identification of the interacted object. In the global alignment, we concatenate each detection feature with the global noun feature followed by a nonlinear activation. The generated feature matrix incorporates both local relevant features and global contextual features, which restrain the features of irrelevant objects.

Local alignment for verb classification. The verb feature contains motion information, which is quite different from the appearance information in noun feature and object features. Thus, we integrate spatially-aligned verb features with object features. In this way, the most relevant verb features will be generated for better alignment with local object features. It eases the difficulties of the integration between verb features and local object features. For each object detection feature, we have a corresponding spatial detection location. We extract regional verb features from the verb branch by pooling from the spatial feature map with the given candidate spatial location. The regional motion feature is then combined with the corresponding detection feature. The final motion-object paired feature incorporates local detection features and location-aware motion features.

The fused object-centric feature matrix contains useful local details. However, due to the existence of inaccurate detection regions, there are a few disturbing background noises in the features. To address this problem, [13] utilized a cross-stream gating mechanism to enhance the interaction between the verb stream and the noun stream. Furthermore, [13] proposed an action-attended relation module to underline the action relevant information.

2.2. Cross-Stream Gating

Taking the noun classification as an example, for an input noun feature matrix, we use the global verb feature to generate gating weights for it. The output features are produced by re-scaling the noun feature matrix with the gating weights. After re-calibrating the object-centric noun feature by the verb feature, the action-unrelated noise can be suppressed. Moreover, the cross-stream gating mechanism enables mutual communication between the two branches, which adaptively exploits the correlations of verbs and nouns. The detailed formulation of cross-stream gating can be found in [13].

2.3. Action-attended Relation Module

The calibrated object-centric feature matrix contains the action-relevant information and implicit guidance about the spatio-temporal position of an on-going action. To make full use of the information, we consider uncovering the relationships among the features [13]. First, we assess the relevance between the global feature and location-aware object-centric features. Second, we sum the object-centric features weighted by the relevance coefficients. Specifically, we perform attention mechanism on the normalized object-centric noun features and the global verb feature. Through the interaction of global feature and object-centric features, our model selects the most action-relevant feature for classification.

2.4. Action Re-weighting

The actions are determined by the pairs of verb and noun. The primary method of obtaining the action score is to calculate the multiplication of verb and noun probability. However, there are thousands of combinations and most verb-noun pairs that do not exist in reality, *e.g.*, "open knife". In fact, there are only 149 action classes with more than 50 samples in the EPIC-Kitchens dataset [4]. Following the approach in [14], we re-weight the final action probability by the occurrence frequency of action in training set.

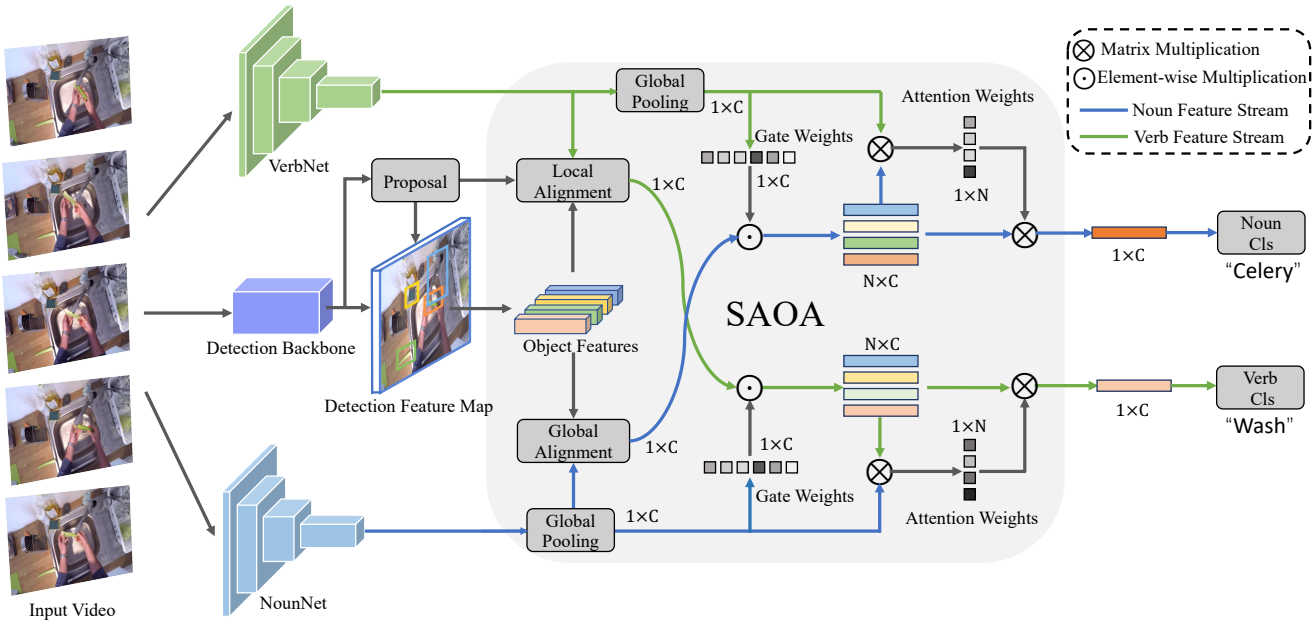


Figure 1. The SAOA framework. The framework consists of three feature extractors and one interaction module. The detection model generates a set of local object features and location proposals. This location-aware information is injected to the two branches by an object-centric alignment method. More details can be found in [13].

Table 1. The results on the EPIC-Kitchens validation set. "Obj" indicates the method leverages the information from the object detection model.

Method	Backbone	Input Type	Pre-training	Actions		Verbs		Nouns	
				top-1	top-5	top-1	top-5	top-1	top-5
SAP [12]	R-50	RGB+Obj	Kinetics	25.0	44.7	55.9	81.9	35.0	60.4
SAP [12]	R-50	Flow+Obj	Kinetics	24.5	43.1	56.1	81.4	33.6	58.7
SAP [12]	R-50	2-Stream	Kinetics	26.6	47.6	59.5	83.1	36.5	62.2
SAP [12]	R(2+1)D-34	RGB+Obj	IG-Kinetics	27.9	47.9	59.1	82.8	38.9	62.2
SAOA [13]	R-50	RGB+Obj	Kinetics	25.7	45.9	57.7	82.3	34.8	59.7
SAOA [13]	I3D	RGB+Obj	Kinetics+ImageNet	24.3	44.3	55.1	80.1	34.7	61.4
SAOA [13]	I3D	Flow+Obj	Kinetics+ImageNet	25.2	43.1	56.9	79.7	35.0	59.7
SAOA [13]	I3D	2-Stream	Kinetics+ImageNet	28.8	48.4	60.4	82.8	37.4	63.8
Ensemble	-	-	-	30.3	50.6	63.4	84.7	40.3	65.6

3. Experiments

3.1. Implementation Details

We followed [12, 13] to train our model. We train the framework in a two-stage optimization scheme. Specifically, we firstly pre-train the base models (VerbNet, NounNet, and the detector) individually. After that, we optimize the subsequent SAOA using extracted features from the base models. Damen *et al.* [3] train the recognition models on EPIC-Kitchens with a dropout layer. This strategy is not used in our models.

Backbone details. We adopt three typical 3D CNNs as our backbones, *i.e.*, ResNet50-3D [6], I3D [2], and R(2+1)D-34 [10].

For ResNet50-3D and I3D, we take the Kinetics [2] pre-trained weights to initialize the backbone. We then train the backbone models (VerbNet and NounNet) individually on the target dataset using 64-frame input clips. The targets for the VerbNet and NounNet are the verb label and noun label, respectively. The videos are decoded at 60 FPS. We adopt the stochastic gradient descent (SGD) with momentum 0.9 and weight decay 0.0001 to optimize the parameters for 35 epochs. The overall learning rate is initialized to 0.003, and then it is changed to 0.0003 in the last 5 epochs. The batch size is 32. During the first training stage, the input frame size is 224×224, and the input frame is randomly cropped from a random scaled video whose side is randomly sampled in [224, 288]. We sample 64 successive frames with stride=2 from each segment to constitute the input clip. The center index of the input clip is randomly chosen in the seg-

Table 2. Results on the leaderboard of EPIC-Kitchens Action Recognition Challenge.

	Method	Top-1 Accuracy			Top-5 Accuracy			Avg Class Precision			Avg Class Recall		
		Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
Seen	Baidu-UTS 2019 [11]	69.80	52.26	41.37	90.95	76.71	63.59	63.55	46.86	25.13	46.94	49.17	26.39
	TBN Single Model [7]	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 [7]	66.10	47.89	36.66	91.28	72.80	58.62	60.74	44.90	24.02	46.82	43.89	22.92
	SAP R-50(RGB)	63.22	48.34	34.76	86.10	71.48	55.91	36.98	41.94	14.60	31.56	45.24	15.94
	SAOA I3D(2-Stream)	67.58	47.79	37.68	89.21	71.83	59.25	57.79	42.13	19.62	42.65	44.75	20.72
	Ensemble w/o IG	70.13	52.49	41.78	90.97	76.71	63.92	60.20	47.38	25.00	45.40	49.57	25.84
	Ensemble	70.41	52.85	42.57	90.78	76.62	63.55	60.44	47.11	24.94	45.82	50.02	26.93
Unseen	Baidu-UTS 2019 [11]	59.68	34.14	25.06	82.69	62.38	45.95	37.20	29.14	15.44	29.81	30.48	18.67
	TBN Single Model [7]	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 [7]	54.46	30.39	20.97	81.23	55.69	39.40	32.57	21.68	10.96	27.60	25.58	13.31
	SAP R-50(RGB)	53.23	33.01	23.86	78.15	58.01	40.53	24.29	28.22	11.02	22.76	28.11	13.72
	SAOA I3D(2-Stream)	58.14	34.38	25.81	82.59	60.40	45.13	38.86	28.69	14.83	28.70	30.06	17.52
	Ensemble w/o IG	60.60	36.09	26.60	83.07	62.89	47.39	40.06	32.09	16.49	29.80	31.80	18.92
	Ensemble	60.43	37.28	27.96	83.06	63.67	46.81	35.23	32.60	17.35	28.97	32.78	19.82

ment during training. For the testing, we sample a center clip per segment. We resize the clip to the size of 256×256 and use a single center crop of 224×224.

For the R(2+1)D-34 backbone training, we use the weights pre-trained on IG-Kinetics-65M [5] as the initialization. The input frames are kept as 32 with stride=4 due to the large GPU memory cost of R(2+1)D-34. We train the model for 20 epochs. The learning rate is initialized to 0.0002 and then decayed by a factor of 0.1 after 9 and 18 epochs. The input size is 112×112 pixels randomly cropped from frames whose side is randomly sampled in [112, 144]. During the second-stage training and the final testing, the input size is 128×128 without cropping.

Detector details. Following [14, 13], we use the same Faster R-CNN to detect objects and extract object features. The detector is first pre-trained on Visual Genome [8] and then fine-tuned on the training split of the EPIC-Kitchens dataset. We use SGD optimizer to train the model with momentum 0.9 and weight decay 0.0001. We use a batch size of 12 and train the model on EPIC-Kitchens for 180k iterations for the trainval/test split. We use an initial learning rate of 0.005, which is decreased by a factor of 10 at iteration 140k and 160k. For the train/val split, we train the model for 150k iterations, and the learning rate decays at iteration 116k and 133k. Finally, our object features are extracted using *RoIAlign* from the detector's feature maps. For each video clip, we perform object detection on a set of frames that are sampled around the clip center within a fixed time duration. The time duration is set to 6 seconds for global alignment and 4 seconds for local alignment. The sample rate is at two frames per second. For each frame, we keep the top five features and proposals according to the confidence scores. Therefore, given a video clip, we obtain 60 detection features during global alignment. In local alignment, we obtain 40 detection features and corresponding locations.

SAOA details. We leverage the pre-trained backbone models and the detection models as the feature extractors. During the second-stage training, only the weights of SAOA are updated. We use SGD with momentum 0.9 and weight decay 0.0001 to optimize the parameters with batch-size of 32. For the model equipped with the I3D backbone, we train the model for 15 epochs. The learning rate is initialized to 0.001 and then reduced to 0.0001 in the last 5 epochs. For the models based on R-50, we train the model for 15 epochs, and the learning rate is set to a constant value 0.0001. Notably, since the detection features have different scales from the I3D features, the features from the I3D backbone need to be normalized before concatenation with detection features in the alignment modules. However, the feature from the R-50 backbone can be directly fed to the SAOA module without normalization. The main reason is the different network types between the detection backbones (based on residual block) and the I3D model (based on Inception block). Specifically, the features produced by the I3D backbone and detection model are l_2-normalized before concatenation. The combined feature is then multiplied by the l_2-norm of the I3D feature to scale the amplitude. A similar normalization strategy is introduced in [9]. During the training and testing of SAOA, we utilize the same temporal sampling strategy during the training and testing of the backbone. For each input video clip, we resize it to the size of 256. Then we feed the 64-frame clip to the network without spatial cropping. More training details of SAOA can be found in [13].

3.2. Results

We train three backbones and two models (SAP and SAOA) on the EPIC-Kitchens dataset. Following [1], we split the original training set of EPIC-Kitchens into the new train and validation set. The results of different models on the validation set are shown in Table 1. "2-Stream"

indicates the results obtained by fusing the predictions of the "RGB+Obj" model and the "Flow+Obj" model. Our 2-Stream SAOA based on the I3D backbone achieves the highest performance compared to other models without ensemble. This shows that our 2-Stream SAOA framework is capable of integrating benefits from both RGB and Flow input. Our SAOA R-50 (RGB) achieves higher verb top-1 accuracy than SAP R-50 (RGB) by 1.8%. This demonstrates the effectiveness of the local alignment method for the verb branch. "Ensemble" indicates the result obtained by fusing the predictions of the above model. The ensemble improves the top-1 action accuracy by 0.9% over the SAOA I3D (2-stream) model.

The results on the test seen set and unseen set are shown in Table 2. Our single model SAOA I3D (2-stream) outperforms the ensemble of TBN [7]. The best result is achieved by "Ensemble" that fuses the predictions of all models (trained on the entire training set) in Table 1. we also show the result of the ensemble ("Ensemble w/o IG"), which fuses the predictions of all other models except the SAP R(2+1)D-34 model. The SAP R(2+1)D-34 model is first pre-trained on the IG-Kinetics dataset and then fine-tuned on EPIC-Kitchens. We observe that pre-training on such a large-scale video dataset (in the format of verb-noun labeling) clearly boosts the noun classification on the unseen set. The ensemble ("Ensemble") is ranked first on both seen and unseen set in the EPIC-Kitchens Action Recognition Challenge 2020.

4. Conclusion

In this report, we described the model details of SAP and SAOA. We introduced the object features and locations to enable concentration on the occurring actions. Moreover, we utilize the symbiotic attention mechanism to discriminate interactions in the egocentric videos. We reported the results of the two models with different input modalities and backbones. Our method achieved state-of-the-art on the EPIC-Kitchens dataset.

References

[1] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. Object level visual reasoning in videos. In *ECCV*, 2018.

[2] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[3] Dima Damen, Hazel Doughty, Giovanni Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE T-PAMI*, 2020.

[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] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *CVPR*, 2019.

[6] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*, 2018.

[7] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *ICCV*, 2019.

[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] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *ECCV*, 2018.

[10] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018.

[11] Xiaohan Wang, Yu Wu, Linchao Zhu, and Yi Yang. Baiduuts submission to the epic-kitchens action recognition challenge 2019. In *CVPR-W*, 2019.

[12] Xiaohan Wang, Yu Wu, Linchao Zhu, and Yi Yang. Symbiotic attention with privileged information for egocentric action recognition. In *AAAI*, 2020.

[13] Xiaohan Wang, Linchao Zhu, Yu Wu, and Yi Yang. Symbiotic attention for egocentric action recognition with object-centric alignment. *In Submission, available at* https://drive.google.com/file/d/1v5Oq_QO5q0zlXdTXpsHY9t1hp7jYiXLC.

[14] 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.

EPIC-Kitchens Egocentric Action Recognition Challenge 2020
User "action_banks" Team "NUS_CVML" Technical Report

Fadime Sener[1,2], Dipika Singhania[2], Angela Yao[2]
[1]University of Bonn, Germany
[2]National University of Singapore
sener@cs.uni-bonn.de, dipika16@comp.nus.edu.sg, ayao@comp.nus.edu.sg

Abstract

In this technical report, we describe the Team "NUS_CVML" approach to the EPIC-Kitchens Egocentric Action Recognition Challenge 2020. Action recognition requires reasoning from current observations and the past and the future context in video and raises several fundamental questions. How should the observations and long-term context and their sequential relationships be modelled? What temporal extent of context needs to be processed? At what temporal scale should they be derived? We address these questions with a flexible multi-granular temporal aggregation framework. We show that it is possible to achieve competitive results on egocentric action recognition using simple techniques such as max-pooling and attention.

1. Introduction

In this report, we tackle long-term video understanding, specifically classification of trimmed segments in long videos to action classes. Motivated by the questions of temporal modelling, extent, and scaling, we propose a general framework for encoding long-term video. We split video streams into snippets of equal length and max-pool the frame features within the snippets. We then create ensembles of multi-scale feature representations that are aggregated bottom-up based on scaling and temporal extent. The model we used for this challenge is described in detail in [7], and we refer the reader to this paper for further details of our model, including ablation studies and evaluations on other datasets.

2. Representations

We begin by introducing the representations, which are inputs to the building blocks of our framework, see Fig. 1. We had two rationales when designing our network. First, the coupling blocks relate recent observations to long-range future and past context, since some actions directly determine what current actions can or cannot be. Second, to represent recent and long-term context at various granularities, we pool snippets over multiple scales.

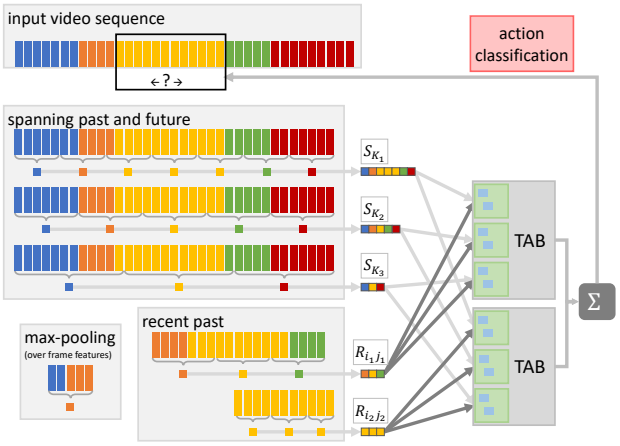


Figure 1. Model overview: In this example we use 3 scales for computing the "spanning context" snippet features $\mathbf{S}_{K_1}, \mathbf{S}_{K_2}, \mathbf{S}_{K_3}$, and 2 "recent" snippet features, $\mathbf{R}_{i_1,j_1}, \mathbf{R}_{i_2,j_2}$, by max-pooling over the frame features in each snippet. Each recent snippet is coupled with all the spanning snippets in our Temporal Aggregation Block (TAB). An ensemble of TAB outputs is used for action recognition. Best viewed in color.

2.1. Pooling

For a video of length T, we denote the feature representation of a single video frame indexed at time t as $f_t \in \mathbb{R}^D, 1 \leq t \leq T$. f_t can be derived from low-level features, such as I3D [2], or high-level abstractions, such as sub-activity labels derived from temporal action segmentation algorithms. To reduce computational load, we work at a snippet-level instead of at the frame level. We define a snippet feature $\mathbf{F}_{ij;K}$ as the concatenation of max-pooled features from K snippets, where snippets are partitioned consecutively from frames starting at i and ending at j:

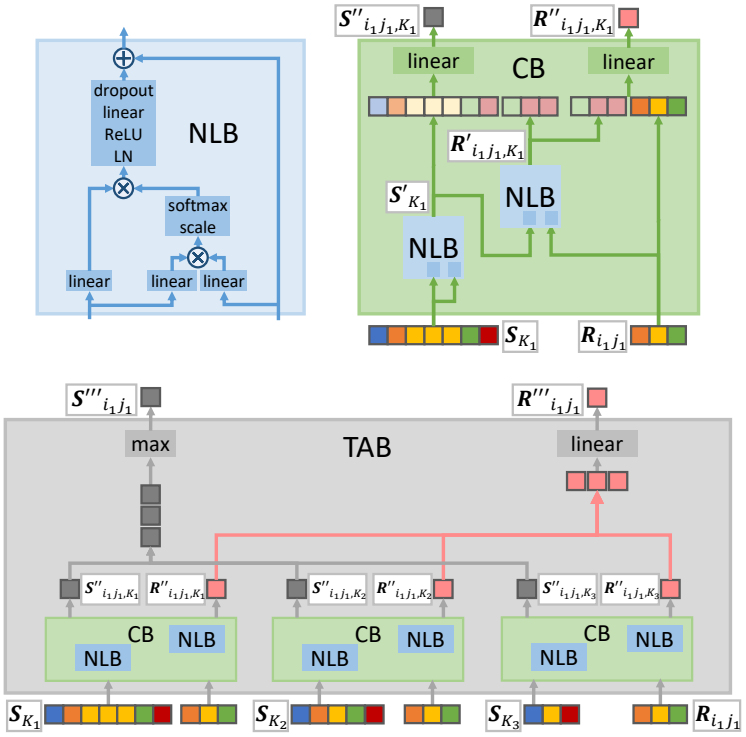


Figure 2. Overview of model components: Non-Local Blocks (NLB) compute interactions between two representations via attention (see Sec. 3.1). Two such NLBs are combined in a Coupling Block (CB) which calculates the attention-reweighted recent and spanning context representations (see Sec. 3.2). We couple each recent with all spanning representations via individual CBs and combine their outputs in a Temporal Aggregation Block (TAB) (see Sec. 3.3). The outputs of multiple such TABs are combined to perform classification through ensembles of multiple recents, see Fig. 1. Best viewed in color.

$$\mathbf{F}_{ij;K} = [F_{i,i+k}, F_{i+k+1,i+2k}, ..., F_{j-k+1,j}], \text{ where}$$
$$(F_{p,q})_d = \max_{p \le t \le q} \{f_t\}_d, \; 1 \le d \le D, \;\; k = (j-i)/K. \quad (1)$$

Here, $F_{p,q}$ indicates the maximum over each dimension d of the frame features in a given snippet between frames p and q, though it can be substituted with other alternatives.

2.2. Recent vs. Spanning Representations

Based on different start and end frames i and j and number of snippets K, we define two types of snippet features: *"recent"* features $\{\mathcal{R}\}$ from the action segment and *"spanning"* features $\{\mathcal{S}\}$ drawn from the longer past and future context around the action boundary. The recent snippets cover the action segment and several seconds around the action boundary to combat the annotation error. Spanning snippets refer to the longer context to make use of long-term sequence information for identifying the action. For *"recent"* snippets, the number of snippets is fixed to K_R. Recent snippet features $\mathcal{R}$ can be defined as a feature bank of snippet features with different start and end frames i, j, *i.e.*

$$\begin{aligned}\mathcal{R} &= \{\mathbf{F}_{i_1 j_1;K_R}, \mathbf{F}_{i_2 j_2;K_R}, ..., \mathbf{F}_{i_R j_R;K_R}\} \\ &= \{\mathbf{R}_{i_1 j_1}, \mathbf{R}_{i_2 j_2}, ..., \mathbf{R}_{i_R j_R}\},\end{aligned} \quad (2)$$

where $\mathbf{R}_{i,j} \in \mathbb{R}^{D \times K_R}$ is a shorthand to denote $\mathbf{F}_{ij;K_R}$, since the number of snippets K_R are fixed. In Fig. 1 we use two pairs of starting and ending points, (i_1, j_1) and (i_2, j_2), to compute the "recent" snippet features and represent each with $K_R = 3$ number of snippets (&).

For *"spanning"* snippets, start and end frames, s and e, are fixed to a long context window around action boundary, *i.e.* $s = i - c, e = j + c$. Spanning snippet features $\mathcal{S}$ are defined as a feature bank of snippet features with varying number of snippets K, *i.e.*

$$\begin{aligned}\mathcal{S} &= \{\mathbf{F}_{s\,e;K_1}, \mathbf{F}_{s\,e;K_2}, ..., \mathbf{F}_{s\,e;K_S}\} \\ &= \{\mathbf{S}_{K_1}, \mathbf{S}_{K_2}, ..., \mathbf{S}_{K_S}\},\end{aligned} \quad (3)$$

where $\mathbf{S}_K \in \mathbb{R}^{D \times K}$ is a shorthand for $\mathbf{F}_{e\,s;K}$. In Fig. 1 we use three scales to compute the "spanning context" snippet features with $K = \{7, 5, 3\}$ (, &).

Key to both types of representations is the ensemble of snippet features from multiple scales. We achieve this by

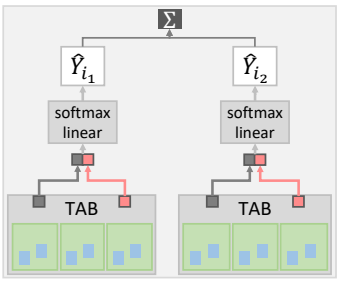


Figure 3. Prediction model for action recognition.

varying the number of snippets K for the spanning context. For recent, it is sufficient to keep the number of snippets K_R fixed, and vary only the start and end points i, j, due to redundancy between $\mathcal{R}$ and $\mathcal{S}$ for the snippets that overlap.

3. Framework

In Fig. 2 we present an overview of the components used in our framework, which we build in a bottom up manner, starting with the recent and spanning features $\mathcal{R}$ and $\mathcal{S}$, which are coupled with non-local blocks (NLB) (Sec. 3.1) within coupling blocks (CB) (Sec. 3.2). The outputs of the CBs from different scales are then aggregated inside temporal aggregation blocks (TAB) (Sec. 3.3). Outputs of different TABs can then be chained together for action recognition. (Sec. 4.1).

3.1. Non-Local Blocks (NLB)

We apply non-local operations to capture relationships amongst the spanning snippets and between spanning and recent snippets. Non-local blocks [10] are a flexible way to relate features independently from their temporal distance and thus capture long-range dependencies. We use the modified non-local block from [11], which adds layer normalization [1] and dropout [8] to the original one in [9]. Fig. 2 (left) visualizes the architecture of the block, the operation of which we denote as $\text{NLB}(\cdot,\cdot)$.

3.2. Coupling Block (CB)

Based on the NLB, we define attention-reweighted spanning and recent outputs as:

$$\mathbf{S}'_K = NLB(\mathbf{S}_K, \mathbf{S}_K) \quad \text{and} \quad \mathbf{R}'_{ij,K} = NLB(\mathbf{S}'_K, \mathbf{R}_{ij}). \tag{4}$$

The coupling is done by concatenating $\mathbf{R}'_{ij,K}$ with either $\mathbf{R}_{ij}$ or $\mathbf{S}'_K$ and passed through linear layers. This results in the fixed-length representations $\mathbf{R}''_{ij,K}$ and $\mathbf{S}''_{ij,K}$, where i is the starting point of the recent snippet, j is ending point of the recent snippet and K is the scale of the spanning snippet.

3.3. Temporal Aggregation Block (TAB)

The final representation for recent and spanning context is computed by aggregating outputs from multiple CBs. For the same recent, we concatenate $\mathbf{R}''_{ij,K_1}, ..., \mathbf{R}''_{ij,K_S}$ for all spanning scales and pass the concatenation through a linear layer to compute $\mathbf{R}'''_{ij}$. The final spanning context representation $\mathbf{S}'''_{ij}$ is a max over all $\mathbf{S}''_{ij,K_1}, ..., \mathbf{S}''_{ij,K_S}$. We empirically find that taking the max outperforms other alternatives like linear layers and/or concatenation for the spanning context.

TAB outputs, by varying recent starting points $\{i\}$ and ending points $\{j\}$ and scales of spanning snippets $\{K\}$, are multi-granular video representations that aggregate and encode both the recent and long-term context. We name these **temporal aggregate representations**. Fig. 1 shows an example with 2 recent starting points and 3 spanning scales.

3.4. Prediction Model

For action recognition task temporal aggregate representations can be used directly with a classification layer (linear + softmax). A cross-entropy loss based on ground truth labels Y can be applied to the predictions $\hat{Y}_{i,j}$, where Y is the action label for recognition, see Fig. 3.

Our final loss formulation is the sum of the cross entropy's over the actions:

$$\mathcal{L}_{\text{cl}} = -\sum_{r=1}^{R}\sum_{n=1}^{N_Y} Y_n \log(\hat{Y}_{i_r})_n, \tag{5}$$

where i_r is one of the R recent starting points, and N_Y is the total number of actions. During inference, the predicted scores are summed for a final prediction, *i.e.* $\hat{Y} = \max_n(\sum_{r=1}^{R} \hat{Y}_{i_r})_n$.

3.5. Implementation Details

We train our model using the Adam optimizer [6] with batch size 10, learning rate 10^{-4} and dropout rate 0.3. We train for 25 epochs and decrease the learning rate by a factor of 10 every 10^{th} epoch. We use 512 dimensions for all non-classification linear layers.

4. Results

Features We use the appearance (RGB), motion (optical flow) and object-based features provided by Furnari and Farinella [5]. They independently train the spatial and motion CNNs using the TSN [9] framework for action recognition on EPIC-Kitchens. They also train object detectors to recognize the 352 object classes of the EPIC-Kitchens dataset. The feature dimensions are 1024, 1024 and 352 for appearance, motion and object features respectively.

Parameters: The spanning scales $\{K\}$, recent scale K_R and recent starting points $\{i\}$ are given in Table 1. In our

# classes	# segments	$\{i, j\}$(in seconds)	spanning scope (s)	K_R	$\{K\}$
2513	39.6K	$\{(i, j), (i-1, j+1), (i-2, j+2), (i-3, j+3)\}$	$s = i - 6, e = j + 6$	5	$\{2, 3, 5\}$

Table 1. Dataset details and our respective model parameters.

	Top-1 Accuracy%			Top-5 Accuracy%			Precision (%)			Recall (%)		
	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
S1	66.56	49.60	**41.59**	90.10	77.03	64.11	59.43	45.62	25.37	41.65	46.25	26.98
S2	54.56	33.46	**26.97**	80.40	60.98	46.43	33.60	30.54	14.99	25.28	28.39	17.97

Table 2. Action recognition on EPIC tests sets, seen (S1) and unseen (S2)

work, we predict the action classes directly rather than predicting the verbs and nouns independently [3]. Directly predicting actions is shown to outperform the latter [4]. We use a validation set provided by Furnari and Farinella [5] for selecting EPIC-Kitchens parameters.

4.1. Recognition on EPIC-Kitchens

We train our model separately for each feature modality (appearance, motion and object) with the same parameters in Table 1; during inference we apply a late fusion of the predictions from the different modalities by average voting. We report our results for hold-out test data on EPIC-Kitchens Egocentric Action Recognition Challenge (2020) in Table 2 for seen kitchens (S1) with the same environments as in the training data and unseen kitchens (S2) of held out environments.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] 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.

[3] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[4] Antonino Furnari, Sebastiano Battiato, and Giovanni Maria Farinella. Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[5] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *International Conference on Computer Vision (ICCV)*, 2019.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Fadime Sener, Dipika Singhania, and Angela Yao. Temporal aggregate representations for long term video understanding. *arXiv preprint arXiv:2006.00830*, 2020.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[9] 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.

[10] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7794–7803, 2018.

[11] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krähenbühl, and Ross Girshick. Long-Term Feature Banks for Detailed Video Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Egocentric Action Recognition by Video Attention and Temporal Context

Juan-Manuel Pérez-Rúa[1] Antoine Toisoul[1] Brais Martinez[1] Victor Escorcia[1]
Li Zhang[1] Xiatian Zhu[1] Tao Xiang[1,2]

[j.perez-rua, a.toisoul, brais.a, v.castillo, li.zhang1, xiatian.zhu, tao.xiang]@samsung.com

[1] Samsung AI Centre, Cambridge [2] University of Surrey
United Kingdom

Abstract

We present the submission of Samsung AI Centre Cambridge to the CVPR2020 EPIC-Kitchens Action Recognition Challenge. In this challenge, action recognition is posed as the problem of simultaneously predicting a single 'verb' and 'noun' class label given an input trimmed video clip. That is, a 'verb' and a 'noun' together define a compositional 'action' class. The challenging aspects of this real-life action recognition task include small fast moving objects, complex hand-object interactions, and occlusions. At the core of our submission is a recently-proposed spatial-temporal video attention model, called 'W3' ('What-Where-When') attention [6]. We further introduce a simple yet effective contextual learning mechanism to model 'action' class scores directly from long-term temporal behaviour based on the 'verb' and 'noun' prediction scores. Our solution achieves strong performance on the challenge metrics without using object-specific reasoning nor extra training data. In particular, our best solution with multimodal ensemble achieves the 2^{nd} best position for 'verb', and 3^{rd} best for 'noun' and 'action' on the Seen Kitchens test set.

1. Introduction

EPIC-Kitchens is a large scale egocentric video benchmark for daily kitchen-centric activity understanding [1]. In this benchmark, the action classes are defined by combining verb and noun classes. By combining all the 352 nouns and 125 verbs, the number of all possible action classes will reach as large as 44,000. This dataset presents a long tail distribution as often occurred in natural scenarios. Besides, human-object interaction actions might be very ambiguous. For example, in a single video clip, a person might be washing a dish whilst interacting with a sponge, faucet and/or sink concurrently, and sometimes the in-interaction active object might be completely occluded. These factors all render action recognition on this dataset extremely challenging. Whilst significant progress has been made since the inception of this challenge [1, 7], it is rather clear from the performance of all previous winner solutions that fine-grained action recognition is still far from being solved.

In this attempt, we present a novel egocentric action recognition solution based on video attention learning and temporal contextual learning jointly. By focusing on the action class related regions in highly redundant video data over space and time, the model inference is made more robust against noisy and distracting observations. To this end, we exploit a recently-proposed What-Where-When (W3) video attention model [6]. Temporal context provides additional useful information beyond individual video clips, as there are inherent interdependent relationships of human actions in performing daily life activities. For instance, it is more likely that a person is grabbing a cup if previously he/she was opening a cupboard, than for example, if the person had just opened a washing machine. A Temporal Context Network (CtxtNet) is introduced to enhance model inference by considering temporally adjacent actions happening in a time window. To make a stronger solution, we adopt multi-modal fusion, as in [4], combining RGB (static appearance), optical flow (motion cue), and audio information together.

2. Methodology

In this section, we present the solution of our submission to the *EPIC-Kitchens Action Recognition challenge*. We first introduce the W3 attention model [6] in Section 2.1, and then describe our proposed temporal action context model (CtxtNet) in Section 2.2.

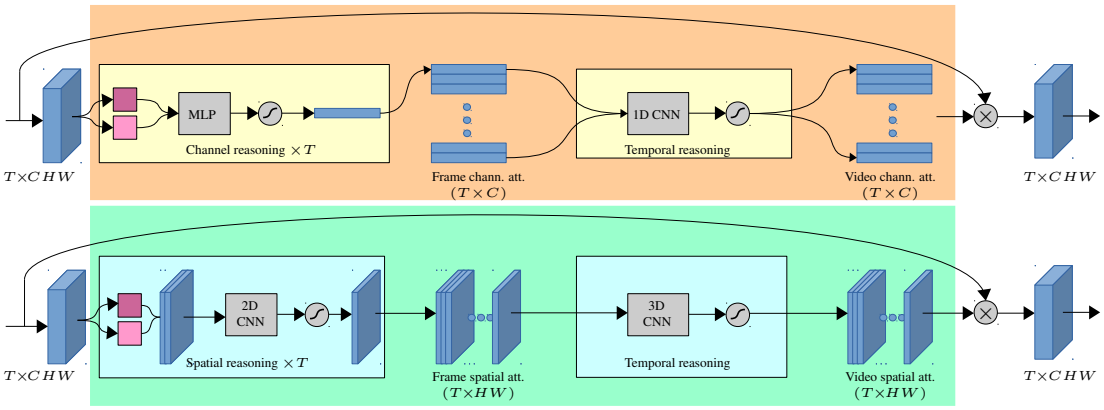


Figure 1: Schematic illustration of the W3 attention module. **Top:** The channel-temporal attention sub-module (orange box). **Bottom:** The spatial-temporal attention sub-module (green box). The symbol $\bigotimes$ denotes point-wise multiplication between the attention maps and input features.

2.1. What-Where-When Attention

Figure 1 gives the schematic illustration of the W3 attention module. Significantly, W3 can be plugged into any existing video action recognition network, *e.g.* TSM [5], for end-to-end learning. Specifically, W3 accepts a single feature map $\mathbf{F}$ as input, which can be derived from any CNN layer, and generates an attention map $\mathbf{M}$ with same dimension to $\mathbf{F}$, *i.e.*, $\mathbf{F}, \mathbf{M} \in \mathbb{R}^{T \times C \times H \times W}$, where T, C, H, W denote the number of video clip frame, number of feature channel, height and width of the frame-level feature map respectively. Attention mask $\mathbf{M}$ is then used to produce a refined feature map $\mathbf{F}'$ in a way that only action class-discriminative cues are allowed to flow forward, whilst irrelevant ones are suppressed. The attention and refined feature learning process is expressed as:

$$\mathbf{F}' = \mathbf{F} \otimes \mathbf{M}, \quad \mathbf{M} = f(\mathbf{F}), \tag{1}$$

where $\otimes$ is the Hadamard product, and $f(.)$ is the W3 attention function.

To facilitate effective and efficient attention learning, W3 adopts an attention factorization scheme by splitting the 4D attention tensor $\mathbf{M}$ into a channel-temporal attention mask $\mathbf{M}^c \in \mathbb{R}^{T \times C}$ and a spatial-temporal attention mask $\mathbf{M}^s \in \mathbb{R}^{T \times H \times W}$. This strategy reduces the complexity of the learning problem as the size of the attention masks are reduced from $TCHW$ to $T(C + HW)$. In principle, the feature attending scheme in Eq 1 is thus reformulated into a two-step sequential process:

$$\mathbf{F}^c = \mathbf{M}^c \otimes \mathbf{F}, \ \mathbf{M}^c = f^c(\mathbf{F}); \tag{2}$$

$$\mathbf{F}^s = \mathbf{M}^s \otimes \mathbf{F}^c, \ \mathbf{M}^s = f^s(\mathbf{F}^c); \tag{3}$$

where $f^c(.)$ and $f^s(.)$ denote the channel-temporal and spatial-temporal attention function respectively.

Channel-temporal attention The channel-temporal attention focuses on the 'what-when' facets of video attention. Specifically it measures the importance of a particular object-motion pattern evolving temporally across a video sequence in a specific way. For this, we squeeze the spatial dimensions ($H \times W$) of each frame-level 3D feature map to yield a compact channel descriptor $\mathbf{d}_{\text{chnl}} \in \mathbb{R}^{T \times C}$ as in [3]. Moreover, we use both max and mean pooling operations as in [9], and denote the two channel descriptors as $\mathbf{d}_{\text{avg-c}}$ and $\mathbf{d}_{\text{max-c}} \in \mathbb{R}^{C \times 1 \times 1}$ (indicated by the purple boxes in the top of Fig. 1).

To extract the inter-channel relationships for a given frame, we then forward $\mathbf{d}_{\text{avg-c}}$ and $\mathbf{d}_{\text{max-c}}$ into a MLP $\theta_{\text{c-frm}}$. The above *frame-level channel-temporal attention* can be expressed as:

$$\mathbf{M}^{\text{c-frm}} = \sigma\Big(f_{\theta_{\text{c-frm}}}(\mathbf{d}_{\text{avg-c}}) \oplus f_{\theta_{\text{c-frm}}}(\mathbf{d}_{\text{max-c}})\Big) \in \mathbb{R}^{C \times 1 \times 1}, \tag{4}$$

where $f_{\theta_{\text{c-frm}}}(.)$ outputs channel frame attention and $\sigma(.)$ is the sigmoid function.

In EPIC-Kitchens, it is critical to model the temporal dynamics of active objects in interaction with the human subject. To capture this information, a small channel temporal attention network $\theta_{\text{c-vid}}$ is introduced, composed of a CNN network with two layers of 1D convolutions, to reason about the temporally evolving characteristics of each channel dimension (Fig. 1 top-right). This results in our channel-temporal attention mask $\mathbf{M}^c$, computed as:

$$\mathbf{M}^{\text{c}} = \sigma\Big(f_{\theta_{\text{c-vid}}}(\{\mathbf{M}_i^{\text{c-frm}}\}_{i=1}^{T})\Big). \tag{5}$$

Concretely, this models the per-channel temporal relationships of successive frames in a local window specified by the kernel size $K_{\text{c-vid}}$, and composed by two layers.

Spatial-temporal attention In contrast to the channel-temporal attention that attends to dynamic object feature

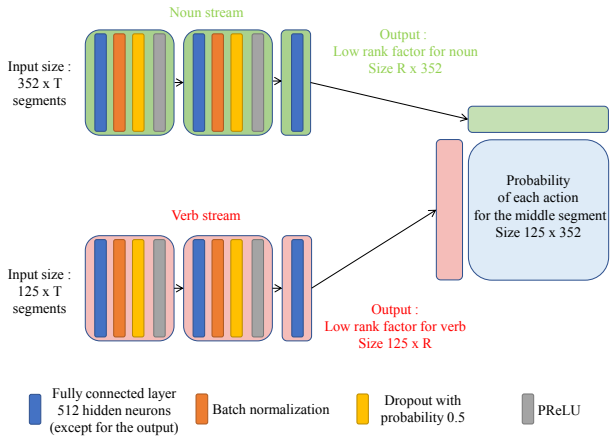


Figure 2: Overview of the proposed Temporal Context Network (CtxtNet). Noun and verb predictions are incorporated through a low-rank factorisation scheme to produce the final action scores.

patterns evolving temporally in certain ways, this sub-module attempts to localize them over time. Similarly to the previous module, we apply average-pooling and max-pooling along the channel axis to obtain two compact 2D spatial feature maps for each video frame, denoted as $\mathbf{d}_{\text{avg-s}}$ and $\mathbf{d}_{\text{max-s}} \in \mathbb{R}^{1\times H\times W}$. We then concatenate the two maps and deploy a spatial attention network $\theta_{\text{s-frm}}$ with one 2D convolutional layer for each individual frame to output the frame-level spatial attention $\mathbf{M}^{\text{s-frm}}$ (see Fig. 1 bottom-left). To incorporate the temporal dynamics to model how spatial attention evolves over time, we further perform temporal reasoning on $\{\mathbf{M}_i^{\text{s-frm}}\}_{i=1}^{T} \in \mathbb{R}^{T\times H\times W}$ using a lightweight 3D CNN $\theta_{\text{s-vid}}$. We adopt a kernel size of $3\times 3\times 3$ (Fig. 1 bottom-right). The *frame-level and video-level spatial attention* are, then:

$$\mathbf{M}^{\text{s-frm}} = \sigma\Big(f_{\theta_{\text{s-frm}}}([\mathbf{d}_{\text{avg-s}}, \mathbf{d}_{\text{max-s}}])\Big) \in \mathbb{R}^{1\times H\times W}, \quad (6)$$

$$\mathbf{M}^{\text{s}} = \sigma\Big(f_{\theta_{\text{s-vid}}}(\{\mathbf{M}_i^{\text{s-frm}}\}_{i=1}^{T})\Big) \in \mathbb{R}^{T\times H\times W}. \quad (7)$$

2.2. Temporal Context Network

The objective of contextual learning is to provide a per-clip action prediction by taking into account the surrounding actions (their verb and noun component), i.e., temporal context. This brings in additional information source on top of the observation of isolated short video clips. For instance, the action "open cupboard" is more likely to be followed by "close cupboard", as compared with "cut onions".

A straightforward method is to learn a non-linear mapping from the combination of verb and noun predictions to the action class label space. However, this is computationally not tractable due to the huge action spaces with 44000 class labels, which also runs a high risk of overfitting.

To alleviate these issues, we propose to learn a low rank factorisation of the action matrix to more efficiently encode context information by designing a Temporal Context Network (CtxtNet). An overview of CtxtNet is shown in Fig. 2.

In particular, CtxtNet is made of two parallel 3-layer MLP streams, one for noun and one for verb. Each stream generates a low rank matrix (of size $125\times R$ for verbs and $R\times 352$ for noun) whose multiplication yields the probability of each action, i.e., the action matrix of size 125×352. The hyperparameter R controls the rank of the factorisation so as allowing to choose the trade-off between complexity of the reconstruction (the number of parameters) and the model capacity. To encode the spatio-temporal context, the two streams act on a temporal context of T frames. In practice, we found that rank $R = 16$ and a time window $T = 5$ leads to the best results on a held-out validation set.

Model	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
TSM [5]	57.9	43.5	87.1	73.9	40.8	23.3	66.1	46.0	28.2	15.0	49.1	28.1
TSM+NL [8]	60.1	49.0	87.3	77.5	42.8	**27.7**	66.4	**51.2**	30.8	18.0	50.0	33.0
TSM+W2	63.4	50.0	**88.8**	77.0	44.3	26.7	68.6	50.0	33.2	17.9	**54.6**	32.7
TSM+**W3**	**64.7**	**51.4**	**88.8**	**78.5**	**44.7**	27.0	**69.0**	50.3	**34.2**	**18.7**	**54.6**	**33.7**

Table 1: TSM with different attention modules. Setting: 8 frames per video, only RGB frames. Backbone: ResNet-50 [2]. S1: Seen Kitchens; S2: Unseen Kitchens. W2: W3 without temporal component. Experiments run with 10-crops and 2 clips per-video.

Model	Verb	Noun	Action
TSM ResNet-50	58.88	42.74	30.40
TSM ResNet-101	62.14	45.16	34.28
TSM ResNet-152	63.39	45.70	34.78
TSM ResNet-152 + **W3**	62.64	46.66	36.86

Table 2: TSM using different ResNet backbones on the validation set. Setting: 8 frames per video, only RGB frames used.

	Verb		Noun		Action	
	Top-1 Acc.		Top-1 Acc.		Top-1 Acc.	
	S1	S2	S1	S2	S1	S2
Action Prior [7]	69.18	57.76	49.58	33.59	38.12	23.72
CtxtNet (Ours)	69.18	57.76	49.58	33.59	**39.30**	23.38

Table 3: Effect of CtxtNet. S1: Seen Kitchens; S2: Unseen Kitchens. Results obtained in the EPIC-Kitchens test server.

3. Experiments

Setup In the video classification track of EPIC-Kitchens, there are three classification tasks involved: noun classifi-

	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
All modalities	69.43 (2)	57.60 (4)	91.23 (2)	81.84 (4)	49.71 (3)	34.69 (4)	73.18 (3)	61.25 (3)	40.00 (3)	24.62 (6)	60.53 (3)	41.38 (6)

Table 4: Final scores on the testing server. Setting: 8 frames per video. Modalities: RGB-W3, RGB, Flow, Audio. Backbone: ResNet-50 [2]. S1: Seen Kitchens; S2: Unseen Kitchens. Results obtained in the EPIC-Kitchens test server with 1 crop and 2 clips per-video. (X): Position in the 2020 ranking.

cation, verb classification, and their combination. Two different held-out testing sets are considered: Seen Kitchens Testing Set (S1), and Unseen Kitchens Testing Set (S2).

Validation set To allow for apples-to-apples comparison to other methods, we used the same validation set as [4].

Experimental details We used the recent Temporal Shift Module (TSM) [5] as the baseline video recognition model in all the experiments. We trained our models for 50 epochs with SGD, at a learning rate of 0.02. The models were initialized by ImageNet pre-training. Unless otherwise mentioned, our default backbone network is a ResNet-152. W3 models were trained with mature feature regularisation (MFR) as described in [6]. The CtxtNet was trained with Adam in a second stage. Firstly, we employed our multi-modal ensemble to compute the verb and noun logits of each video segment. The CtxtNet then maps those verb and noun logits to an action probability matrix. For both branches of CtxtNet, the MLP is a stack of three linear layers. Each of them was formed by a linear projection, batch norm, PReLU and Dropout. For all the experiments, unless otherwise mentioned, we sampled two clips and a single central crop per video. Finally, for our last submission, we assembled two models per modality, except for audio, for which we only used a single model.

3.1. Attention Model Comparison

We compared our W3 attention with existing competitive alternatives. For fair comparison experiments, all attention methods use the same ResNet-50 based TSM [5] as the underlying video model. Table 1 shows that our W3 attention module is the strongest amongst several competitors.

3.2. Backbone Network Evaluation

We tested our method with different backbone networks. Table 2 shows that ResNet-152 [2] is slightly better than ResNet-101, and almost five points better than ResNet-50. Importantly, it is shown that W3 further brings extra model performance improvement on top of the strongest backbone on noun and action classification. However, we observed that the performance of verb is not benefited from W3. This can be exploited by using both type of models for the RGB modality.

3.3. Temporal Context Network

Table 3 shows that our CtxtNet module produces better scores than the action prior method introduced by [7]. CtxtNet brings a large gain on the seen kitchen setting at a small cost on the unseen kitchen setting. Note, noun and verb accuracy scores are unaffected since this does not change their predictions.

3.4. Multi-Modalities

Table 4 reports the final results of our method using three modalities: RGB, optical flow, and audio. This was made by a logit-level ensemble of regular RGB model (ResNet-152), W3-attended RGB model (ResNet-152), optical flow model (ResNet-152), and audio model (ResNet-34). For audio, we used spectrograms with the same format of [4].

4. Conclusion

In this report, we summarised the model designs and implementation details of our solution for video action classification. With the help of the proposed W3 video attention and temporal context learning, we achieved top-3 video action classification performance on the leaderboard.

References

[1] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018. 1

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 4

[3] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *ICCV*, 2018. 2

[4] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *CVPR*, 2019. 1, 4

[5] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019. 2, 3, 4

[6] Juan-Manuel Perez-Rua, Brais Martinez, Xiatian Zhu, Antoine Toisoul, Victor Escorcia, and Tao Xiang. Knowing what, where and when to look: Efficient video action modeling with attention. *arXiv preprint arXiv:2004.01278*, 2020. 1, 4

[7] Will Price and Dima Damen. An evaluation of action recognition models on epic-kitchens. *arXiv preprint arXiv:1908.00867*, 2019. 1, 3, 4

[8] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 3

[9] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *ECCV*, 2018. 2

FBK-HUPBA Submission to the EPIC-Kitchens Action Recognition 2020 Challenge

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

[1]Fondazione Bruno Kessler - FBK, Trento, Italy
[2]Computer Vision Center, Barcelona, Spain
[3]Universitat 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 Action Recognition 2020 Challenge. To participate in the challenge we deployed spatio-temporal feature extraction and aggregation models we have developed recently: Gate-Shift Module (GSM) [1] and EgoACO, an extension of Long Short-Term Attention (LSTA) [2]. We design an ensemble of GSM and EgoACO model families with different backbones and pre-training to generate the prediction scores. Our submission, visible on the public leaderboard with team name FBK-HUPBA, achieved a top-1 action recognition accuracy of 40.0% *on S1 setting, and* 25.71% *on S2 setting, using only RGB.*

1. Introduction

Egocentric action recognition is a challenging problem with some differences to third-person action recognition. In egocentric videos, only the hands and the objects that are manipulated under an action are visible. In addition, the motion in the video is a mixture of scene motion and ego-motion caused by the frequent body movements of the camera wearer. This ego-motion may or may not be representative of the action performed by the observer. Another peculiarity of egocentric videos is that there is typically one 'active object' among many in the scene. To identify which among the candidate objects in a cluttered scene is going to be the 'active object' requires strong spatio-temporal reasoning. Egocentric action recognition is generally posited as a multi-task learning problem to predict verb, noun and action labels which are related to each other.

To participate in the challenge, we put in place two spatio-temporal feature encoding techniques we have developed, that exhibit some complementarity from a video representation learning perspective:

- GSM [1] performs early (and deep) spatio-temporal aggregation of features;
- EgoACO [3] performs late (and shallow) temporal aggregation of frame-level or snippet-level features.

We have developed variants of both approaches for the challenge, by changing their backbone CNNs and pre-training techniques. We compiled an ensemble out of this pool of trained models to generate the verb-noun-action scores on the test set. Our submission, visible on the public leaderboard at https://competitions.codalab.org/competitions/20115#results, was obtained by averaging classification scores from ensemble members.

2. Models

In this section we briefly overview the two model classes utilized for participating in the challenge. More details can be found in the cited papers. Reference implementations of the models in PyTorch framework can be accessed at https://github.com/swathikirans.

2.1. Gate-Shift Module

Gate-Shift Module (GSM) [1] is a light weight feature encoding module capable of converting a 2D CNN into an efficient and effective spatio-temporal feature extractor, see Figure 1. GSM first applies spatial convolution on the layer input; this is the operation inherited from the 2D CNN base model where GSM is build in. Then, grouped spatial gating is applied, that is, gating planes are obtained for each of two channel groups, and applied on them. This separates the 2D convolution output into group-gated features and residual. The gated features are group-shifted forward and backward in time, and zero-padded. These are finally fused (added) with the residual and propagated to the next layer. This way, GSM selectively mixes spatial and temporal information through a learnable spatial gating.

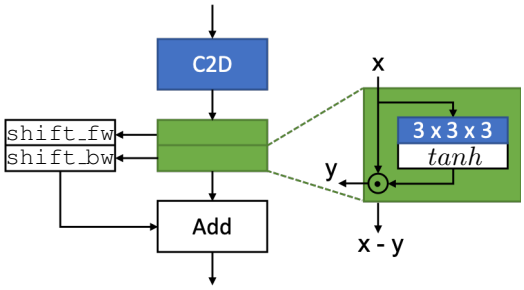


Figure 1: Overview of GSM (from [1]).

A Gate-Shift Network is instantiated by plugging GSM modules inside each of the layers of a 2D CNN, *e.g.* into an Inception style backbone pretrained on ImageNet. The network can be trained for egocentric action recognition in a multi-task setting for verb-noun-action prediction. In the classification layers, we linearly map the action scores into data dependent biases for verb and noun classifiers, as detailed in [2].

2.2. EgoACO

EgoACO is an extension of our previous conference publication, LSTA [2]. LSTA is a recurrent neural unit that extends ConvLSTM with built-in spatial attention and an enhanced output gating. LSTA can be used as frame-level or snippet-level feature sequence aggregator to obtain a video clip descriptor for egocentric action classification. The recurrent spatial attention mechanism enables the network to identify the relevant regions in the input frame or snippet and to maintain a history of the relevant feature regions seen in the past frames. This enables the network to have a smoother tracking of attentive regions. The enhanced output gating constraints LSTA to expose a distilled view of the internal memory. This allows for a smooth and focused tracking of the latent memory state across the sequence, which is used for verb-noun-action classification.

An overview of EgoACO expanding on LSTA is shown in Figure 2.

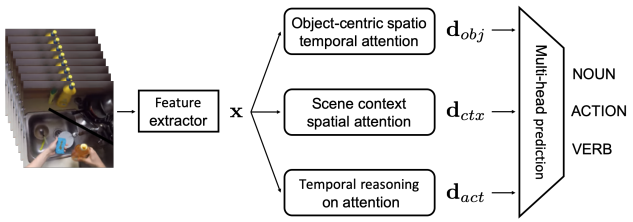


Figure 2: Overview of EgoACO (from [3]).

EgoACO uses LSTA to perform temporal reasoning to encode a feature sequence $\mathbf{x}$ into an action descriptor $\mathbf{d}_{act}$. Furthermore, the same feature $\mathbf{x}$ is used to generate two other clip descriptors, a scene context descriptor $\mathbf{d}_{ctx}$ and an active object descriptor $\mathbf{d}_{obj}$. The object descriptor encoding applies a frame level or snippet level attention as in [4] but in a temporally coherent manner, and we replace recurrent aggregation with parameter free temporal average pooling. The scene context descriptor $\mathbf{d}_{ctx}$ is generated by applying spatial attention on frame-level features independently, with an average pooled temporal aggregation. We use $\mathbf{d}_{obj}$ for noun prediction and $\mathbf{d}_{act}$ for verb prediction while all three descriptors are concatenated for action prediction. We apply action score as bias to verb and noun classifiers as done with GSM models.

3. Ensembling Design

The clip representations developed by the two model families, GSM and EgoACO, are complementary by design. GSM performs deep spatio-temporal fusion of features at each layer of a backbone CNN. Thus, the features encoded by GSM are highly discriminative for spatio-temporal reasoning. On the other hand, EgoACO relies on the features generated from a pre-trained backbone CNN with narrow temporal receptive field, and learns to combine these frame-level or snippet-level features to develop effective clip level representations. To participate in the challenge, we ensemble models from the two approaches to benefit from their complementarity. We instantiated several variants of the two model families by changing backbone CNNs and using different pre-training strategies.

3.1. GSM Variants

GSM variants are obtained in the following ways:

- Backbone: we used InceptionV3 and BNInception 2D CNNs;
- Pre-training: we used ImageNet and EPIC-Kitchens pre-trained models.

Training. The entire network is trained end-to-end using SGD with a momentum of 0.9 and a weight decay of $5e^{-4}$. We use a cosine learning rate schedule with linear warmup for 10 epochs. The initial learning rate is 0.01 and the network is trained for 60 epochs. A dropout of 0.5 is used to avoid overfitting. We use random flipping, scaling and cropping for data augmentation. For the EPIC-Kitchens [5] pre-trained model, we first trained InceptionV3 with EPIC-Kitchens in a TSN [6] fashion. For EPIC-Kitchens pre-trained BNInception, we used the backbone CNN from [7]. We sample 16 RGB frames from the input video for training and inference. We used the full training set for learning, without validation. We use the model parameters at epoch 60 for testing.

Testing. We perform spatially fully-convolutional inference, following [8]. The shorter side of the frames are

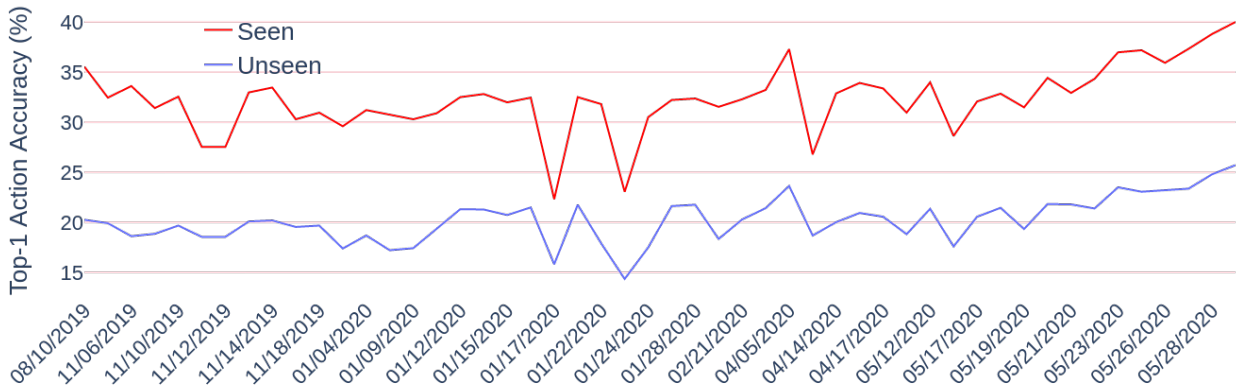


Figure 3: Submission history of our team to the EPIC-Kitchens evaluation server.

scaled to 256 for BNInception and 261 for InceptionV3 backbones. We also sample 2 clips from each video and the predictions of the two clips are averaged to obtain the final score.

3.2. EgoACO Variants

EgoACO variants are obtained in the following ways:

- Backbone: we used ResNet-34, ResNet-101, ResNet-152, InceptionV3, R(2+1)D ResNet-34, GSM-BNInception and GSM-InceptionV3;
- Pre-training: We used ImageNet and IG-65M+Kinetics [9] pre-trained models.

Training. We cloned the top layer of the pretrained CNN three times. The features from these three heads are used for predicting $\mathbf{d}_{obj}$, $\mathbf{d}_{ctx}$ and $\mathbf{d}_{act}$. We train the network in three stages. In the first stage, the weights of the backbone CNN are frozen while parameters of all the other layers are updated. In stage 2 the parameters of the three heads of the backbone CNN are trained together with the layers trained in stage 1. In stage 3, we train the parameters of `conv4_x` for ResNet based models and the last three layers of Inception based models, together with the layers trained in stage 2. SGD with momentum 0.9 and weight decay $5e^{-4}$ is used as the optimizer. We use a cosine learning rate schedule for adjusting the learning rate during training. Stages 1 and 2 are trained with an initial learning rate of 0.01 for 60 epochs while stage 3 is trained for 30 epochs with an initial learning rate of $1e^{-4}$. For LSTA we use a memory size of 512 and 300 output pooling classes. A dropout of 0.5 is applied to prevent overfitting. Random scaling, cropping and flipping are applied as data augmentation. 20 RGB frames sampled from the videos are applied as input to the network for the 2D CNN based backbones. For R(2+1)D based model, we sample 20 snippets consisting of 3 RGB frames while for GSM based models, we sample 16 RGB frames as input. We used the full training set for learning, without validation. We use the model parameters at epoch 30 of stage 3 for testing.

Testing. We use the center crop of the video frame, along with 2 clip sampling strategy during inference.

4. Results and Discussion

Figure 3 tracks our submission history to the EPIC-Kitchens evaluation server. We started off this year's challenge with our third place score from EPIC-Kitchens Action Recognition 2019 challenge. Our participation to the challenge can be divided into three phases. The first one corresponds to the development of GSM followed by the development of EgoACO. The third phase concerns with the development of the different variants of the two model families and their ensembling. Our final score is from the last submission.

Table 1 lists the models that were used as part of our ensemble. We chose the models based on the variability in pre-training, backbones and feature encoding approaches. The best single model is EgoACO with R(2+1)D ResNet-34 backbone that achieves a top-1 action accuracy of 37.32% on S1 and 23.35% on S2 setting. The result obtained from the submission server for each of the three ensembles is listed in Table 2. Model ensembling is done by averaging the prediction scores of the individual models. Our best performing ensemble, ensemble 3, results in a top-1 action accuracy of **40.0%** and **25.71%** in seen and unseen settings, respectively.

Acknowledgements

This research was supported by AWS Machine Learning Research Awards and ICREA under the ICREA Academia programme.

Method	Backbone	Pre-training	Ensemble 1	Ensemble 2	Ensemble 3
EgoACO	IncV3	ImageNet	✓	✓	×
	Res-34	ImageNet	✓	✓	×
	Res-101	ImageNet	✓	✓	✓
	Res-152	ImageNet	×	×	✓
	R(2+1)D Res-34	IG-65M + Kin.	×	✓	✓
GSM	BNInc	EPIC	✓	✓	✓
	IncV3	ImageNet	✓	✓	×
	IncV3	EPIC	×	✓	✓
GSM+EgoACO	BNInc	EPIC	✓	✓	×
	IncV3	EPIC	×	✓	✓
Ensemble 1	-	-	-	×	✓
Ensemble 2	-	-	×	-	✓

Table 1: List of individual models used as part of the ensembles.

	Method	Top-1 Accuracy (%)			Top-5 Accuracy (%)			Precision (%)			Recall (%)		
		Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
S1	Ensemble 1	66.98	47.21	37.28	89.71	71.05	58.18	57.91	43.25	18.84	45.41	43.96	22.70
	Ensemble 2	68.41	48.76	38.80	90.54	72.41	**60.41**	59.64	44.74	21.40	**47.45**	45.56	23.78
	Ensemble 3	**68.68**	**49.35**	**40.00**	**90.97**	**72.45**	60.23	**60.63**	**45.45**	**21.82**	47.19	**45.84**	**24.34**
S2	Ensemble 1	54.15	31.38	23.63	80.81	56.54	41.99	27.72	24.43	12.02	23.50	28.46	17.71
	Ensemble 2	55.96	32.23	24.79	81.56	58.83	44.14	**32.97**	26.43	**12.84**	**25.16**	**29.56**	**18.31**
	Ensemble 3	**56.67**	**33.90**	**25.71**	**81.87**	**59.68**	**44.42**	30.72	**27.25**	12.74	25.09	29.46	17.93

Table 2: Comparison of recognition accuracies obtained with various ensembles of models in EPIC-Kitchens dataset.

References

[1] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. Gate-Shift Networks for Video Action Recognition. In *Proc. CVPR*, 2020.

[2] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. LSTA: Long Short-Term Attention for Egocentric Action Recognition. In *Proc. CVPR*, 2019.

[3] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. Learning to Reason about Actions on Objects in Egocentric Video with Attention Dictionaries. *arXiv 2020.*

[4] Swathikiran Sudhakaran and Oswald Lanz. Attention is All We Need: Nailing Down Object-centric Attention for Egocentric Activity Recognition. In *Proc. BMVC*, 2018.

[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. The EPIC-KITCHENS Dataset: Collection, Challenges and Baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[6] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal Segment Networks for Action Recognition in Videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 41(11):2740–2755, 2018.

[7] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. Epic-Fusion: Audio-Visual Temporal Binding for Egocentric Action Recognition. In *Proc. ICCV*, 2019.

[8] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local Neural Networks. In *Proc. CVPR*, 2018.

[9] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Proc. CVPR*, 2019.

GT-UWISC Submission to Epic-Kitchens Action Recognition Challenge 2020

Miao Liu
Georgia Institute of Technology

Yin Li
University of Wisconsin-Madison

James M. Rehg
Georgia Institute of Technology

Abstract

In this report we briefly describe the technical details of our submission to the EPIC-Kitchens 2020 Action Recognition Challenge. Our model is consist of CSN152 backbone network and probabilistic attention module. The probabilistic attention module can effective suppress background objects in first person view and therefore boost the performance of action recognition. Our submission achieved a top-1 action recognition accuracy of 41.34% *on seen kitchen set, and* 27.35% *on unseen kitchen set.*

1. Introduction

In this work, we address the challenging task of action recognition in egocentric videos. A journal manuscript [7] of our method and the efforts on collecting EGTEA Gaze+ dataset is online at arXiv. Our previous work [6] show that the links between gaze and actions provide natural guidance for designing a novel deep model for action recognition. Specifically, we characterized gaze as a probabilistic distribution of attention in the context of an action. In this report, we further show that probabilistic attention can boost the performance of action recognition in first person videos even without human gaze as supervision. More importantly, we show that our model achieves competitive results using a single RGB stream network.

2. Approach

In this section, we detailed our approach of probabilistic attention modeling. A detailed illustration of probabilistic attention and our full model can be found in [8, 6] . We denote the input video as $x = \{x^1, x^2, ..., x^T\}$, where x^t is a frame of resolution $H \times W$ with t as the frame number. Given x, our goal is to predict a video-level action label y. Specifically, we leverage the intermediate output of a 3D convolutional network ϕ to represent x. Our model generates a probabilistic attention $\mathcal{A}$ and furthers uses $\mathcal{A}$ to guide action recognition.

Figure 1 presents an overview of our model. Consider an analogy between our model and the well-known R-CNN framework for object detection [3, 11]. Our model takes a video x as input and outputs the attention distribution $\mathcal{A}$ as an intermediate result. We then sample the attention map $\mathcal{A}$ from this predicted distribution. $\mathcal{A}$ encodes location information for actions and thus can be viewed as a source of action proposals—similar to the object proposals in R-CNN. Finally, we use the attention map to select features from the network hierarchy for recognition. This can be viewed as Region of Interest (ROI) pooling in R-CNN, where visual features in relevant regions are selected for recognition.

2.1. Attention Generation

Our probabilistic attention module models the distribution of linear mapping outputs as discussed in [6, 8], namely

$$\mathcal{A} \sim p(\mathcal{A}) = softmax(w_a * \phi(x)) \tag{1}$$

where we model the distribution of A. $w_a \in R^{C_\phi}$ is a linear mapping function and $*$ is the 1x1 convolution on 3D feature grids. Softmax is applied on every time slice to normalize each 2D map. During training, an attention map can be sampled from $p(\mathcal{A})$ using Gumbel Softmax trick [4, 10]. We follow [6] to regularize the learning by adding additional loss term of

$$\mathcal{L}^R = \sum_t KL\left[\mathcal{A}(t)||U\right], \tag{2}$$

where $KL[\cdot]$ is the Kullback-Leibler divergence and U is the 2D uniform distribution ($H_\phi \times W_\phi$). This term matches each time slice of the attention map to the prior distribution. It is derived from variational learning and accounts for (1) the prior of attention maps and (2) additional regularization by spatial dropout [6]. During testing, we directly plug in $p(\mathcal{A})$ (the expected value of $\mathcal{A}$) for approximate inference.

Note that for both approaches, we restrict w_a to a linear mapping without a bias term. In practice, this linear mapping avoids a trivial solution of generating a uniform attention map by setting w to all zeros. This all-zero solution almost never happens during our training when using a proper initialization of w.

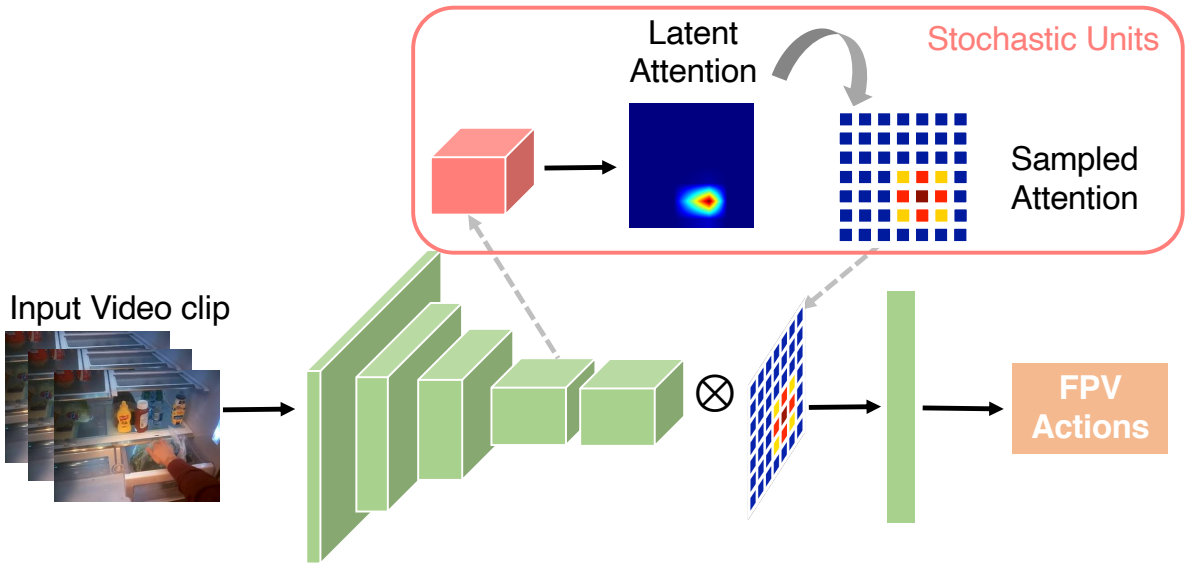


Figure 1. Overview of our probabilistic attention model. Our model takes multiple RGB frames as inputs, and generates an attention distribution in the middle layers. We then sample a attention map from this distribution. This map is used to selectively pool visual features at higher layers of the network for action recognition. During training, our model only receives action labels as supervisory signal. We show that this probabilistic attention network can facilitate visual representation learning in FPV.

2.2. Attention Guided Recognition

Our recognition module makes use of an attention map $\mathcal{A}$ to select features from $\phi(x)$.
Inspired by [9], we design the function $F_{\mathcal{R}}$ as

$$\tilde{y} = F_{\mathcal{R}}(\phi(x), \mathcal{A}) = softmax\left(W_r^T(\mathcal{A} \otimes \phi(x))\right) \quad (3)$$

where $\otimes$ denotes the tilted multiplication. This operation is equivalent to weighted average pooling with the weights shared across all channels.

2.3. Our Full Model

Our training loss is defined as

$$\mathcal{L} = CE(\tilde{y}, y) + \lambda \sum_t KL\left[\mathcal{A}^A(t) || U\right], \quad (4)$$

where CE is the cross entropy loss between the predicted labels $\tilde{y}$ and the ground-truth y. The KL term regularizes the learning of the visual attention. The coefficients λ is used to balance the loss terms. We choose $\lambda = 1/(T_\phi \times W_\phi \times H_\phi)$.

3. Implementation Details

We downsampled all video frames to 320×256 and randomly cropped 224×224 regions from 24 frames for training. We also performed random horizontal flip and color jittering for data augmentation. For testing, we send the frames with a resolution of 320×256 and their flipped version. For action recognition, we averaged pool scores of all clips within a video.

We adopt recent CSN152 model pre-trained by [13] as our backbone network. We used a batch size of 16, paralleled over 4 GPUs. We train the model for 18 epochs when competing on seen test set and 13 epochs when competing on unseen test set. We also adopt cosine learning rate decay. All our models are trained using SGD with momentum of 0.9 and weight decay of 0.00004. When using SGD for variational learning, we draw a single sample for each input within a mini-batch, and multiple samples of the same input will be drawn at different iterations.

4. Results

To better investigate the performance gap between using uniform prior and gaze prior for probabilistic attention modeling, we present an ablation study on EGTEA Gaze+ dataset [6]. As shown in 1, Prob-Uniform model outperforms the I3D baseline by 0.8%, and underperforms Prob-Gaze model by 0.7%. This suggests that using uniform distribution as prior for probabilistic modeling can facilitate the learning of visual representation. In addition, a more informative prior distribution (e.g. gaze) can further improve the performance.

We further compared our model to state-of-the-art methods [12, 15, 5, 2, 14] in Table 2. Our final model, using only RGB frames, achieves state-of-the-art results in comparison to all prior work, including those use optical flow [12], object detector [14] or audio data [5]. Our single model outperforms the best published single-model entries [5, 2] by a significant margin of 4.0%, 1.8% on the seen and unseen set, respectively. Moreover, our single model even beats large ensembles [5]. Our results are *ranked the 2nd*

Methods	Mean Accuracy
I3D	55.7
Prob-Uniform	56.5
Prob-Gaze	57.2

Table 1. Ablation study of probabilistic attention on EGTEA Gaze+ dataset. We report mean class accuracy of I3D baseline and probabilistic attention model using uniform distribution (Prob-Uniform) and gaze prior (Prob-Gaze).

	Method	Top1/Top5 Accuracy		
		Verb	Noun	Action
s1	2SCNN [1]	40.44 / 83.04	30.46 / 57.05	13.67 / 33.25
	TSN (fusion) [1]	48.23 / 84.09	36.71 / 62.32	20.54 / 39.79
	LSTA-2S [12]	62.12 / 87.95	40.41 / 64.47	32.60 / 52.85
	LFB Max [15]	60.00 / 88.40	45.00 / 71.80	32.70 / 55.30
	EPIC-Fusion [5]	64.75 / 90.70	46.03 / 71.34	34.80 / 56.65
	R(2+1)D [2]	65.20 / 87.40	45.10 / 67.80	34.50 / 53.80
	2SI3D+Obj [14]	69.80 / 90.95	52.27 / 76.71	41.37 / 63.59
	Ours (CSN152)	68.51 / 89.32	49.96 / 72.30	38.75 / 59.00
s2	2SCNN [1]	36.16 / 71.97	18.03 / 38.41	7.31 / 19.49
	TSN (fusion) [1]	39.40 / 74.29	22.70 / 45.72	10.89 / 25.26
	LSTA-2S [12]	48.89 / 77.88	24.27 / 46.06	18.71 / 33.77
	LFB Max [15]	50.90 / 77.60	31.50 / 57.80	21.20 / 39.40
	EPIC-Fusion [5]	52.69 / 79.93	27.86 / 53.78	19.06 / 39.40
	R(2+1)D [2]	57.30 / 81.10	35.70 / 58.70	25.60 / 42.70
	2SI3D+Obj [14]	58.96 / 82.69	33.90 / 62.27	25.20 / 45.48
	Ours (CSN152)	60.05 / 81.97	38.14 / 63.81	27.35 / 45.24

Table 2. **Action recognition results on Epic-Kitchens test sets**. We follow [1] to report top1/top5 accuracy for verb / noun / action on seen (s1) and unseen sets (s2). Our results are further compared against previous methods that use a single model. Our model (Ours CSN152) ranks2nd for the unseen set on the EPIC-Kitchens Action Recognition Challenge Leaderboard.

on the unseen set on EPIC-Kitchen leaderboard, where the top ranked entry on the seen set used object detector and model ensembles [14]. We must point out that the uniform attention prior used in our model is over-simplified, yet our model still achieves very competitive results on this larger benchmark. We speculate that better modeling of the attention prior (e.g. gaze) can further improve the performance. Note that our model achieves this performance without any side information (optical flow, object feature etc.). Fusing with optical flow and object feature can further improve the performance, but it is beyond the scope of our current investigation.

5. Conclusion

In this paper, we report the model details of our entry on EPIC-Kitchens action recognition challenge. We show that probabilistic attention model can improve the performance of action recognition even without gaze fixation as prior. Importantly, Our model achieves competitive results on EPIC-Kitchens dataset. We believe our model provide a solid step towards advancing First Person Vision.

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] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *CVPR*, 2019.

[3] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[4] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

[5] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *ICCV*, 2019.

[6] Yin Li, Miao Liu, and James M. Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *ECCV*, 2018.

[7] Yin Li, Miao Liu, and James M. Rehg. In the eye of the beholder: Gaze and actions in first person video, 2020.

[8] Miao Liu, Xin Chen, Yun Zhang, Yin Li, and James M Rehg. Paying more attention to motion: Attention distillation for learning video representations. *arXiv preprint arXiv:1904.03249*, 2019.

[9] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *CVPR*, 2019.

[10] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*. 2015.

[12] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. Lsta: Long short-term attention for egocentric action recognition. In *CVPR*, 2019.

[13] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *ICCV*, 2019.

[14] Xiaohan Wang, Yu Wu, Linchao Zhu, and Yi Yang. Baidu-UTS submission to the EPIC-kitchens action recognition challenge 2019. *arXiv preprint arXiv:1906.09383*, 2019.

[15] 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.

Gradient-Blending (G-Blend)
EPIC-Kitchens Action Recognition Challenge Entry

Weiyao Wang, Du Tran, Matt Feiszli
Facebook AI
{weiyaowang,trandu,mdf}@fb.com

Abstract

In this report we describe the technical details of our submission to the EPIC-Kitchens 2020 action recognition challenge (entry weiyaowang). We participate in the challenge with a simple late-fusion by concatenation model of a visual backbone and an audio backbone. The model is trained with an optimal blending strategy of the two signals [10]. We show that such a simple model can achieve surprisingly good results with 37.12% on S1 setting and 26.56% on S2 setting.

1. Introduction

Video understanding has been one of the most active research areas in computer vision recently. One unique feature with respect to video is multi-modality, and many previous work has focused on the fusion architecture of multiple modalities; in particular previous entry TBN-Ensemble [6] designs a novel method to combine audio, RGB Frames and optical flow. Different from previous works, we focus on the optimization aspect of multi-modal networks: how to jointly learn and optimally blend multi-modal signals.

We use late-fusion by concatenation of a visual model ip-CSN [9] and a ResNet-based audio model [3]. The visual backbone is pre-trained on IG-65M [2] and the audio backbone is pre-trained on ImageNet [8]. The joint model is then fine-tuned end-to-end with Gradient-Blending [10].

2. Our Approach

2.1. Backbone Models

We use ip-CSN-152 [9] to extract visual clip features. The model has a good accuracy-computation trade-off by leveraging 3D depthwise convolutions and achieves state-of-the-arts performance in multiple benchmarks such as Kinetics400 [5] and Sports-1M [4]. The model is pre-trained on large-scale weakly supervised IG-65M [2] dataset.

We use ResNet-152 [3] to extract audio spectrogram features. Similar to the training strategy of [6], we adopt ImageNet [8] to pre-train the audio backbone. Audio and visual came from the same clip sampled from the video and are aligned temporally.

The visual features and audio features are fused after the final pooling layer by concatenation (Fig. 1b). We train a three layer MLP (3 FCs and 2 ReLUs) to learn the concatenated features and make final clip-level prediction.

2.2. Joint-training via Gradient-Blending

We take the pre-trained visual backbone and audio backbone, and fine-tune the model end-to-end with the fusion layers. However, training this network end-to-end may be sub-optimal and lead to a classifier worse than uni-modal ones due to overfitting: on one hand, multi-modal network overfits more with the increased capacity; on the other hand, different modalities have different rates of learning, overfitting and generalizing [10].

[10] has proposed a quantitative method to measure the overfitting behavior through Overfitting-over-Generalization Ratio (OGR) and propose a method to compute an optimal blending of the gradient of different modalities to minimize OGR of the joint model. The method, Gradient-Blending (G-Blend), computes an optimal set of per-modality weights and use the weights to re-calibrate loss (equivalent to gradient) by taking weighted sum (Fig. 1c). In particular, a held-out set is constructed similar to the unseen kitchen, and different weights are learned for verb and noun.

3. Experiments

3.1. Experimental setup

Input processing. We extract 32 224-by-224 RGB frames to form a visual clip. This results in a tensor of $32\times224\times224$ input clip. During training the clip is sampled randomly from a video with standard scale jittering of [320, 256], and at test time, we extract 10 center-cropping clips uniformly and average their predictions. We extract audio

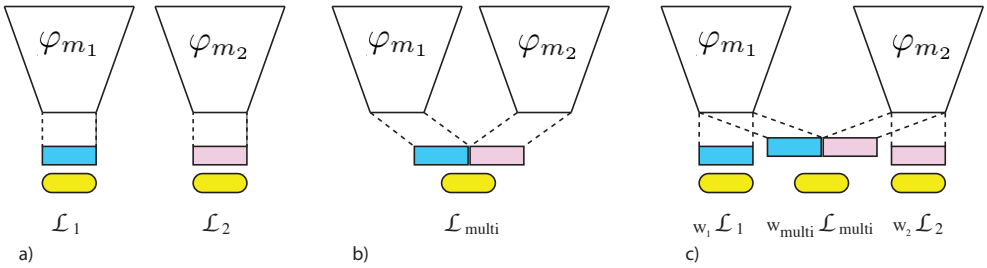


Figure 1: **Uni- vs. multi-modal joint training**. a) Uni-modal training of two different modalities. b) Naive joint training of two modalities by late fusion. c) Joint training of two modalities with weighted blending of supervision signals. Different deep network encoders (white trapezoids) produce features (blue or pink rectangles) which are concatenated and passed to a classifier (yellow rounded rectangles).

from the same visual clip (temporally aligned), and convert it to a log-spectrogram representations using STFT of 256 frequency bands, window length 16ms and hop length 10ms. We use linear interpolation to construct a fixed dimension of 256 on time axis of the spectrogram. This results in a 2D matrix input of 256×256.

Training Details. We train our models with synchronous distributed SGD with 0.9 momentum on GPU clusters using Caffe2 [1]. We train a total of 6 epochs with cosine learning rate decay [7]. S1 (seen kitchen) uses finetuning learning rate 0.0005 for noun and 0.0004 for verb; S2 (unseen kitchen) uses finetuning learning rate 0.0002 for both noun and verb. We speculate that a smaller learning rate may under-fit the model in the finetuning and thus helps in unseen case but hurts in seen case. Weight decay of 0.0001 is applied to all scenarios. No other regularizers are applied. The models for each modality are first pre-trained and then fine-tuned end-to-end.

Testing. We train the model once and use the single model to extract clip-level predictions from the 10 uniformly sampled clips (in contrast to ensemble of models). Video level prediction is made by averaging the 10 clip level predictions.

3.2. Learning Gradient-Blending weights

Intuitively, audio and visual models may have different behaviors on verb and noun. Therefore, we learn the G-Blend's per-modality weights for verb and noun separately. To measure model's overfitting behavior, we construct a separate hold-out set following the unseen-kitchen's construction (S2). As a result, the weights learned here are optimized towards S2, and it's possible that a different set of weights is more optimal for seen kitchen.

As summarized in Table 1, the optimal blending weights are quite different between noun classifier and verb classifier. We saw that audio takes a major importance in its weight in verb while being less considered during training by noun classifier.

	Audio	Visual	Joint
Noun	0.175	0.460	0.364
Verb	0.524	0.247	0.229

Table 1: **Per-Modality (gradient) weights learned from Gradient-Blending.** The optimal blending weights learned from G-Blend are quite different between noun and verb. Audio takes more importance in verb while being less important than noun.

3.3. Results

In Table 2, we present both the visual-only model of ip-CSN and naively trained audio-visual model. We observe a rather surprising phenomenon: with the addition of audio signal, the naive late fusion model is performing worse than the visual only model on a number of settings: verb classification on both S1 and S2, and noun classification on S2 (top-1 video prediction). This is consistent with the observation in [10], where multi-modal network may be outperformed by its uni-modal counterpart.

With Gradient-Blending, we observe that there is a significant gain on all settings compared to both visual-only model and the naively trained late-fusion model. On one hand, Gradient-Blending guarantees the multi-modal network to perform no worse than uni-modal ones; on the other hand, G-Blend proves the usefulness of audio signal: when optimized correctly, audio can be a useful signal even with such a simple fusion architecture.

We note that we use the same set of weights learned from holdout set constructed by following unseen kitchen's construction (S2). As a result, it is possible to find a more suitable set of weights for S1 and further improve the S1 results. We leave this for future work.

method	noun		verb		action	
Metric	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Test Unseen Kitchen (S2)						
Visual:ip-CSN-152	35.8	59.6	56.2	80.9	25.1	41.2
Naive Late Fusion	34.3	60.2	54.4	80.5	24.1	41.5
G-Blend	**36.7**	**60.3**	**58.3**	**81.3**	**26.6**	**43.6**
Test Seen Kitchen (S1)						
Visual:ip-CSN-152	45.1	68.4	64.5	88.1	34.4	52.7
Naive Late Fusion	45.5	70.8	62.7	87.6	33.3	53.8
G-Blend(ours)	**48.5**	**71.4**	**66.7**	**88.9**	**37.1**	**56.2**

Table 2: **Comparison with baseline methods.** G-Blend offers significant improvement over both the visual-only baseline and naive multi-modal baseline, on both S1 and S2. Note that on verb recognition, naively trained audio-visual model performs worse than the visual-only model.

4. Discussion

In uni-modal networks, diagnosing and correcting overfitting typically involves manual inspection of learning curves. Here we have shown that for multi-modal networks it is essential to measure and correct overfitting in a principled way. Our adopted method, Gradient-Blending, uses this insights to obtain significant improvements over baselines and achieves competitive results on the final leaderboard.

References

[1] Caffe2-Team. Caffe2: A new lightweight, modular, and scalable deep learning framework. https://caffe2.ai/. 2

[2] D. Ghadiyaram, M. Feiszli, D. Tran, X. Yan, H. Wang, and D. K. Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *CVPR*, 2019. 1

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[4] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 1

[5] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017. 1

[6] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *ICCV*, 2019. 1

[7] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with restarts. In *ICLR*, 2017. 2

[8] 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, 2012. 1

[9] D. Tran, H. Wang, L. Torresani, and M. Feiszli. Video classification with channel-separated convolutional networks. In *ICCV*, 2019. 1

[10] W. Wang, D. Tran, and M. Feiszli. What makes training multi-modal classification networks hard? In *CVPR*, 2020. 1, 2

VPUTeam: A Prospective Study on Sequence-Driven Temporal Sampling and Ego-Motion Compensation for Action Recognition in the EPIC-Kitchens Dataset

Alejandro López-Cifuentes
Video Processing and Understanding Lab
Universidad Autónoma de Madrid
alejandro.lopezc@uam.es

Marcos Escudero-Viñolo
Video Processing and Understanding Lab
Universidad Autónoma de Madrid
marcos.escudero@uam.es

Jesús Bescós
Video Processing and Understanding Lab
Universidad Autónoma de Madrid
j.bescos@uam.es

Abstract

Action recognition is currently one of the top-challenging research fields in computer vision. Convolutional Neural Networks (CNNs) have significantly boosted its performance but rely on fixed-size spatio-temporal windows of analysis, reducing CNNs temporal receptive fields. Among action recognition datasets, egocentric recorded sequences have become of important relevance while entailing an additional challenge: ego-motion is unavoidably transferred to these sequences. The proposed method aims to cope with it by estimating this ego-motion or camera motion. The estimation is used to temporally partition video sequences into motion-compensated temporal chunks showing the action under stable backgrounds and allowing for a content-driven temporal sampling. A CNN trained in an end-to-end fashion is used to extract temporal features from each chunk, which are late fused. This process leads to the extraction of features from the whole temporal range of an action, increasing the temporal receptive field of the network. This document is best viewed offline where some figures play as animation.[1]

1. Introduction

Video action classification is a highly emerging research topic in computer vision [8, 1, 10] due to its potential wide range of applications.

Among reported approaches, those relying on the use of Convolutional Neural Networks (CNNs) have reported the highest performances. These methods are based on the extraction of spatio-temporal features on the video, which are then used to classify the action recorded. Due to processing constraints, these features are instead usually extracted on video segments—i.e. fixed-size spatio-temporal windows obtained by sampling and cutting the video, hence discarding the rest of the video sequence. CNNs with reduced temporal receptive fields emerge from this design criteria.

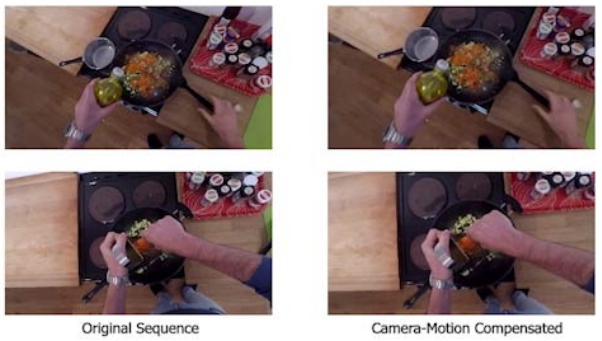


Figure 1. Left Column: Original video sequences corresponding to actions *Pour Oil* and *Grating Carrot*. Right Column: Results of the proposed camera motion compensation approach with unsupervised clustering. Best viewed with zoom in Adobe Reader where figure should play as videos.

There are several video action classification datasets in the literature, among them, egocentric ones, i.e. those recorded from a first person point of view, have become of crucial importance [2]. The egocentric domain presents a set of relevant advantages with respect to third-person recordings. The point of view in egocentric videos, closely related to the position of humans eyes, provides a view similar to what of what we humans actually see. Besides, the close proximity of the wearable camera to the undergoing

[1] You can find a dynamic preprint version of the paper at: Dynamic Version

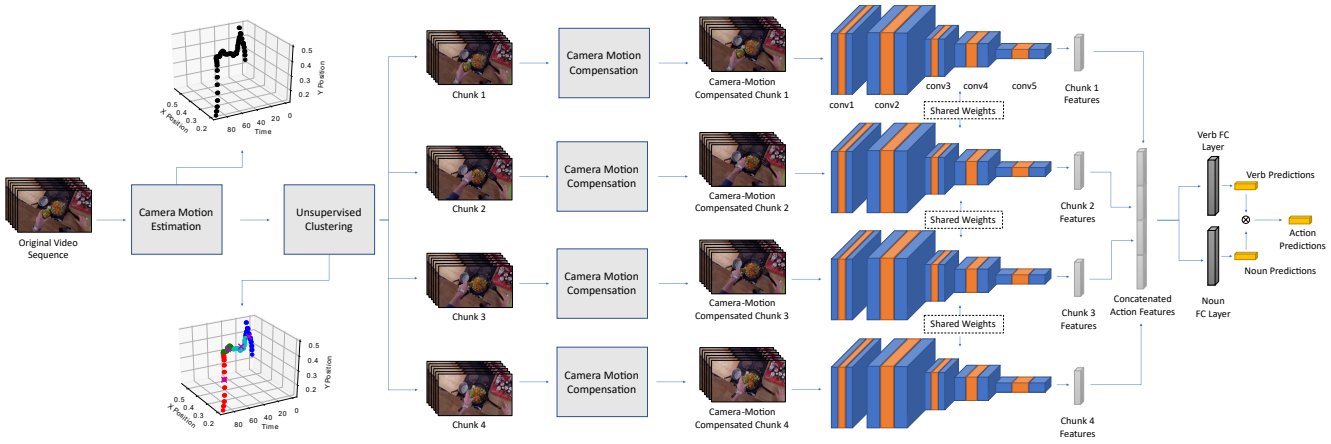


Figure 2. Proposed architecture for the action classification task. Camera motion is estimated for a given action sequence. Using this estimation an unsupervised 3D clustering based on spatial and temporal positions of the camera is performed, dividing the action sequence into temporal *chunks*. Camera motion is compensated independently for each chunk leading to quasi-static sub-sequences. Resulting *chunks* are fed to a 3D CNN which obtains features for each of them. Late fusion aims to gather features from all the *chunks* to obtain the final predictions for verb, noun and action.

action provides closer and more detailed objects representations compared to third-person view recordings.

However, egocentric recordings entail an additional challenge: human body or ego-motion is inevitably transferred to video sequences (see left column in Figure 1), creating motion and temporal patterns that may occlude or befoul the action's ones. This ego-motion might hinder the performance of modern CNNs trained to recognize actions in videos [9, 4, 11, 12] as these are focused on extracting the representative temporal features defining an action.

Our proposal aims to cope with ego-motion problems while providing a sequence-driven adaptive temporal sampling scheme. This process aims to filter out the majority of the ego-motion, leading to an action segment with a quasi-static background. Camera motion compensation accentuates the representative motion in a specific action, such as the movement of objects or hands (Figure 1), which might lead to more representative and distinguishable action features. In addition, we use the camera motion estimation to temporally divide the sequence into representative context temporal *chunks*. This temporal partitioning benefits the feature extraction process by: 1) Easing the process of camera motion compensation by having *chunks* that represent contextually similar backgrounds with limited camera motion. 2) Enabling the use of an irregular temporal sampling policy hence expanding the CNN temporal receptive field. We adopt a shared end-to-end fashion CNN to independently analyze each of the camera motion compensated *chunks*. Finally, features from each chunk are late fused to obtain final action predictions.

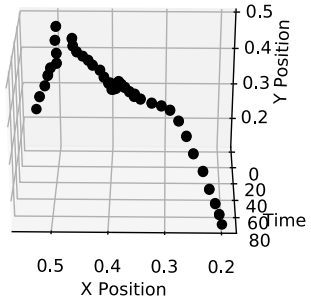


Figure 3. Camera motion estimation example. Each black dot corresponds to the projected middle coordinates of a frame using and homography with respect to a reference frame. X and Y axes represent normalized spatial movement while Z axis represents time in frame scale. Best viewed in Adobe Reader where figure should play as animation.

2. Method

Four different stages are posed for the action classification task. First, camera motion is estimated for an action video sequence (Section 2.1). Then, the resulting estimation is used to partition the original video sequence into temporal *chunks* via unsupervised clustering (Section 2.2). Camera motion is then compensated for each temporal chunk independently (Section 2.3). Finally, motion compensated *chunks* are fed to a CNN which obtains the final predictions (Section 2.4). The proposed pipeline is depicted and detailed in Figure 2.

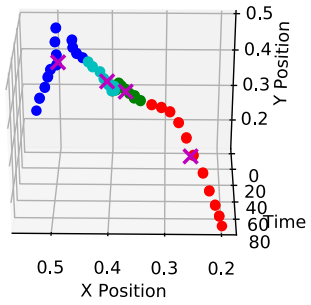


Figure 4. Unsupervised clustering of a video sequence into different *chunks* relying on camera motion estimation. This figure represents an example where 4 *chunks* are used. *Chunks* are represented using different color dots. Purple X markers represent the cluster center. Best viewed with zoom in Adobe Reader where figure should play as animation.

2.1. Camera Motion Estimation

We estimate the camera motion of a given sequence by using a classical stereo image rectification technique [6]. First, a pre-trained D2Net CNN [3] is used to obtain reliable pixel-level features for each video frame. Features of every frame are then matched to the ones from a chosen reference one—typically the first or the last frame. These matching correspondences are used to compute planar homographies using RANSAC [5]. So-extracted homographies define the transforms between every frame and the reference frame. An estimation of the camera motion during the sequence is obtained by projecting the middle point coordinates from every frame using the corresponding homography (Figure 3).

Video sequences might include high camera motion in terms of panning or tilting. This severe motion leads to high background variation along frames, hindering computation of a motion-compensated sequence. To overcome this issue we propose to divide each video sequences into different temporal chunks using unsupervised clustering.

2.2. Unsupervised Partition in Temporal Chunks

Unsupervised clustering is performed using the KMeans technique [7] fed by the spatio-temporal coordinates extracted in the camera motion estimation stage. By this, the temporal span of a video sequence is reduced and the original video sequence is divided into smaller parts with contextually similar backgrounds. An example of this process is depicted in Figure 4.

2.3. Camera Motion Compensation

The image rectification process described in section 2.1 is performed individually for each chunk reusing the features there extracted but defining a new reference frame for each chunk. Using the new obtained homographies, the frames in a *chunk* are warped to its *chunk*-reference frame to a camera motion compensated chunk (Figure 1 right column). These compensated *chunks* are the ones used for the end-to-end training of the shared Convolutional Neural Network (CNN). The proposed technique for camera motion compensation and sequence partition into chunks is highly improvable. This is a first approach to prove that this benefits the action recognition process.

2.4. CNN Architecture

To obtain temporal features from *chunks* the Slow-Fast ResNet-50 CNN [4] is used as backbone. The Slow-Fast is composed by the Spatial branch and the Temporal branch. For our participation, due to some computational constraints, we have only explored the use of the Temporal branch CNN. This branch expands temporal channels while reducing the spatial ones, ensuring that final features are closer to the temporal domain than to the spatial one.

Each chunk is independently fed-forwarded to the same 3D CNN to obtain features. Chunk features are later fused through concatenation to obtain a final feature vector. The use of *chunks* leads to the extraction of features from the whole temporal range of the action, hence increasing the temporal receptive field of the network with respect to analyzing only a fixed-size temporal window.

Two different fully-connected layers and regular logarithmic softmax are used to obtain verb (v) and noun (n) probabilities p_v and p_n respectively. p_v and p_n are used to obtain action predictions as:

$$p_a = p_v \otimes p_n, \tag{1}$$

where $(\otimes)$ is defined as the outer product between two vectors.

Inspired by the approaches in [12, 13], we re-weight action probabilities p_a by using training priors. These priors represent the probability $p_p(v, n)$ of a pair verb-noun being an action in the training set. Unobserved actions as *"peeling knife"* have $p_p(v, n) = 0$ whereas observed actions as *"open door"* have $p_p(v, n) = 1$. Final action probabilities are computed as:

$$p_a = p_p(v, n) \odot (p_v \otimes p_n), \tag{2}$$

where $(\odot)$ represents the Hadamard product.

2.5. Training Procedure

Given that the Challenge needs separate predictions for verb, noun and action, we have decided to train the action classification CNN in a multi-task fashion. The final loss guiding the training process is computed as:

Table 1. Preliminary results. (%)

Method	Number of Parameters	Top@1			Top@5		
		Action	Verb	Noun	Action	Verb	Noun
Baseline	0.8 M	29.00 %	52.44 %	40.78 %	48.22 %	**85.56 %**	**67.78 %**
Ours	1 M	**30.25 %**	**54.56 %**	**42.67 %**	**49.67 %**	85.33 %	66.89 %

Table 2. EPIC Kitchens Seen Test S1 results. (%)

Team	Position	Top@1			Top@5			Precision			Recall		
		Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
UTSBaidu	1	70.41	52.85	42.57	90.78	76.62	63.55	60.44	47.11	24.94	45.82	50.02	26.93
NUS CVML	2	66.56	49.60	41.59	90.10	77.03	64.11	59.43	45.62	25.37	41.65	46.25	26.98
FBK HuPBA	3	68.68	49.35	40.00	90.97	72.45	62.23	60.63	45.45	21.82	47.19	45.84	24.34
						[...]							
EPIC	37	39.00	13.93	6.01	77.03	33.98	16.90	16.33	5.59	1.10	12.19	6.47	1.32
SU	38	41.98	8.50	4.44	75.31	21.29	10.84	13.95	7.14	1.61	13.22	5.40	1.12
VPULab	39	24.12	6.06	2.55	58.47	16.35	6.71	12.67	8.73	2.70	8.49	6.11	1.41

Table 3. EPIC Kitchens Unseen Test S2 results. (%)

Team	Position	Top@1			Top@5			Precision			Recall		
		Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
UTSBaidu	1	60.43	37.28	27.96	83.07	63.67	46.81	35.23	32.60	17.35	28.97	32.78	19.82
GT WISC MPI	2	60.05	38.14	27.35	81.97	63.81	45.24	33.59	31.94	16.52	29.30	33.91	20.05
NUS CVML	3	54.56	33.46	26.97	80.40	60.98	46.43	33.60	30.54	14.99	25.28	28.39	17.97
						[...]							
EPIC	37	37.28	11.85	4.75	71.56	28.41	14.82	14.93	2.93	1.17	11.60	6.26	2.05
SU	38	33.05	4.88	2.15	66.17	14.20	6.21	9.01	2.33	0.99	9.23	3.30	0.99
VPULab	39	18.09	3.28	1.02	49.61	13.11	3.45	6.74	3.11	0.96	5.00	3.28	0.64

$$L = L_a + L_v + L_n, \tag{3}$$

with L_a, L_v and L_n being action loss, verb loss and noun loss respectively. The three losses are computed using regular Negative Log-Likelihood (NLL) loss using p_a, p_v and p_n. The influence of the L_a loss is explored in the Results Section.

3. Experiments

3.1. Implementation

3.1.1 Training

For the spatial domain each input image is adapted to the network input by resizing the smaller edge to 256 and then randomly cropping to a square shape of 224×224. Data augmentation via regular horizontal flips is also used.

For the temporal domain, the number of unsupervised *chunks* has been fixed to 4. From each of them, 6 consecutive frames are randomly extracted resulting in a total of 24 non-uniformly sampled frames along the video sequence.

In this submission, due to computational limitations, we have just performed a prospective study, using for training only a subset of 3600 from the 28.561 training sequences (i.e. a 12.60%)[2]. Final results on the whole set of test sequences defined in the challenge are highly biased by this issue.

To minimize the loss function in Equation 3 and optimize the network's trainable parameters, the regular Stochastic-Gradient-Descend with Momentum (SGD) algorithm is used. In all our experiments the initial learning rate was set to 0.1, Momentum was set to 0.9 and weight decay was set to 0.0001. Learning rate was decayed every 200 epochs by $1e-1$. Finally, batch size was set to 32 video sequences.

3.1.2 Inference

Following common practice, given a test sequence and its 4 *chunks*, we uniformly sample 5 clips for each chunk along its temporal axis. For each clip, we scale the shorter spatial side to 256 pixels and take 5 crops of 224×224 to cover the spatial dimension. This results in 25 different views per chunk and so, 100 views per test sequence. We average the softmax scores for the final prediction.

We report action, verb and noun prediction probabilities for the challenge as explained in Section 2.4.

[2]The specific training subset used for the study is available at: EPIC Kitchens 2020 Subset

3.2. Main Results

As stated in Section 3.1.1, due to computational limitations we use a subset of the training set for training and validation purposes. Specifically, from the 28.561 available sequences, we selected 4500. An 80% of these sequences were dedicated for training while the rest was used for validation. These division leads to 3600 training sequences which represents a 12.60% of the available training data. Given this fact, the following prospective study is a preliminary set of experimental results which may validate our starting hypothesis. Future work will extend and complete this work by using all the available training data.

3.2.1 Preliminary Experiments

The aim of this section is to gauge the influence of the ego-motion compensation approach and the designed shared CNN architecture. To this aim, we have uses the original Temporal branch architecture from SlowFast [4] as a baseline. This CNN is fed by non-compensated temporal windows of 24 consecutive frames from EPIC Kitchens sequences. To present a fair comparison with the proposed approach, training and inference details from Section 3.1 are applied in the same manner to the baseline. Results are presented in Table 1.

Comparing results obtained by the baseline (non compensated videos, fixed temporal window) with the proposed *Ours* method (compensated videos, non-uniform temporal sampling) we observe an increase in performance. This preliminary experiment suggests that the use of compensated sequences along with the non-overlapping sampling, while implemented via a highly improvable approach, increases Top@1 results by a 1.25%, a 2.12% and a 1.89% for action, verb and noun respectively.

3.2.2 2020 Challenge Results

Although results are highly biased due to the reduced amount of used training data, we report 2020 Challenge results for both Seen (S1) and Unseen (S2) tests sets in Tables 2 and Table 3 respectively.

4. Conclusions

This participation describes a novel approach for action recognition in egocentric videos based on camera motion compensation and a non-uniform temporal sampling. To this aim, ego-motion is estimated for a given video sequence. To overcome the problems of motion compensation in sequences with high camera motion, each video sequence is partitioned into temporal *chunks* using unsupervised clustering. Camera motion is compensated for each chunk independently leading to subsequences with a quasi-static background. Compensated *chunks* are then fed to a shared CNN to obtain features from each of them. Late fusion gathers features from all the chunk expanding the temporal receptive field of the CNN.

We have performed a prospective study on a limited set of training data from the whole EPIC Kitchens Dataset. Preliminary results indicate that the proposed approach, while implemented in a highly improvable way, increases performance with respect to a baseline based on non-compensated sequences and a fixed temporal window of analysis.

Future work will continue exploring this line of research besides of using the whole training set available.

References

[1] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018. 1

[2] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018. 1

[3] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2019. 3

[4] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, pages 6202–6211, 2019. 2, 3, 5

[5] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 3

[6] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 3

[7] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002. 3

[8] 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 preprint arXiv:1705.06950*, 2017. 1

[9] Will Price and Dima Damen. An evaluation of action recognition models on epic-kitchens. *arXiv preprint arXiv:1908.00867*, 2019. 2

[10] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016. 1

[11] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 2

[12] Xiaohan Wang, Yu Wu, Linchao Zhu, and Yi Yang. Baiduuts submission to the epic-kitchens action recognition challenge 2019. *arXiv preprint arXiv:1906.09383*, 2019. 2, 3

[13] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 284–293, 2019. 3

EPIC-Kitchens Egocentric Action Anticipation Challenge 2020
User "action_banks" Team "NUS_CVML" Technical Report

Fadime Sener[1,2], Dipika Singhania[2], Angela Yao[2]
[1]University of Bonn, Germany
[2]National University of Singapore
sener@cs.uni-bonn.de, dipika16@comp.nus.edu.sg, ayao@comp.nus.edu.sg

Abstract

This technical report describes Team "NUS_CVML" approach to the EPIC-Kitchens Egocentric Action Anticipation Challenge 2020. In this report, we predict upcoming actions in long videos of daily activities. Future prediction requires reasoning from current and past observations and raises several fundamental questions. How should temporal or sequential relationships be modelled? What temporal extent of information and context needs to be processed? At what temporal scale should they be derived? We address these questions with a flexible multi-granular temporal aggregation framework. We conduct experiments on the EPIC-Kitchens dataset and show that it is possible to achieve competitive results using simple techniques such as max-pooling and attention.

1. Introduction

We tackle long-term video understanding, specifically anticipating not-yet-observed but upcoming actions. Motivated by the questions of temporal modelling, extent, and scaling, we propose a general framework for encoding long-term video. We split video streams into snippets of equal length and max-pool the frame features within the snippets. We then create ensembles of multi-scale feature representations that are aggregated bottom-up based on scaling and temporal extent. The model we used for this challenge is described in detail in [7], and we refer the reader to this paper for further details and evaluations on other datasets.

2. Representations

We begin by introducing the representations, which are inputs to the building blocks of our framework, see Fig. 1. We had two rationales when designing our network. First, the coupling blocks relate recent observations to long-range past, since some actions directly determine what future actions can or cannot be. Second, to represent recent and long-term past at various granularities, we pool snippets over multiple scales.

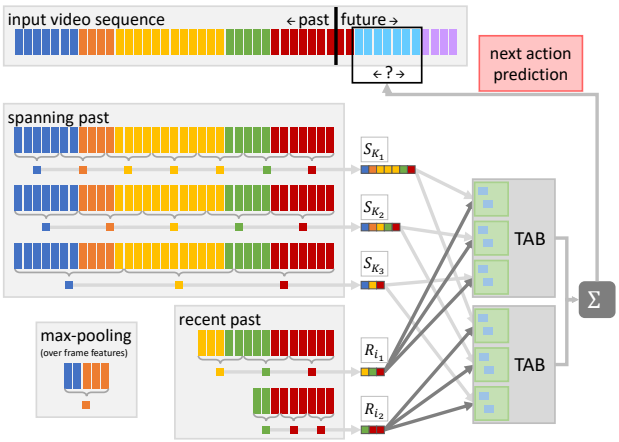


Figure 1. Model overview: In this example we use 3 scales for computing the "spanning past" snippet features $\mathbf{S}_{K_1}, \mathbf{S}_{K_2}, \mathbf{S}_{K_3}$, and 2 starting points to compute the "recent past" snippet features, $\mathbf{R}_{i_1}, \mathbf{R}_{i_2}$, by max-pooling over the frame features in each snippet. Each recent snippet is coupled with all the spanning snippets in our Temporal Aggregation Block (TAB). An ensemble of TAB outputs is used for next action anticipation. Best viewed in color.

2.1. Pooling

For a video of length T, we denote the feature representation of a single video frame indexed at time t as $f_t \in \mathbb{R}^D, 1 \leq t \leq T$. f_t can be derived from low-level features, such as I3D [2], or high-level abstractions, such as sub-activity labels derived from temporal action segmentation algorithms. To reduce computational load, we work at a snippet-level instead of at the frame level. We define a snippet feature $\mathbf{F}_{ij;K}$ as the concatenation of max-pooled features from K snippets, where snippets are partitioned consecutively from frames starting at i and ending at j:

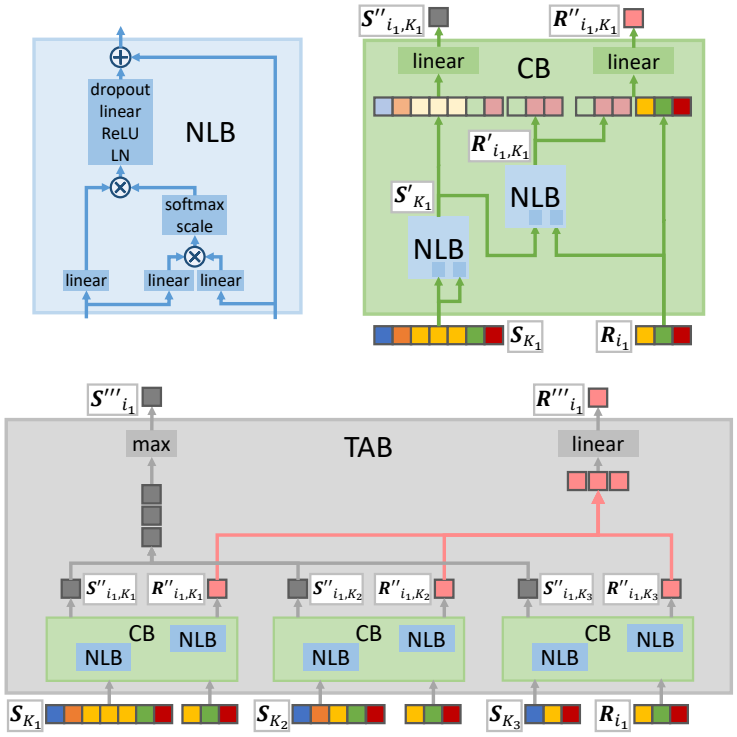


Figure 2. Overview of model components: Non-Local Blocks (NLB) compute interactions between two representations via attention (see Sec. 3.1). Two such NLBs are combined in a Coupling Block (CB), which calculates the attention-reweighted spanning and recent past representations (see Sec. 3.2). We couple each recent with all spanning representations via individual CBs and combine their outputs in a Temporal Aggregation Block (TAB) (see Sec. 3.3). The outputs of multiple such TABs are combined to perform anticipation, see Fig. 1. Best viewed in color.

$$\mathbf{F}_{ij;K} = [F_{i,i+k}, F_{i+k+1,i+2k}, ..., F_{j-k+1,j}], \text{ where}$$
$$(F_{p,q})_d = \max_{p \le t \le q} \{f_t\}_d, \; 1 \le d \le D, \quad k = (j-i)/K. \tag{1}$$

Here, $F_{p,q}$ indicates the maximum over each dimension d of the frame features in a given snippet between frames p and q, though it can be substituted with other alternatives.

2.2. Recent vs. Spanning Representations

Based on different start and end frames i and j and number of snippets K, we define two types of snippet features: *'recent'* features $\{\mathcal{R}\}$ from recent observations and *"spanning"* features $\{\mathcal{S}\}$ drawn from the long-term video. The recent snippets cover the couple of seconds (or up to a minute, depending on the temporal granularity) before the current time point, while spanning snippets refer to the long-term past and may last up to ten minutes. For *"recent"* snippets, the end frame j is fixed to the current time point t and the number of snippets is fixed to K_R. Recent snippet features $\mathcal{R}$ can be defined as a feature bank of snippet features with different start frames i, *i.e.*

$$\begin{aligned}\mathcal{R} &= \{\mathbf{F}_{i_1 t;K_R}, \mathbf{F}_{i_2 t;K_R}, ..., \mathbf{F}_{i_R t;K_R}\} \\ &= \{\mathbf{R}_{i_1}, \mathbf{R}_{i_2}, ..., \mathbf{R}_{i_R}\},\end{aligned} \tag{2}$$

where $\mathbf{R}_i \in \mathbb{R}^{D \times K_R}$ is a shorthand to denote $\mathbf{F}_{i\,t;K_R}$, since endpoint t and number of snippets K_R are fixed. In Fig. 1 we use two starting points to compute the "recent past" snippet features and represent each with $K_R = 3$ number of snippets (&).

For *"spanning"* snippets, i and j are fixed to the start of the video and current time point, *i.e.* $i = 0, j = t$. Spanning snippet features $\mathcal{S}$ are defined as a feature bank of snippet features with varying number of snippets K, *i.e.*

$$\begin{aligned}\mathcal{S} &= \{\mathbf{F}_{0\,t;K_1}, \mathbf{F}_{0\,t;K_2}, ..., \mathbf{F}_{0\,t;K_S}\} \\ &= \{\mathbf{S}_{K_1}, \mathbf{S}_{K_2}, ..., \mathbf{S}_{K_S}\},\end{aligned} \tag{3}$$

where $\mathbf{S}_K \in \mathbb{R}^{D \times K}$ is a shorthand for $\mathbf{F}_{0\,t;K}$. In Fig. 1 we use three scales to compute the "spanning past" snippet features with $K = \{7, 5, 3\}$ (, , &).

Key to both types of representations is the ensemble of snippet features from multiple scales. We achieve this by varying the number of snippets K for the spanning past.

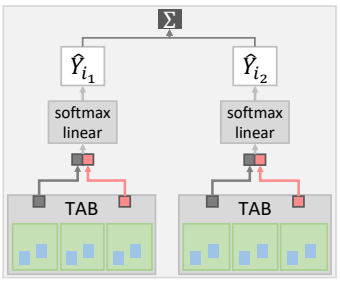


Figure 3. Prediction model next action anticipation.

For the recent past, it is sufficient to keep the number of snippets K_R fixed, and vary only the start point i, due to redundancy between $\mathcal{R}$ and $\mathcal{S}$ for the snippets that overlap.

3. Framework

In Fig. 2 we present an overview of the components used in our framework, which we build in a bottom up manner, starting with the recent and spanning features $\mathcal{R}$ and $\mathcal{S}$, which are coupled with non-local blocks (NLB) (Sec. 3.1) within coupling blocks (CB) (Sec. 3.2). The outputs of the CBs from different scales are then aggregated inside temporal aggregation blocks (TAB) (Sec. 3.3). Outputs of different TABs can then be chained together for next action anticipation (Sec. 4.1).

3.1. Non-Local Blocks (NLB)

We apply non-local operations to capture relationships amongst the spanning snippets and between spanning and recent snippets. Non-local blocks [10] are a flexible way to relate features independently from their temporal distance and thus capture long-range dependencies. We use the modified non-local block from [11], which adds layer normalization [1] and dropout [8] to the original one in [9]. Fig. 2 (left) visualizes the architecture of the block, the operation of which we denote as $\mathrm{NLB}(\cdot,\cdot)$.

3.2. Coupling Block (CB)

Based on the NLB, we define attention-reweighted spanning and recent outputs as:

$$\mathbf{S}'_K = NLB(\mathbf{S}_K, \mathbf{S}_K) \quad \text{and} \quad \mathbf{R}'_{i,K} = NLB(\mathbf{S}'_K, \mathbf{R}_i). \tag{4}$$

The coupling is done by concatenating $\mathbf{R}'_{i,K}$ with either $\mathbf{R}_i$ or $\mathbf{S}'_K$ and passed through linear layers. This results in the fixed-length representations $\mathbf{R}''_{i,K}$ and $\mathbf{S}''_{i,K}$, where i is the starting point of the recent snippet and K is the scale of the spanning snippet.

3.3. Temporal Aggregation Block (TAB)

The final representation for recent and spanning past is computed by aggregating outputs from multiple CBs. For the same recent starting point i, we concatenate $\mathbf{R}''_{i,K_1}, ..., \mathbf{R}''_{i,K_S}$ for all spanning scales and pass the concatenation through a linear layer to compute $\mathbf{R}'''_i$. The final spanning past representation $\mathbf{S}'''_i$ is a max over all $\mathbf{S}''_{i,K_1}, ..., \mathbf{S}''_{i,K_S}$. We empirically find that taking the max outperforms other alternatives like linear layers and/or concatenation for the spanning past.

TAB outputs, by varying recent starting points $\{i\}$ and scales of spanning snippets $\{K\}$, are multi-granular video representations that aggregate and encode both the recent and long-term past. We name these **temporal aggregate representations**. Fig.1 shows an example with 2 recent starting points and 3 spanning scales.

3.4. Prediction Model

For single-label classification task temporal aggregate representations can be used directly with a classification layer (linear + softmax). A cross-entropy loss based on ground truth labels Y can be applied to the predictions $\hat{Y}_i$, where Y is the next action label for next action prediction, see Fig. 3.

Our final loss formulation is the sum of the cross entropies over the actions:

$$\mathcal{L}_{\text{cl}} = -\sum_{r=1}^{R}\sum_{n=1}^{N_Y} Y_n \log(\hat{Y}_{i_r})_n, \tag{5}$$

where i_r is one of the R recent starting points, and N_Y is the total number of actions. During inference, the predicted scores are summed for a final prediction, *i.e.* $\hat{Y} = \max_n(\sum_{r=1}^{R} \hat{Y}_{i_r})_n$.

3.5. Implementation Details

We train our model using the Adam optimizer [6] with batch size 10, learning rate 10^{-4} and dropout rate 0.3. We train for 25 epochs and decrease the learning rate by a factor of 10 every 10^{th} epoch. We use 512 dimensions for all non-classification linear layers.

4. Results

Features We use the appearance (RGB), motion (optical flow) and object-based features provided by Furnari and Farinella [5]. They independently train the spatial and motion CNNs using the TSN [9] framework for action recognition on EPIC-Kitchens. They also train object detectors to recognize the 352 object classes of the EPIC-Kitchens dataset. The feature dimensions are 1024, 1024 and 352 for appearance, motion and object features respectively.

Parameters: The spanning scales $\{K\}$, recent scale K_R and recent starting points $\{i\}$ are given in Table 1. In our work, we anticipate the action classes directly rather than

Dataset	# classes	# segments	$\{i\}$(in seconds (s))	spanning scope (s)	K_R	$\{K\}$
EPIC	2513	39.6K	$\{t-1.6, t-1.2, t-0.8, t-0.4\}$	6	2	$\{2, 3, 5\}$

Table 1. Dataset details and our respective model parameters.

	Top-1 Accuracy%			Top-5 Accuracy%			Precision (%)			Recall (%)		
	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
S1	37.87	24.10	**16.64**	79.74	53.98	36.06	36.41	25.20	9.64	15.67	22.01	10.05
S2	29.50	16.52	**10.04**	70.13	37.83	23.42	20.43	12.95	4.92	8.03	12.84	6.26

Table 2. Action anticipation on EPIC tests sets, seen (S1) and unseen (S2)

anticipating the verbs and nouns independently [3]. Directly predicting actions is shown to outperform the latter [4]. We use a validation set provided by Furnari and Farinella [5] for selecting EPIC-Kitchens parameters.

4.1. Anticipation on EPIC-Kitchens

The **anticipation** task of EPIC-Kitchens requires anticipating the future action $\tau_\alpha = 1$s before it starts. We train our model separately for each feature modality (appearance, motion and object) with the same parameters in Table 1; during inference we apply a late fusion of the predictions from the different modalities by average voting. We report our results for hold-out test data on EPIC-Kitchens Egocentric Action Anticipation Challenge (2020) in Table 2 for seen kitchens (S1) with the same environments as in the training data and unseen kitchens (S2) of held out environments.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] 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.

[3] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[4] Antonino Furnari, Sebastiano Battiato, and Giovanni Maria Farinella. Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[5] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *International Conference on Computer Vision (ICCV)*, 2019.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Fadime Sener, Dipika Singhania, and Angela Yao. Temporal aggregate representations for long term video understanding. *arXiv preprint arXiv:2006.00830*, 2020.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[9] 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.

[10] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7794–7803, 2018.

[11] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krähenbühl, and Ross Girshick. Long-Term Feature Banks for Detailed Video Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Team VI-I2R Technical Report on EPIC-Kitchens Action Anticipation Challenge 2020

Ying Sun, Yi Cheng, Mei Chee Leong, Hui Li Tan, Kenan E. Ak
Institute for Infocomm Research, A*STAR, Singapore

Abstract

In this report, we present our investigations on feature representation, hand mask modality, past action prediction, and model ensemble, for the EPIC-Kitchens Action Anticipation Challenge. Building upon an existing action anticipation model, i.e., RULSTM, our framework effectively utilizes enhanced feature representation, gives more emphasis on many-shot objects, and incorporates additional hand mask modality. We also explore a network modification to capture past action prediction. Furthermore, to exploit all the training data and aggregate the complementary information from different models, we employ model ensemble. We achieved top-1 action anticipation accuracy of 16.02% for Seen Kitchens (S1), and 10.11% for Unseen Kitchens (S2). Our submission, under the team name VI-I2R, achieved 2nd place for both seen and unseen kitchens, in terms of top-1 action anticipation accuracy.

1. Introduction

EPIC-Kitchens [1] is a large-scale video benchmark, capturing daily action in-situ from egocentric perspective. Presenting challenges such as natural multi-tasking interactions, dynamic scene changes, long tail class distribution, unseen test environment, EPIC-Kitchens aims to push boundaries in fine-grained video understanding. There are three benchmarking challenges over seen and unseen kitchens: object detection, action recognition and action anticipation. This report details our investigation and approach for the action anticipation challenge. Given an anticipation time, set as 1 second before the anticipated action starts, the action anticipation challenge is to classify the future action into its action class (i.e., the pair of verb and noun classes).

The main contributions of this report are our investigations on feature representation, hand mask modality, past action prediction, and model ensemble. Our investigations are based upon an existing action anticipation model. We examine how feature representation affects model performance, as well as investigate additional hand mask modality. To better model the dependencies of past and future actions, we also propose a network modification to jointly capture past action prediction and investigate its effect on action anticipation. Furthermore, to boost the predictive performance from the constituent models, we propose and exploit model ensemble. Our experimental results show that, while improvements to feature representations and modalities bring about performance improvement, model ensemble, which effectively aggregates the constituent models lead to the largest performance boost.

This report is organized as follows. An overview of our framework is introduced in Section 2. Subsequently, feature representations, hand mask modality, past action prediction, and model ensemble are presented in Section 3. The experimental results are provided in Section 4, and conclusion is discussed in Section 5.

2. Framework

As shown in Figure 1, our framework is built upon an existing action anticipation model, Rolling-Unrolling LSTM (RULSTM) [2], which has shown promising performance on the EPIC-Kitchens dataset. The RULSTM model, as proposed by Antonino and Giovanni, consists of two LSTMs - the Rolling LSTM (R-LSTM) that encodes past action, and the Unrolling LSTM (U-LSTM) that predicts future action. The model takes in a snippet of 14 frames before the start of the anticipated action, where the first six frames are used for encoding and the subsequent eight frames are used for prediction. We use time step of 0.25 second following the original work.

Existing input modalities for RULSTM consist of spatial, motion and object representations. The spatial and motion features are extracted from the pre-trained action recognition temporal segment network (TSN) model (with BN-Inception as base model) [3] on the RGB frames and optical flow respectively. The outputs of the extracted features are represented in 1024-D vectors. On the other hand, the object feature is represented by its detection confidence score, obtained from the pre-trained Faster R-CNN model (with ResNet-101 as backbone) [4], in a 352-D vector. The prediction scores for each modality are then fused with a weighted attention mechanism to obtain the final prediction.

3. Methodology

We investigate and improve on feature representation of existing modalities, as well as additional hand mask modality. Furthermore, we propose and explore modified

network with past action prediction. Finally, we employ model ensemble whereby the best validated models from the investigations are selected for the ensemble to obtain fused scores as the final prediction for the test set.

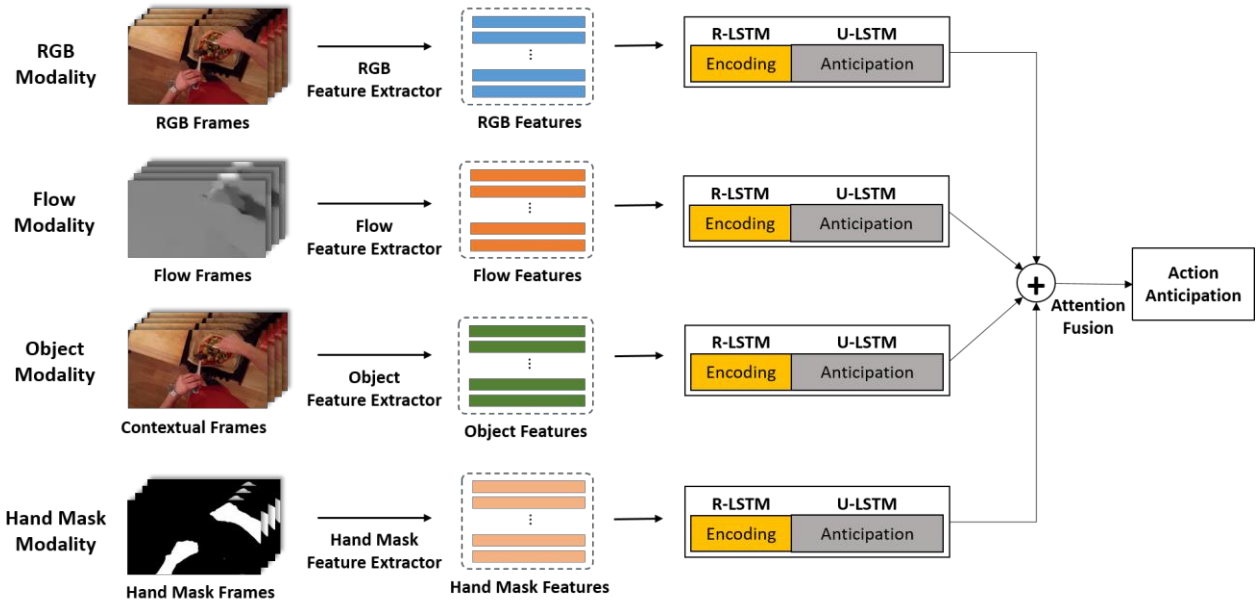


Figure 1: Overview of our framework with four input modalities: RGB, flow, object and hand mask. Prediction scores from each branch are fused for the final prediction.

3.1. Feature Representation

To improve the spatial and motion feature representation, we investigate other pre-trained action recognition models for feature extraction, temporal relation network (TRN) [5] and temporal shift module (TSM) [6], as provided by the challenge organizers [7]. We train the RGB and flow modalities following the implementation of RULSTM, where each modality is pre-trained with sequence completion, followed by fine-tuning for action anticipation.

We further investigate the performance of the Faster R-CNN object detection model. As most of the active objects are those that are defined as the many-shot object classes, we propose to train our own detection model focusing on only the many-shot objects. To this end, the Faster R-CNN model is adopted to detect the many-shot objects present in the scene and to extract the corresponding object features. Specifically, we use the Faster R-CNN (with ResNet-101 as backbone) pre-trained on the ImageNet and train the model on the EPIC-Kitchens object detection dataset with an initial learning rate of 0.003 for 10 epochs and 0.0003 for another 5 epochs. The many-shot object features are extracted in the similar way as [2]. We generate the object features by fusing the results from the object detector for all classes and that for the many-shot classes. As the maximum fusion method outperforms the average fusion and concatenation method, we use the maximum fusion for the final model ensemble.

3.2. Hand Mask Modality

Apart from the existing modalities on RGB, optical flow and object representation, we also consider the hand's positional information as another important feature that draws attention to the active action and object. Existing work by Liu et al. [8] also shares similar intuition and proposed to train a model that jointly predicts future hand motion, interaction hotspots, as well as the anticipated action class. Since the ground truth annotations for both the hand trajectories and hand-object interaction regions are not readily available, labour intensive manual annotations are required to allow supervised training. Different from the existing approach, we mainly focus on the surrounding region of the hands, and do not require manual annotation for our model training.

We utilize an existing pre-trained hand segmentation model and apply it to the EPIC-Kitchens dataset to extract hand segmentations. We adopt the encoder-decoder based deep neural network proposed in [9] to segment the hand regions from the complex background in the videos. The encoder is SeResNeXt50 (32×4d) [10] pre-trained on the ImageNet dataset and the decoder is Feature Pyramid Network (FPN) aiming to utilize the features at multiple scales for more robust hand mask prediction. We train this network on the Extended GTEA Gaze+ dataset [11], which contains cooking activities from 86 unique sessions of 32 subjects. The model is trained with an initial learning rate of 0.001 for 60 epochs and 0.0001 for another 40 epochs. With the trained model, we extract the hand masks of all the frames.

Figure 2: Illustration of masked RGB with transformed hand mask segmentation.

To generate the surrounding region of the segmented hands, we first dilate the hand mask with a kernel size of 80, followed by applying a Gaussian filter of size 40. This outputs a weighted mask where higher attention is given to regions nearer to the hands, while further boundaries have lower attention. The output mask is then mapped onto its corresponding RGB frame such that the surrounding scene and potential active objects that appear near the hands can be observed, as shown in Figure 2. We then encode these hand-attended regional frames using the same approach as our RGB feature extraction. The extracted hand-masked feature, represented in 1024-D, is added as the fourth input modality for our framework.

3.3. Past Action Prediction

In this section, we propose to improve the network for better learning on past action dependencies. As understanding past actions and predicting future events are both important tasks for action anticipation, we propose to modify the existing network to perform multi-task classification, which jointly predicts both past and future action classes.

To obtain the annotations for the past action labels, we assume the segmented actions in each video are in sequential order, and the past action happens within the range of the encoding frames (3.5seconds before the start of the anticipated action). For each video, the first action segment has the same annotation for its past and future class labels. This assumption allows us to have past action labels for all the video segments to ease our model training. However, in the actual case, some of the action segments could be overlapped, not in sequential order or the gap between two segmented actions could be far apart.

For multi-task classification, the first six encoding frames are used for past action prediction, while the subsequent eight anticipated frames are used for predicting anticipated action. We design the model to learn different past action label for each modality. The spatial branch is used to predict the past action, the motion branch is used for past verb prediction, while the object branch is used for past noun prediction. The prediction scores for each modality are then fused with an attention weight when training with the fusion model. During training, we optimize the weighted loss function of the past action and the anticipated action predictions. The weights are defined according to the proportion of the encoding frames and anticipated frames, which are 6/14 and 8/14 respectively. An overview of our improved architecture with past action prediction is shown in Figure 3.

3.4. Model Ensemble

To fully utilize all the training data and achieve effective model selection, we generate five sets of training/validation splits and train one model on each set. For testing, we carry out model ensemble by averaging the prediction scores from each of the five models. Meanwhile, to take advantage of the complementary information learned from different models, we ensemble the results from models trained with different features and networks using averaging and ranking. For ranking, final score S is calculated as:

$$S = \sum_{i=1}^{n} \frac{W_i}{Rank_i},$$

where W_i is the weight assigned to model i, $Rank_i$ is the class ranking number of model i, and n is the number of models for ensemble. In this report, W_i is set as 1.

4. Experiments

4.1. Feature Representation

In our experiment, we study the performance of the RGB and flow modalities using features extracted from pre-trained multi-scale TRN (M-TRN) (base model BN-Inception) and TSM (base model ResNet-50). We use the same training/validation split as RULSTM and present the comparison of the top-5 action class accuracy in Table 1. As M-TRN can effectively encode frame relation at multiple time scales, it provides more discriminative representation for action anticipation. Hence, we choose to replace the feature extractor of RGB and optical flow to M-TRN, while preserving the input dimension of 1024.

4.2. Hand Mask Extraction

For our hand mask experiment, we tested on features extracted from the masked RGB after applying the hand mask transformation. The masked RGB with original hand

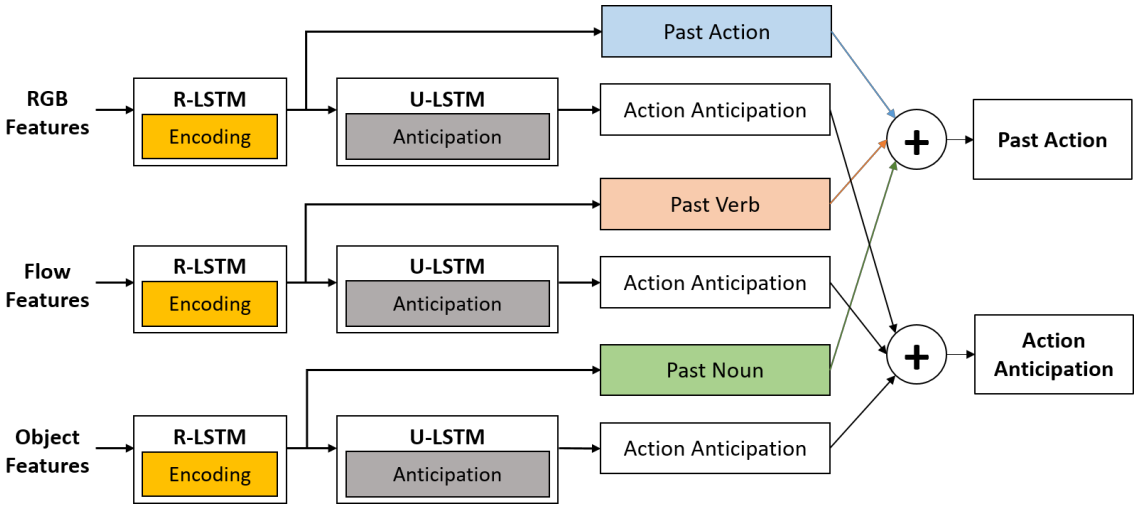


Figure 3: Improved architecture for multi-task learning. Each modality is trained to predict past action/verb/noun and action anticipation.

Table 1: Comparison of top-5 action class accuracy for different feature extractors: TSN, M-TRN and TSM.

Feature Extractor	Base model	Input size	Modality		Fusion
			RGB	Optical flow	
TSN (baseline)	BN-Inception	1024	30.67	21.30	34.35
M-TRN	BN-Inception	1024	**35.10**	**21.70**	**37.59**
TSM	Resnet-50	2048	16.45	10.46	30.69

mask is also presented as a baseline for comparison. Table 2 presents the top-5 action class accuracy for the validation set using different masked RGB features.

The model fusion takes four modalities: TSN RGB, TSN flow, object and hand mask. From the presented results, we observe that applying larger kernel sizes for dilation and Gaussian filtering perform better than the original masked RGB. This may be attributed to the larger seen area surrounding the hands, which brings more attention to current active action and nearby potential active objects.

Table 2: Validation results for masked RGB with hand mask.

Masked RGB	Dilation kernel	Gaussian kernel	Single modality	Fusion
Handmask_ original	-	-	16.71	36.87
Handmask_ d40	40	-	26.17	37.01
Handmask_ d40g20	40	20	26.83	37.33
Handmask_ d80g40	80	40	**30.29**	**37.71**

4.3. Past Action Prediction

Our improved architecture with past action prediction is trained using the same strategy as the original network. To compare its performance with the existing model, we use the same feature extractor as in Table 1. Each branch is trained to jointly predict the past action label (verb/ noun/ action) and action anticipation. The validation results are shown in Table 3. Similar to the results shown in Table 1, M-TRN feature performs the best among the three feature extractors. In overall, our improved architecture shows slight improvements over the baseline as seen in the fusion results across the three models.

4.4. Model Ensemble

The models trained under different settings are fused using two different model ensemble techniques: averaging and ranking. Specifically, four models are trained:

- **Model A:** the baseline network trained using baseline RGB, flow and object features;
- **Model B**: the baseline network trained using our improved RGB, flow and object features;
- **Model C**: the baseline network trained using our improved RGB, flow, object and hand mask features;
- **Model D**: our improved network trained with the baseline RGB, flow and object features.

The ensemble models are submitted to the server for evaluation on the testing dataset.

Table 3: Validation results of our improved model trained on RGB, flow and object to jointly learn past action and action anticipation.

Feature Extractor	Base model	Input size	Modality			Fusion
			RGB (Past action)	Optical flow (Past Verb)	Object (Past Noun)	
TSN	BN-Inception	1024	31.54	20.62	30.25	35.28
M-TRN	BN-Inception	1024	**35.60**	**21.48**	30.25	**37.77**
TSM	ResNet-50	2048	14.44	10.30	30.25	31.28

Table 4: Anticipation accuracies of the ensembles of different models on the testing dataset, including two different settings: Seen Kitchens (S1) and Unseen Kitchens (S2).

Models	Ensemble Method	Top-1 Accuracy						Top-5 Accuracy					
		Seen Kitchens			Unseen Kitchens			Seen Kitchens			Unseen Kitchens		
		Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action	Verb	Noun	Action
A, B	Averaging	36.72	24.61	16.01	28.71	17.21	10.04	80.39	54.90	**37.47**	71.77	40.49	23.69
A, B	Ranking	36.72	24.61	16.02	28.71	17.21	**10.11**	80.39	54.90	37.11	71.77	40.49	23.46
A, C	Ranking	36.72	24.37	**16.03**	28.88	16.83	9.94	80.48	55.04	37.34	71.22	40.22	23.49
B, D	Ranking	36.47	24.64	15.71	29.56	16.73	9.97	80.37	55.09	37.27	71.49	40.36	**23.73**

The results are shown in Table 4. For simplicity, we compare the performance of different ensembles in terms of the top-1 accuracy on the testing set. From the table, the ensemble of models A and B with ranking gives slightly better results than that with averaging on both seen and unseen kitchens settings. Compared with the ensemble of models A and B with ranking, the ensemble of models A and C with ranking gives 0.01% performance gain in seen kitchens setting while much lower accuracy in unseen kitchens setting. We also submitted the ensemble of models B and D with ranking. However, it produces the worst results among the four ensembles. In summary, the best model is the ensemble of models A and B with ranking and therefore we select it as our final submission to the leaderboard.

5. Conclusion

In this report, we investigated the improvement of the RULSTM architecture for action anticipation task in three aspects, namely, enhanced feature representation, additional hand modality, and incorporation of past action prediction. Furthermore, we employed model ensemble to make full use of the training data, as well as, combine the complementary information from different models. Experimental results show that significant performance improvement can be achieved for individual modalities compared to the baseline, although the degree of improvement tends to diminish after the fusion of multiple modalities. With further performance boost by model ensemble that effectively aggregates the constituent models, our method achieves highly competitive results on the leaderboard.

Acknowledgments

We would like to thank Dr. Liyuan Li and Dr. Joo Hwee Lim for their support, constructive suggestions and insightful guidance. This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project #A18A2b0046) and the National Research Foundation, Singapore under its NRF-ISF Joint Call (Award NRF2015-NRF-ISF001-2541).

References

[1] Damen D, Doughty H, Farinella G M, Fidler S, Furnari A, Kazakos E, Moltisanti D, Munro J, Perrett T, Price W, Wray M. The EPIC-KITCHENS Dataset: Collection, Challenges and Baselines[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI). 2020.

[2] Furnari A, Farinella G M. What would you expect? Anticipating egocentric actions with rolling-unrolling lstms and modality attention[C]. Proceedings of the IEEE International Conference on Computer Vision. 2019: 6252-6261.

[3] Wang L, Xiong Y, Wang Z, et al. Temporal segment networks for action recognition in videos[J]. IEEE transactions on pattern analysis and machine intelligence.

[4] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[C]. Advances in neural information processing systems. 2015: 91-99.

[5] Zhou B, Andonian A, Oliva A, et al. Temporal relational reasoning in videos[C]. Proceedings of the European Conference on Computer Vision (ECCV). 2018: 803-818.

[6] Lin J, Gan C, Han S. TSM: Temporal Shift Module for Efficient Video Understanding[C]. IEEE/CVF International Conference on Computer Vision (ICCV). 2019: 7082-7092.

[7] Price W, Damen, D. An Evaluation of Action Recognition Models on EPIC-Kitchens. https://github.com/epic-kitchens/action-models. 2019.

[8] Liu M, Tang S, Li Y, Rehg J. Forecasting Human Object Interaction: Joint Prediction of Motor Attention and Egocentric Activity[C]. IEEE/CVF International Conference on Computer Vision (ICCV). 2019.
[9] Kirillov A, He K, Girshick R, et al. A unified architecture for instance and semantic segmentation [J]. 2017.
[10] Hu J, Shen L, Sun G. Squeeze-and-excitation networks[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 7132-7141.
[11] Li Y, Liu M, Rehg J M. In the eye of beholder: Joint learning of gaze and actions in first person video[C]. Proceedings of the European Conference on Computer Vision (ECCV). 2018: 619-635.

Egocentric Object Manipulation Graphs

Eadom Dessalene Michael Maynord Chinmaya Devaraj Cornelia Fermüller
Yiannis Aloimonos
University of Maryland, College Park
{edessale,maynord,chinmayd,yiannis}@cs.umd.edu, fer@umiacs.umd.edu

Abstract

We introduce Egocentric Object Manipulation Graphs (Ego-OMG) - a novel representation for activity modeling and anticipation of near future actions integrating three components: 1) semantic temporal structure of activities, 2) short-term dynamics, and 3) representations for appearance. We evaluate Ego-OMG on the EPIC Kitchens Action Anticipation Challenge, demonstrating state-of-the-art performance and outranking all other previous published methods by large margins. We rank first on the unseen test set and second on the seen test set of the EPIC Kitchens Action Anticipation Challenge. We attribute the success of Ego-OMG to the modeling of semantic structure captured over long timespans. We note this technical report is an extract from [4].

1. Introduction

Most work on action understanding has centered on the task of action recognition. We instead work on the task of action anticipation: the task of classifying future actions from observations preceding the start of the action by a set anticipation time.

Our proposal integrates three necessary components for the anticipation of near future actions: 1) *semantic temporal structure*, 2) *short-term dynamics*, and 3) representations for *appearance*. *Semantic temporal structure* is modeled through a graph, embedded through a Graph Convolutional Network, whose states model characteristics of and relations between hands and objects. These state representations derive from all three levels of abstraction, and span segments delimited by the making and breaking of hand-object contact. *Short-term dynamics* are modeled in two ways: A) through 3D convolutions, and B) through anticipating the spatiotemporal end points of hand trajectories, where hands come into contact with objects. *Appearance* is modeled through deep spatiotemporal features produced through existing methods. We note that in Ego-OMG it is simple to swap these appearance features, and thus Ego-OMG is complementary to most existing action anticipation methods.

2. Joint Action Anticipation Architecture

The architecture of Ego-OMG is shown in Figure 2. Input consists of a single clip spanning 60 seconds - or 900 frames. This clip spans from start time $\tau_s - 60$ seconds to end time τ_s seconds, where τ_s lies τ_a seconds before the start of the action to be anticipated. The output consists of a logit layer predicting the class of the action that begins τ_a seconds after the end of the observation. The architecture is comprised of two streams: One modeling the appearance and short term dynamics of the last few seconds of the clip; the other modeling hand dynamics and long-term semantic temporal relations.

In the first stream, we model appearance and short-term dynamics with a Channel-Separated Convolutional Network (CSN), a 3D CNN factorizing 3D convolutions in channel and space-time in similar fashion to Xception-Net [3] which factorizes 2D convolutions in channel and space. The weights are pre-trained on the largescale IG-65M video dataset [7]. The network takes as input 32 consecutive frames of size $T \times 256 \times 256$, where $T = 15$ is the number of frames and 256 is the height and width of the cropped inputs. We apply horizontal flipping, color jittering and random crops during training, with centered crops during testing. The model is trained using SGD with a batch size of 16, a learning rate of 2.5×10^{-3} and a momentum of 0.9.

In the second stream we model dynamics of interactions between hands and objects, as well as longer term temporal semantic relations between the actions of the activity. We capture this structure in the form of a graph, described in detail in Section 3. After computing the graph over the entire EPIC Kitchens dataset, we feed it through two graph convolution layers of hidden layer size 256 and 128 respectively. The GCN training achieves fast convergence, reaching peak validation accuracy after 5 epochs or roughly 0.25 hours of training on a NVIDIA GeForce GTX 1080 GPU. At test time, we convert an input video of 900 frames to a sequence of states and from each state's respective node em-

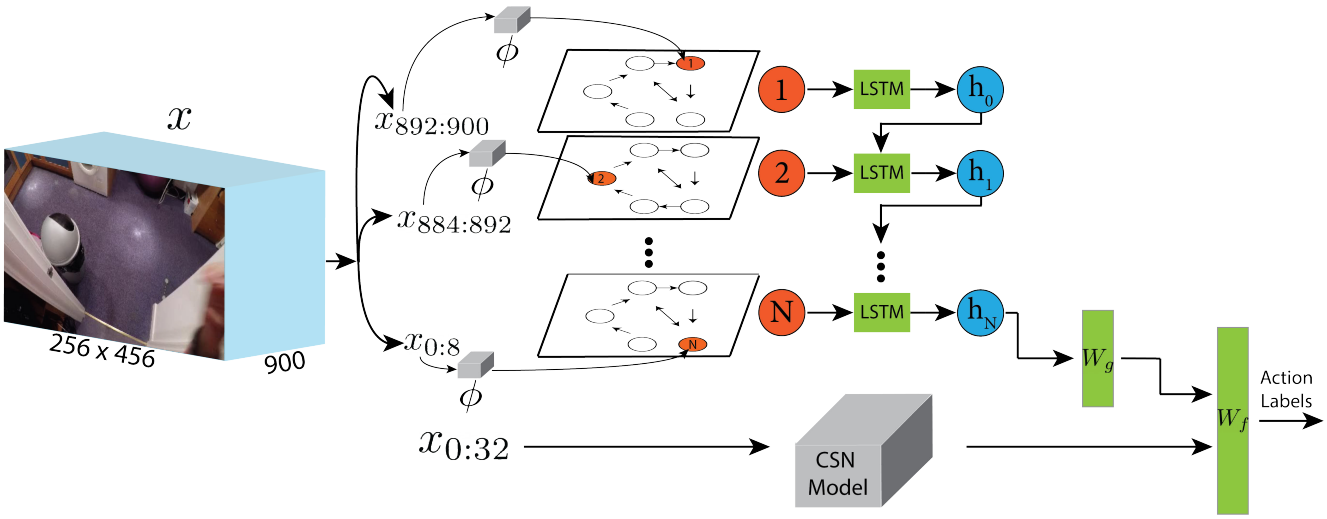


Figure 1. Overview of Ego-OMG's architecture. Ego-OMG consists of two streams: 1) The top stream consists of the extraction of a discretized sequence of states from an unconstrained egocentric video clip x of 900 frames using the Contact Anticipation Network ϕ. The nodes predicted by ϕ are embedded through GCN layers and then fed to an LSTM. This is then followed by a 1-layer MLP W_g to generate softmax scores for the anticipated future action. 2) The second stream generates softmax scores for the anticipated future action through feeding a short history (the last 32 frames of x) of video to a CSN model. A 1-layer MLP W_f processes the concatenated L2-normalized softmax scores to perform action anticipation.

bedding g_n for $n \in N$, we aggregate the state history with a 1-layer LSTM. From the final hidden state h_N, we apply a 1-layer MLP W_g to classify the next most likely action. We feed the sequence of node embeddings into an LSTM [8]. The LSTM carries hidden states of size 128. A batch size of 16 and a learning rate of 7×10^{-5} is used with ADAM optimizer.

We concatenate the L2-normalized softmax scores from each respective stream, freezing the two sub-networks while training a 1-layer MLP W_f with a batch size of 16 and learning rate of 0.01 on top of the joint softmax scores to classify the next most likely action. We find a late fusion approach provides slight benefits in practice as opposed to an early fusion of the two streams, likely due to the different learning dynamics of the individual streams.

3. Structured Graph of Egocentric Activity

Action anticipation involves reasoning about complex contextual and temporal semantic relations. We design a graph based representation to capture these relations.

The structure of an egocentric activity video dataset is consolidated into graph G, from which we can retrieve a high level sequence of node states $S_i = \{S_1, S_2, ..., S_n\}$ extracted from video V_i at test time, over which we can perform soft reasoning to anticipate the next most likely action.

Given the egocentric setting and the general observation that hands are the central driving force of change in object manipulation activities, graph states represent *contact* and *anticipated contact* relations between hands and objects, where each hand is represented independently. This enables us to more finely model tasks requiring complex bi-manual manipulation.

3.1. Contact Anticipation

As illustrated in Figure 3 part b), we feed x_t into a *Contact Anticipation Network* [1] ϕ, whose purpose is to anticipate hand object contacts. Anticipated contact is represented through a 4 channel output, consisting of object segmentation masks $\{\Psi_{t_r}, \Psi_{t_l}, \Gamma_{t_r}, \Gamma_{t_l}\}$, where Ψ_{t_r} and Ψ_{t_l} denote the next active object predictions, and Γ_{t_r} and Γ_{t_l} denote the objects detected to be presently in contact with the hand, both for the right and left hands respectively. We feed each segmentation frame to an object classifier, arriving at predicted object classes $o_t = \{\psi_{t_r}, \psi_{t_l}, \gamma_{t_r}, \gamma_{t_l}\}$. We note that for the purposes of this work we predict up to 1 object each for ψ_{t_r}, ψ_{t_l}, γ_{t_r}, and γ_{t_l}. This limitation prevents us from modelling scenarios where multiple objects are held by the same hand for tasks requiring dexterous manipulation (though, these are uncommon scenarios).

The contact anticipation network ϕ is a 3D ConvNet video object segmentation architecture that makes use of additional supervisory signals corresponding to time of progression of directed hand movements along with ground truth segmentation masks of objects of interaction. It takes as input x_t with 8 frames of video along with a self-generated history of contact anticipation maps, a fine-grained representation of *time-to-contact* between the hands and each pixel belonging to the environment. [1] further de-

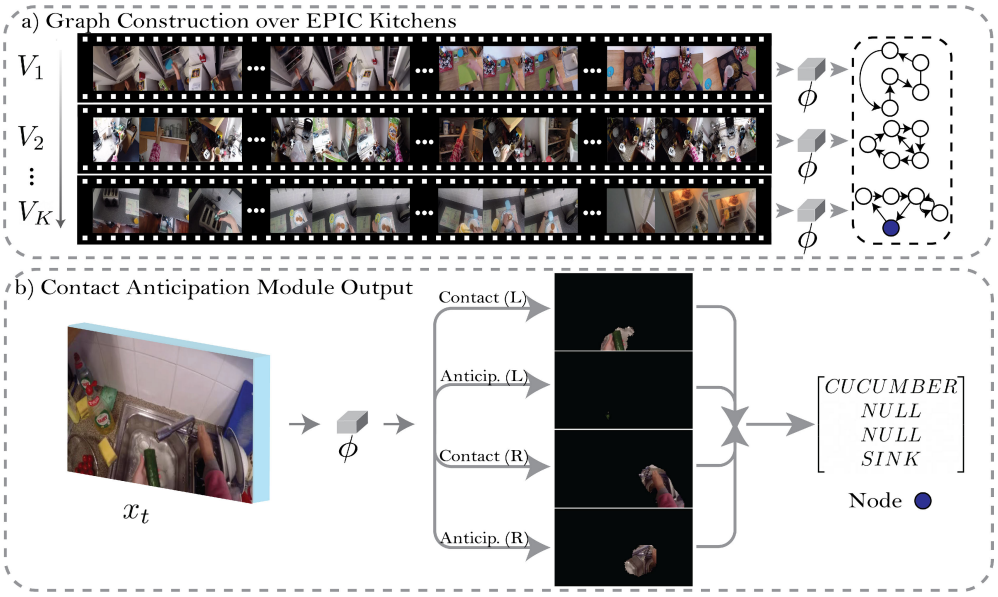


Figure 2. Overview illustrating the construction of a consolidated graph G of hand-object interactions over a set of K egocentric unconstrained videos. In (a), the Contact Anticipation Network ϕ is applied in a sliding window fashion over each video V_k, whose outputs are then classified w.r.t. object classes, and then consolidated into a single graph. (b) illustrates ϕ applied to a single window x_t of 8 frames where the output of the anticipation network is shown as 4 separate binary segmentation masks representing the pixels belonging to the objects in contact with the left and right hand, and the pixels belonging to the objects anticipated to come into contact with the left and right hand. Each segmentation channel is fed to a classifier, and the resulting tuple is represented as a node in the graph.

scribes the details of how the contact anticipation module is trained.

In practice, while the contact anticipation module succeeds at localizing contacted objects, the classifier tends to mis-classify currently held objects due to the severe occlusion imposed by the hand, especially for small objects like scissors and utensils. Therefore, in building the graph we impose the constraint that every object currently contacted by each hand *must have been anticipated* at some previous instance in time, before the presence of occlusion. In classifying the objects currently in contact with the hand, we take the intersection of top-5 object class predictions for that object with the object classes previously predicted in anticipation within the past 7 seconds.

3.2. Graph Construction

As illustrated in Figure 3 part a), we have a set of K training videos, $V = \{V_1, V_2, ..., V_K\}$, where $K = 272$ as there are 272 recorded videos in the EPIC Kitchens dataset. To detect the objects involved in interaction, which are needed to build the graph, we utilize the Contact Anticipation Network, ϕ, described in subsection 3.1. The network ϕ iterates over each video $V_i \in V$ using a sliding window with an 8-frame width, and a stride of 2. Feeding each of 4 output channels of ϕ to the object classifier then produces detections $O_i = \{o_1, o_2, ..., o_{T_i/2}\}$ for V_i, where T_i is the frame count of V_i. From the per-frame predictions of the object classes o_i, we suppress consecutive duplicate predictions, where $S_k = \{o_j : j = T_i/2 - 1 \vee o_j \neq o_{j+1}\}$ for $0 \leq j < T_i/2$, arriving at non-repeating states $S_k = \{s_1, s_2, ..., s_n\}$, an ordered set where temporal order is preserved.

With the input to graph construction defined, we now

	Method	Top-1	Top-5
S1	2SCNN (RGB) [11]	4.32	15.21
	TSN (RGB) [12]	6.00	18.21
	TSN + MCE [5]	10.76	25.27
	RULSTM[6]	15.35	35.13
	Camp. et al[2]	15.67	**36.31**
	Liu et al[9]	15.42	34.29
	Ego-OMG	**16.02**	34.53
S2	2SCNN (RGB) [11]	2.39	9.35
	TSN (RGB) [12]	2.39	9.63
	TSN + MCE [5]	5.57	15.57
	RULSTM[6]	9.12	21.88
	Camp. et al[2]	9.32	23.28
	Liu et al[9]	9.94	23.69
	Ego-OMG	**11.80**	**23.76**

Table 1. Action anticipation results on the EPIC Kitchens test set for *seen* kitchens (**S1**) and *unseen* kitchens (**S2**) during the EPIC Kitchens Action Anticipation Challenge. Only published submissions are shown.

consider the graph $G = (V, E)$, where E consists of the set of all edges, and V consists of the set of all nodes. $V = \{V_s, V_a\}$ consists of nodes of two types: state nodes, and action nodes. State nodes consist of the intersection of all S_k, that is: $V_s = \bigcap_{k=1}^{K} S_k$, and action nodes V_a consist of the set of all action classes $a_i \in A$, where A is the set of all actions. In doing so, we represent both states and actions in graph G.

We construct the adjacency matrix as follows. Each node has an edge connecting it to itself: $e_{ii} \in E$ for $1 \leq i \leq |V|$ with weight 1. We add weighted directed edges $e_{ij} \in E$ for consecutive states s_i and s_j for $0 \leq i < n$ and $j = i + 1$, where the weight σ_{ij} is transition probability $p(s_{i+1}|s_i)$. We also add weighted directed edges between states and actions by adding weighted edge $e_{ij} \in E$ if action i takes place within the timespan of state s_j, where weight σ_{ij} is equal to $p(a_i|s_j)$.

Graph G has a total number of nodes equal to the number of unique states $z = |S| + |A|$, where S is the set of unique states and A is the set of annotated actions. Let $X \in R^{z \times m}$ be a matrix containing all z nodes with their corresponding features of dimension m. Rather than set X to identity matrix I, we initialize each node with feature embeddings extracted from a pre-trained GloVe-600 model [10]. When representing states $s \in S$, we average the feature embeddings from each object noun in s. When representing actions $a \in A$, we average the embeddings for the verb and noun embeddings. We find that utilizing pretrained word embeddings for G results in substantial performance gains over using $X = I$.

To allow the application of Graph Convolutions over nodes extracted from the test set, we repeat the node extraction process over test videos, only incorporating nodes not already included in the graph constructed from training set videos. To prevent the graph convolutions from attending to statistics inherent in the test set, we set the transition probabilities of each node extracted from the test set as $\sigma_{ij} = 1/deg(s_i)$ for $s_i \in V_s$ and node s_j extracted from the test set, incorporating node s_j into V_s.

We feed the weighted adjacency matrix and X as input into the GCN as described in Section 2.

4. Conclusion

We have introduced Ego-OMG, a novel action anticipation method including representations for all of semantic temporal structure of activities, short term dynamics, and appearance of actions. Ego-OMG makes use of a novel graph representation capturing action structure at all three levels, whose nodes are embedded into a natural vector representation through use of a Graph Convolutional Network. This graph constitutes the core component of the first stream of Ego-OMG. Ego-OMG's second stream consists of CSN features. This second stream is easily swappable with other representations, making Ego-OMG complementary to many existing alternative action anticipation approaches.

References

[1] A. Anonymous. Contact anticipation maps for forecasting hand object interaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020 under review.

[2] Guglielmo Camporese, Pasquale Coscia, Antonino Furnari, Giovanni Maria Farinella, and Lamberto Ballan. Knowledge distillation for action anticipation via label smoothing. *arXiv preprint arXiv:2004.07711*, 2020.

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017.

[4] Eadom Dessalene, Michael Maynord, Chinmaya Devaraj, Cornelia Fermuller, and Yiannis Aloimonos. Egocentric object manipulation graphs. *arXiv preprint arXiv:2006.03201*, 2020.

[5] Antonino Furnari, Sebastiano Battiato, and Giovanni Maria Farinella. Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 389–405, 2018.

[6] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6252–6261, 2019.

[7] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12046–12055, 2019.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[9] Miao Liu, Siyu Tang, Yin Li, and James Rehg. Forecasting human object interaction: Joint prediction of motor attention and egocentric activity. *arXiv preprint arXiv:1911.10967*, 2019.

[10] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[11] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1049–1058, 2016.

[12] 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.

Semantic Label Smoothing EPIC-Kitchens Action Anticipation Challenge 2020 Team UNIPD-UNICT

Guglielmo Camporese*, Pasquale Coscia*, Antonino Furnari†, Giovanni Maria Farinella†, Lamberto Ballan*
*Department of Mathematics "Tullio Levi-Civita", University of Padova, Italy
†Department of Mathematics and Computer Science, University of Catania, Italy

***Abstract*—In this report we describe the technical details of our submission to the EPIC-Kitchens 2020 action anticipation challenge. We propose an extension of the label smoothing technique that integrates the semantic content of the action in the target labels. This procedure can be seen as a knowledge distillation process where the teacher injects semantic information of the labels to the student network. We apply this technique to a state-of-the-art model and to a custom encoder-decoder network with multi-head attention modules. The entry of the challenge then, is an ensemble of these two models. Our label smoothing method achieves competitive performances respect to the state-of-the-art with a Top-1 action accuracy of 15.67% in the S1 dataset and a Top-1 action accuracy of 9.32% in the S2 dataset.**

I. Introduction

Human action analysis is a central task in computer vision that has a enormous impact on many applications, such as, video content analysis [1], [2], video surveillance [3], [4], and automated driving vehicles [5], [6]. Systems interacting with humans also need the capability to promptly react to the context changes, and plan their actions accordingly. Most previous works focus on the tasks of action recognition [7], [8], [9] or early-action recognition [10], [11], [12], i.e., recognition of an action after its observation (happened in the past) or recognition of an ongoing action from its partial observation (only part of the current action is available). A more challenging task is to predict near future, i.e., to forecast actions that will be performed ahead in time. Predicting future actions before observing the corresponding frames [13], [14] is demanded by many applications which need to anticipate human behaviour. For example, intelligent surveillance systems may support human operators to avoid hazards or assistive robotics may help non-self-sufficient people. Such task requires to analyze significant spatio-temporal variations among actions performed by different people. For this reason, multiple modalities (e.g., appearance and motion) are typically considered to improve the identification of similar actions. Egocentric scenarios provide useful settings to study early-action recognition or action anticipation tasks. Indeed, wearable cameras offer an explicit point-of-view to capture human motion and object interaction.

In this work, we address the problem of anticipating egocentric human actions in an indoor scenario at several time steps. More specifically, we anticipate an action by leveraging video segments that precede the action. We disentangle the processing of the video into encoding and decoding stages. In the first stage, the model summarizes the video content while in the second stage the model predicts at multiple anticipation times τ_a the next action (see Fig. 1). We exploit a recurrent neural network (RNN) to capture temporal correlations between subsequent frames and consider three different modalities for representing the input: appearance (RGB), motion (optical flow) and object-based features. An important aspect to consider when dealing with human action anticipation is that the future is uncertain, which means that different prediction of future actions are equally likely to occur. For example, the actions "*sprinkle over pumpkin seeds*" and "*sprinkle over sunflower seeds*" may be equally performed when preparing a recipe. For this reason, to deal with the uncertainty of future predictions, we propose to group similar actions comparing several label smoothing techniques in order to broaden the set of possible futures and reduce the uncertainty caused by one-hot encoded labels. Label smoothing is introduced in [15]

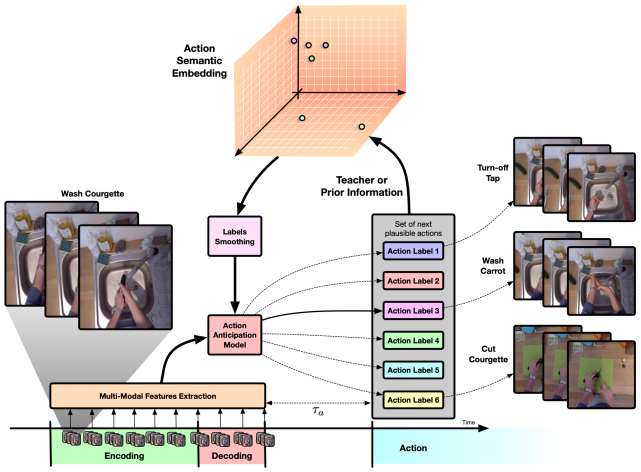


Fig. 1: Our framework for action anticipation. After an encoding procedure of the video, during the decoding stage our model anticipates the next action that will occur in τ_a seconds. Afterwards, a teacher model distills semantic information via label smoothing into the action anticipation model during training in order to reduce the uncertainty on the future predictions.

as a form of regularization for classification models since it introduces a positive constant value into wrong classes components of the one-hot target. A peculiar feature of such method is to make models robust to overfitting especially when labels in the dataset are noisy, e.g., the targets are ambiguous. In our work [16], we extend label smoothing by using them as a bridge for distilling knowledge into the model during training. Our experiments on the large-scale EPIC-Kitchens dataset show that label smoothing increases the performance of state-of-the-art models and yields better generalization on test data.

II. Proposed Approach

Anticipating human actions is essential for developing intelligent systems able to avoid accidents or guide people to correctly perform their actions. We study the suitability of label smoothing techniques to address the issue.

A. Label Smoothing

As investigated in [17], there is an inherent uncertainty on predicting future actions. In fact, starting from the current state observation of an action there can be multiple, but still plausible, future scenarios that can occur. Hence, the problem can be reformulated as a multi-label task with missing labels where, from all the valid future realizations, only one is sampled from the dataset. All previous models designed for action anticipation are trained with cross-entropy using one-hot labels, leveraging only one of the possible future scenario as the ground truth. A major drawback of using hard labels is to favour logits of correct classes weakening the importance of other plausible ones. In fact, given the one-hot encoded vector $y^{(k)}$ for a class k, the prediction of the model p and the logits of the model z such that $p(i) = e^{z(i)}/\sum_j e^{z(j)}$, the cross entropy is minimized only if $z(k) \gg z(i)\ \forall i \neq k$. This fact encourages the model to be over-confident about its predictions since during training it tries to focus all the energy on one single logit leading to overfitting and scarce adaptivity [15]. To this purpose, we smooth the target distribution enabling the chance of negative classes to be plausible. However, the usual label smoothing procedure introduces a uniform positive component among all the classes, without capturing the difference between actions. To this end, we propose several ways of designing such smoothing procedure by encoding semantic priors into the labels and weighting the actions according their feature representation. We can connect our soft labels approach to the knowledge distillation framework [18] where the teacher provides useful information to the student model during training. What differs is that the teacher does not depend on the input data but solely on the target, i.e., it distills information using ground-truth data. Since teacher's prediction is constant w.r.t. the input, such information can be encoded before training into the target via label smoothing.

As a form of regularization, [15] introduces the idea of smoothing hard labels by averaging one-hot encoded targets with constant vectors as follows:

$$y^{soft}(i) = (1-\alpha)y(i) + \alpha/K, \tag{1}$$

where y is the one-hot encoding, α is the smoothing factor ($0 \leq \alpha \leq 1$) and K represents the number of classes. Since cross entropy is linear w.r.t. its first argument, it can be written as follows:

$$\begin{aligned} CE[y^{soft}, p] &= \sum_i -y^{soft}(i)\log(p(i)) = \\ &= (1-\alpha)CE[y,p] + \alpha CE[1/K, p]. \end{aligned} \tag{2}$$

The optimization based on the above loss can be seen as a distillation knowledge procedure [18] where the teacher randomly predicts the output, i.e., $p^{teacher}(i) = 1/K,\ \forall i$. Hence, the connection with the distillation loss proves that the second term in Eq. (1) can be seen as a prior knowledge, given by an agnostic teacher, for the target y. Although using an agnostic teacher seems an unusual choice, uniform label smoothing can be seen as a form of regularization [15] and thus it can improve the model's generalization capability. Taking this into account, we extend the idea of smoothing labels by modeling the second term of Eq. (1), i.e., the prior knowledge of the targets, as follows:

$$y^{soft}(i) = (1-\alpha)y(i) + \alpha\pi(i), \tag{3}$$

where $\pi \in \mathbb{R}^K$ is the prior vector such that $\sum_i \pi(i) = 1$ and $\pi(i) \geq 0\ \forall i$.

Therefore, the resulting cross entropy with soft labels is written as follows:

$$CE[y^{soft}, p] = (1-\alpha)CE[y,p] + \alpha CE[\pi, p] \tag{4}$$

This loss not only penalizes errors related to the correct class but also errors related to the positive entries of the prior. Starting from this formulation, we introduce *Verb-Noun*, *GloVe* and *Temporal* priors for smoothing labels in the knowledge distillation procedure. In the following, we detail our label smoothing techniques.

Verb-Noun label smoothing. EPIC-KITCHENS [13] contains action labels structured as verbs-noun pairs, like "*cut onion*" or "*dry spoon*". More formally, if we define $\mathcal{A}$ the set of actions, $\mathcal{V}$ the set of verbs, and $\mathcal{N}$ the set of nouns, then an action is represented by a tuple $a = (v, n)$ where $v \in \mathcal{V}$ and $n \in \mathcal{N}$. Let $\mathcal{A}_v(\bar{v})$ the set of actions sharing the same verb $\bar{v}$ and $\mathcal{A}_n(\bar{n})$ the set of actions sharing the same noun $\bar{n}$, defined as follows:

$$\mathcal{A}_v(\bar{v}) = \{(\bar{v}, n) \in \mathcal{A}\ \forall\ n \in \mathcal{N}\}, \tag{5}$$

$$\mathcal{A}_n(\bar{n}) = \{(v, \bar{n}) \in \mathcal{A}\ \forall\ v \in \mathcal{V}\}, \tag{6}$$

where $\bar{n} \in \mathcal{N}$ and $\bar{v} \in \mathcal{V}$.

We define the prior of the k^{th} ground-truth action class as

$$\pi^{(k)}_{VN}(i) = \mathbb{1}\left[a^{(i)} \in \mathcal{A}_v(v^{(k)}) \cup \mathcal{A}_n(n^{(k)})\right]\frac{1}{C_k}, \tag{7}$$

where $a^{(i)}$ is the i^{th} action, $v^{(k)}$ and $n^{(k)}$ are the verb and the noun of the k^{th} action, $\mathbb{1}[\cdot]$ is the indicator function, and $C_k = |\mathcal{A}_v(v^{(k)})| + |\mathcal{A}_n(n^{(k)})| - 1$ is a normalization term.

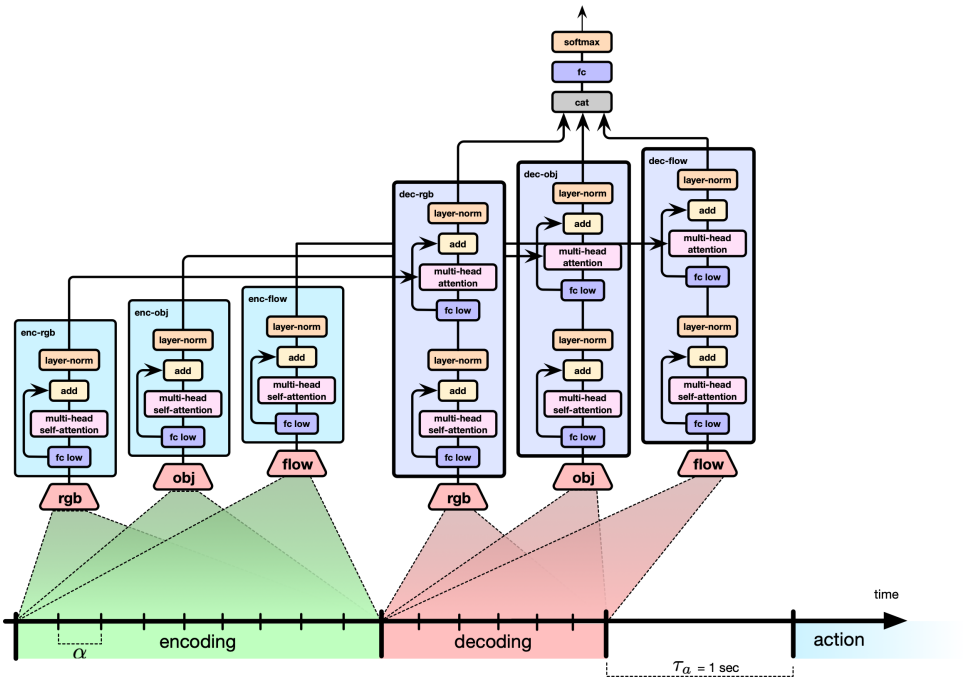


Fig. 2: Action anticipation protocol based on the encoding and decoding stages. We summarize past observations by processing video snippets sampled every $\alpha = 0.25$ seconds in the encoding stage. In the decoding stage instead, we start making predictions every α seconds to anticipate the future action. For the challenge we make only one prediction at anticipation time $\tau_a = 1$ second before the action starts.

Using such encoding rule, the cross entropy not only penalizes the error related to the correct class but also the errors with respect to all the other "*similar*" actions with either the same verb or noun[1].

GloVe based label smoothing. An important aspect to consider when dealing with actions represented by verbs and/or nouns is their semantic meaning. In the *Verb-Noun* label smoothing, we define the prior considering a rough yet still meaningful semantic where actions that share either the same verb or noun are considered similar. To extend this idea, we extrapolate the prior from the word embedding of the action. One of the most important properties of word embeddings is to put closer words with similar semantic meanings and to move dissimilar ones far away, as opposed to hard labels that cannot capture at all similarity between classes since $y^{(i)T} y^{(j)} = 0 \; \forall i \neq j$.

Using such action representation, we enable the distillation of useful information into the model during training since the cross entropy not only penalizes the error related to the correct class but also the error related to all other similar actions. In order to compute the word embeddings of the actions we use the GloVe model [19] pretrained on the Wikipedia 2014 and Gigaword 5 datasets. We use the GloVe model since it does not rely just on local statistics of words, but incorporates global statistics to obtain word vectors. Since the model takes as input only single words, we encode the action as follows:

$$\phi^{(k)} = Concat\left[GloVe(v^{(k)}), GloVe(n^{(k)})\right] \tag{8}$$

where ϕ is the obtained action representation of $a^{(k)} = (v^{(k)}, n^{(k)})$ and $GloVe(\cdot) \in \mathbb{R}^{300}$ is the output of the GloVe model. We finally compute the prior probability for smoothing the labels as the similarity between two action representations, which is computed as follows:

$$\pi^{(k)}_{GL}(i) = \frac{|\phi^{(k)T} \phi^{(i)}|}{\sum_j |\phi^{(k)T} \phi^{(j)}|}. \tag{9}$$

Hence, $\pi^{(k)}_{GL}(i)$ in Eq. (9) represents the similarity between the k^{th} and the i^{th} action.

Temporal label smoothing. Some actions are more likely to co-occur than others. Furthermore, only specific action sequences may be considered plausible. For this reason, it could be reasonable to focus on most frequent action sequences since they may reveal possible valid paths in the actions space. In this case, we build the prior probability of their observation by considering subsequent actions of length two, i.e., we estimate from the training set the transition probability from the i^{th} to the k^{th} action as follows:

$$\pi^{(k)}_{TE}(i) = \frac{Occ\left[a^{(i)} \rightarrow a^{(k)}\right]}{\sum_j Occ\left[a^{(j)} \rightarrow a^{(k)}\right]} \tag{10}$$

[1]It can be proved that in terms of scalar product two different classes having the same noun or verb and encoded with Verb-Noun label smoothing are closer respect to classes encoded with hard labels

where $Occ\left[a^{(i)} \rightarrow a^{(k)}\right]$ is the number of times that the i^{th} action is followed by the k^{th} action. Using such representation, we reward both the correct class and most frequent actions that precede the correct class.

B. Action Anticipation Architecture

For the challenge we apply our label smoothing procedure to RU-LSTM [14] and to a custom encoder-decoder model based on multi-head attention [20]. We follow the same protocol as in [14] for handling the video. We process the frames preceding the action that we want to anticipate grouping them into video snippets of length 5. Each video snippet is collected every $\alpha = 0.25$ seconds and processed considering three different modalities: *RGB* features computed using a Batch Normalized Inception CNN [21] trained for action recognition, *objects* features computed using Fast-R CNN [22] and *optical flow* computed as in [23], processed through a Batch Normalized Inception CNN trained for action recognition. Our multi-modal architecture, depicted in Fig. 2, processes the inputs and encompasses two building blocks: an encoder which recognizes and summarises past observations and a decoder which predicts future actions at different anticipation time steps. As shown in Fig. 2, during the encoding stage each modality is separately processed by a fully connected layer with 512 units, a multi-head self-attention module and a layer normalization [24]. In the decoding phase each modality is processed as depicted in Fig. 2 attending the respectively encoded sequence and then concatenated. The fused decoded activations are then passed to the final fully connected layer with softmax activation. For the challenge the decoding stage consists only in the frame collocated $\tau_a = 1$ second before the action we want to anticipate.

III. Results

A. Models and Baselines

In the comparative analysis, we exploit the architecture proposed in Sec. II-B employing the different label smoothing techniques defined in Sec. II-A. In our experiments we consider models trained using one-hot vectors (One Hot), uniform smoothing (Smooth Uniform), Verb-Noun soft labels (Smooth VN) and GloVe based soft labels (Smooth GL).

To implement the model architecture, we used PyTorch. Each model is trained for 30 epochs with batch size of 128 using Adam optimizer with learning rate of $lr = 3e{-}5$ for our custom encoder-decoder model and $lr = 2e{-}4$ for RU-LSTM. For each model we select the best parameter configuration using early stopping by monitoring the Top-1 and the Top-5 accuracy at anticipation time $\tau_a = 1$ on the validation set. For each label smoothing method we set the smooth factor $\alpha = 0.6$.

Results. Table I reports our results on the EPIC-KITCHENS validation set. We reported the Top-1 and Top-5 accuracy of all the models trained with the different label smoothing priors. We can see that models trained with GloVe label soft labels achieves better results for both Smoothed ED MHA and Smoothed RU-LSTM. We reported also the results of the ensembles made with the model candidates from $ID = 0$ to $ID = 15$. We perform ensambling by averaging the logits of the predictions on the validation set of the corresponding models. Table II reports the results of the model "Ensemble All T1" on the seen (S1) and unseen (S2) test kitchens dataset.

Finally, Fig. 3 depicts prior component representations of the proposed label smoothing procedures.

IV. Conclusion

This study proposed a knowledge distillation procedure via label smoothing for leveraging the multi-modal future component of the action anticipation problem. We generalized the idea of label smoothing by designing semantic priors of actions that are used during training as ground truth labels. We implemented a LSTM baseline model that can anticipate actions at multiple time steps starting from multi-modal representation of the input video. Experimental results corroborate out findings compared to state-of-the-art models highlighting that label smoothing systematically improves performance when dealing with future uncertainty.

Acknowledgements: Research at the University of Padova is partially supported by MIUR PRIN-2017 PREVUE grant. Authors of Univ. of Padova gratefully acknowledge the support of NVIDIA for their donation of GPUs, and the UNIPD CAPRI Consortium, for its support and access to computing resources. Research at the University of Catania is supported by Piano della Ricerca 2016-2018 linea di Intervento 2 of DMI and MIUR AIM - Attrazione e Mobilità Internazionale Linea 1 - AIM1893589 - CUP E64118002540007.

References

[1] L. Ballan, M. Bertini, A. Del Bimbo, L. Seidenari, and G. Serra, "Event detection and recognition for semantic annotation of video," *Multimedia tools and applications*, vol. 51, no. 1, pp. 279–302, 2011.

[2] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, Jun 2014.

[3] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, Jan 2013.

[4] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[5] A. Bhattacharyya, M. Fritz, and B. Schiele, "Long-term on-board prediction of people in traffic scenes under uncertainty," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[6] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[7] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

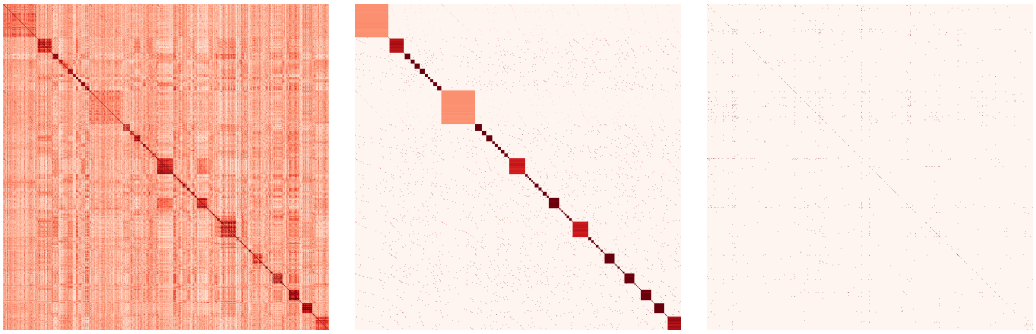
[8] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[9] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] M. S. Ryoo, "Human activity prediction: Early recognition of ongoing activities from streaming videos," in *IEEE International Conference on Computer Vision (ICCV)*, 2011.

ARCHITECTURE	ID	LABEL SMOOTHING KIND	EARLY STOPPING METRIC	TOP-1	TOP-5
Smoothed ED MHA	0	None	Top-1	14.31%	34.57%
	1	None	Top-5	14.09%	34.57%
	2	GloVe	Top-1	15.20%	34.71%
	3	GloVe	Top-5	15.02%	35.16%
	4	Verb-Noun	Top-1	14.36%	33.89%
	5	Verb-Noun	Top-5	14.23%	34.57%
	6	Uniform	Top-1	14.58%	34.89%
	7	Uniform	Top-5	14.58%	34.89%
Smoothed RU-LSTM	8	None	Top-1	15.31%	34.65%
	9	None	Top-5	15.33%	35.44%
	10	GloVe	Top-1	16.33%	35.48%
	11	GloVe	Top-5	15.55%	36.00%
	12	Verb-Noun	Top-1	15.59%	35.48%
	13	Verb-Noun	Top-5	15.12%	35.68%
	14	Uniform	Top-1	15.41%	35.32%
	15	Uniform	Top-5	15.16%	35.48%
Ensemble Only Top-1 (even ID) T1	{10}	{GloVe}	Top-1	16.33%	35.58%
Ensemble Only Top-1 (even ID) T5	{0, 2, 10, 14}	{None, GloVe, Uniform}	Top-5	16.05%	**37.54%**
Ensemble Only Top-5 (odd ID) T1	{1, 3, 9, 13, 15}	{None, GloVe, Verb-Noun, Uniform}	Top-1	16.15%	36.92
Ensemble Only Top-5 (odd ID) T5	{1, 5, 11, 13, 15}	{None, GloVe, Verb-Noun, Uniform}	Top-5	16.05%	37.28
Ensemble All T1	{2, 4, 8, 10, 12, 13, 15}	{None, GloVe, Verb-Noun, Uniform}	Top-1	**16.47%**	36.75%
Ensemble All T5	{0, 2, 10, 14}	{None, GloVe, Uniform}	Top-5	16.05%	**37.54%**

TABLE I: Top-1 and Top-5 accuracy for the action anticipation task at $\tau_a = 1$ second on the EPIC-KITCHENS validation set. We choose "Ensemble All T1" with model ID $\{2, 4, 8, 10, 12, 13, 15\}$ as the candidate model for the challenge since it achieves the highest Top-1 accuracy on the validation dataset.



(a) GloVe Smoothing Matrix. (b) Verb-Noun Smoothing Matrix. (c) Temporal Smoothing Matrix.

Fig. 3: Smooth Labels Matrices. Each matrix represents the prior component for the label smoothing procedure reported in Eq. (3) and each row corresponds to a single label prior. In the GloVe and Verb-Noun priors can be recognized squared structures on the diagonal because the labels are alphabetically ordered. The Temporal prior has sparse row entries since there is no major occurrence trend or structure in the training set.

		Top-1 Accuracy%			Top-5 Accuracy%			Avg Class Precision%			Avg Class Recall%		
	MODEL	VERB	NOUN	ACTION	VERB	NOUN	ACTION	VERB	NOUN	ACTION	VERB	NOUN	ACTION
S1	Ensemble All T1	36.73	24.26	15.67	79.87	53.76	36.31	35.86	25.16	7.42	14.12	21.30	7.62
S2	Ensemble All T1	28.51	16.59	9.32	71.66	37.97	23.28	13.15	13.26	4.72	7.86	13.77	5.07

TABLE II: Results of action anticipation on the test sets of EPIC-Kitchens. The test set is divided into kitchens already seen $S1$ or unseen $S2$ from the model.

[11] Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. Mark Siskind, and S. Wang, "Recognize human activities from partially observed videos," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2013.

[12] R. De Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars, "Online action detection," in *European Conference on Computer Vision (ECCV)*, 2016.

[13] 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 (ECCV)*, Sep 2018.

[14] A. Furnari and G. M. Farinella, "What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[16] G. Camporese, P. Coscia, A. Furnari, G. M. Farinella, and L. Ballan, "Knowledge distillation for action anticipation via label smoothing," 2020.

[17] A. Furnari, S. Battiato, and G. Maria Farinella, "Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation," in *European Conference on Computer Vision Workshops (ECCVW)*, 2018.

[18] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[19] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances*

in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, 2015.

[22] R. B. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[23] 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 (ECCV)*, 2016.

[24] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016. [Online]. Available: http://arxiv.org/abs/1607.06450

GT-UWISC-MPI Submission to Epic-Kitchens Action Anticipation Challenge 2020

Miao Liu[1,3,4], Siyu Tang[3,4], Yin Li[2], and James M. Rehg[1]

[1]Georgia Institute of Technology, Atlanta, United States
[2]University of Wisconsin-Madison, Madison, United States
[3]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[4] ETH Zürich, Zürich, Switzerland

Abstract

In this report we describe the technical details of our submission to the EPIC-Kitchens 2020 action anticipation challenge. We adopt intentional hand movement as a feature representation, and propose a novel deep network that jointly models and predicts the egocentric hand motion, interaction hotspots and future action. Specifically, we consider the future hand motion as the motor attention, and model this attention using probabilistic variables in our deep model. The predicted motor attention is further used to select the discriminative spatial-temporal visual features for predicting actions. Our submission achieved a top-1 action anticipation accuracy of 15.42% *on seen kitchen set, and* 9.94% *on unseen kitchen set.*

1. Introduction

In this work, we address the challenging task of action anticipation in egocentric videos. To this end, we propose a novel deep model that predicts "motor attention"—the future trajectory of the hands, as an anticipatory representation of actions. Based on motor attention, our model further recognizes the type of future interactions. Importantly, we characterize motor attention as probabilistic variables modeled by stochastic units in the network. These units naturally deal with the uncertainty of future hand motion and produce attention maps that highlight discriminative spatial-temporal features for action anticipation. A detailed version of our report can be found in [6].

2. Approach

In this section, we detail our model design. Given an input video clip, our model seeks to anticipate the future action y by jointly modeling and predicting future hand trajectory (motor attention) $\mathcal{M}$ and interaction hotspots $\mathcal{A}$ at the last observable frame. Predicting the future is fundamentally ambiguous, since the observation of future interaction only represents one of the many possibilities characterized by an underlying distribution. Our key idea is thus to model motor attention and interaction hotspots as probabilistic variables to account for their ambiguity.

Formally, we consider motor attention $\mathcal{M}$ and interaction hotspots $\mathcal{A}$ as probabilistic variables, and model the conditional probability of future action label y given the input video x as a latent variable model, where

$$p(y|x) = \int_{\mathcal{M}} \int_{\mathcal{A}} p(y|\mathcal{A}, \mathcal{M}, x) p(\mathcal{A}|\mathcal{M}, x) p(\mathcal{M}|x)\, d\mathcal{A}\, d\mathcal{M},$$

$p(\mathcal{M}|x)$ first estimates motor attention from video input x. $\mathcal{M}$ is further used to estimate interaction hotsports A ($p(\mathcal{A}|\mathcal{M}, x)$). Given x, $\mathcal{M}$ and $\mathcal{A}$, action label y is determined by $p(y|\mathcal{A}, \mathcal{M}, x)$. Our model thus consists of three main components.

Motor Attention Module tackles $p(\mathcal{M}|x)$. Given the network features $\phi_2(x)$, our model uses a function F_M to predict motor attention $\mathcal{M}$. $\mathcal{M}$ is represented as a 3D tensor of size $T_m \times H_m \times W_m$. Moreover, $\mathcal{M}$ is normalized within each temporal slice, i.e., $\sum_{w,h} \mathcal{M}(t, w, h) = 1$.

Interaction Hotspots Module targets at $p(\mathcal{A}|\mathcal{M}, x)$. Our model uses a function F_A to estimate the interaction hotspots $\mathcal{A}$ based on the network feature $\phi_3(x)$ and sampled motor attention $\tilde{\mathcal{M}}$. $\mathcal{A}$ is represented as a 2D attention map of size $H_a \times W_a$. A further normalization constrained that $\sum_{w,h} \mathcal{A}(w, h) = 1$.

Anticipation Module makes use of the predicted motor attention and interaction hotspots for action anticipation. Specifically, sampled motor attention $\tilde{\mathcal{M}}$ and sampled interaction hotspots $\tilde{\mathcal{A}}$ are used to aggregate feature $\phi_5(x)$ via

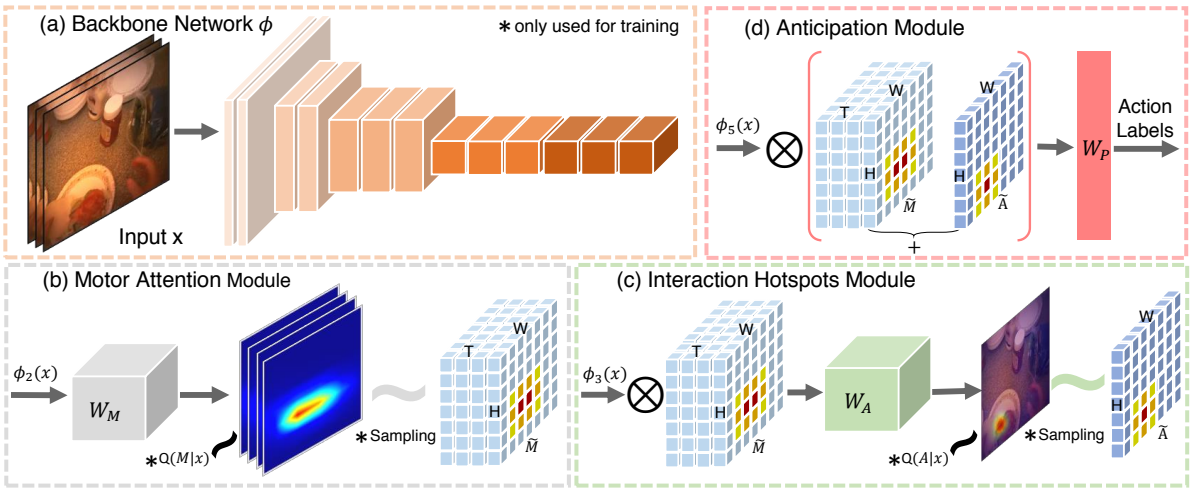


Figure 1. Overview of our model. A 3D convolutional network $\phi(x)$ is used as our backbone network, with features from its i^{th} convolution block as $\phi_i(x)$ (a). A motor attention module (b) makes use of stochastic units to generate sampled future hand trajectories $\tilde{\mathcal{M}}$ used to guide interaction hotspots estimation in module (c). Module (c) further generates sampled interaction hotspots $\tilde{\mathcal{A}}$ with a similar stochastic units as in module (b). Both $\tilde{\mathcal{M}}$ and $\tilde{\mathcal{A}}$ are used to guide action anticipation in anticipation module (d). During testing, our model takes only video clips as inputs, and predicts motor attention, interaction hotspots, and action labels. Note that $\otimes$ represents element-wise multiplication for weighted pooling.

weighted pooling. An action anticipation function F_P further maps the aggregated features to future action label y.

We present an overview of our model is shown in Fig. 1. Specifically, we make use of a 3D backbone network $\phi(x)$ for learning video representation. Following the convention in [8, 5], we assume the network has 5 convolutional blocks, and denote the features from the i^{th} convolution block of the network as $\phi_i(x)$. Based on $\phi(x)$, our motor attention module (b) predicts future hand trajectories as motor attention $\mathcal{M}$ and uses stochastic units to sample from $\mathcal{M}$. The sampled motor attention $\tilde{\mathcal{M}}$ is an indicator of important spatial-temporal features for interaction hotspots estimation. Our interaction hotspots module (c) further produces interaction hotspots distribution $\mathcal{A}$ and its sample $\tilde{\mathcal{A}}$. Finally, our anticipation module (d) further uses both $\tilde{\mathcal{M}}$ and $\tilde{\mathcal{A}}$ to aggregate network features and predicts the future interaction y.

3. Annotations and Implementation Details

Annotations. Our model requires supervisory signals of interaction hotspots and hand trajectories during training. We provide extra annotations for EPIC-Kitchens datasets. These annotations will be made publicly available. Specifically, we manually annotated interaction hotspots as 2D points on a subset of instances on EPIC-Kitchens. This is because many nouns labels in Epic-Kitchens have very few instances, hence we focus on interaction hotspots of action instances that include many-shot nouns [1] in the training set. As EPIC-Kitchens does not provide hand masks, we instead annotated the fingertip closest to an interaction hotspots on the last observable frame. A linear interpolation of 2D motion between the fingertip and the interaction hotspots was used to approximate the motor attention. Note that all annotations are obtained on *the last observable frame and are used only for training*. Therefore our model does not violate the anticipation challenge rule.

Implementation Details. We downsampled all frames to 512×288 with 30 fps for the EPIC-Kitchens dataset. For training, we applied several data augmentation techniques, including random flipping, rotation, cropping and color jittering to avoid overfitting. We adopt CSN-152 [9] network pre-trained on IG-65M [4]. For training, our model takes an input of 32 frames (subsampled by 2 from a 64 consecutive frames) with resolution of 224×224. For inference, our model samples 30 clips from a video (3 along width of frame and 10 in time). Each clip has 32 frames with a resolution of 256×256. We average the scores of all sampled clips for the video level prediction results. Our model was trained with cross entropy loss using SGD with momentum of 0.9. The batch size was 16 distributed on 4 GPUs (CSN-152). Synchronized batch normalization was enabled. The initial learning rate was 2.5e-4 (linear rescaled for smaller batch size) with cosine decay. Our model was trained for 18 epochs with cosine learning rate decay.

4. Results

In our recent work [6], we provide systematic ablation studies of how motor attention facilitate the learning of future representation. Table 1 compares our results to state-of-the-arts on EPIC-Kitchens. Our model outperforms strong baselines (TSN and 2SCNN) reported in [1] by a very large margin. Compared to previous best results from

	Method	Top1/Top5 Accuracy Verb	Noun	Action
s1	2SCNN [1]	29.76 / 76.03	15.15 / 38.65	4.32 / 15.21
	TSN [1]	31.81 / 76.56	16.22 / 42.15	6.00 / 18.21
	TSN+MCE [2]	27.92 / 73.59	16.09 / 39.32	10.76 / 25.28
	Trans R(2+1)D [7]	30.74 / 76.21	16.47 / 42.72	9.74 / 25.44
	RULSTM [3]	33.04 / **79.55**	22.78 / 50.95	14.39 / 33.73
	Ours	34.99 / 77.05	20.86 / 46.45	14.04 / 31.29
	Ours+Obj	**36.25** / 79.15	**23.83 / 51.98**	**15.42 / 34.29**
s2	2SCNN [1]	25.23 / 68.66	9.97 / 27.38	2.29 / 9.35
	TSN [1]	25.30 / 68.32	10.41 / 29.50	2.39 / 9.63
	TSN+MCE [2]	21.27 / 63.66	9.90 / 25.50	5.57 / 25.28
	Trans R(2+1)D [7]	28.37 / 69.96	12.43 / 32.20	7.24 / 19.29
	RULSTM [3]	27.01 / 69.55	15.19 / 34.38	8.16 / 21.20
	Ours	28.27 / 70.67	14.07 / 34.35	8.64 / 22.91
	Ours+Obj	**29.87 / 71.77**	**16.80 / 38.96**	**9.94 / 23.69**

Table 1. Action anticipation results on Epic-Kitchen test sets. Ours+Obj model outperforms previous state-of-the-art results by a relative improvement of **7%/22%** on seen/unseen set. At the time of submission, this model is ranked the second on both seen (s1) and unseen (s2) test set on EPIC-Kitchens leaderboard. See discussions of Ours+Obj in Sec. 4.2.

RULSTM [3], our model archives +2%/-1.9%/-0.3% for verb/noun/action on seen set, and +1.3%/-1.1%/+0.6% on unseen set of EPIC-Kitchens. Our results are better for verb, worse for noun and comparable or better for actions. Notably, RULSTM requires object boxes & optical flow for training and object features & optical flow for testing. In contrast, our method uses hand trajectories and interaction hotspots for training and needs *only RGB frames* for testing. To further improve the performance, we fuse the object stream from RULSTM with our model (Ours+Obj). Compared to RULSTM, Ours+Obj has a performance gain of $+3.2\%/+2.9\%$ for verb, $+1.1\%/+1.6\%$ for noun, and $+1.0\%/+1.8\%$ for action (seen/unseen). It is worthy pointing out that RULSTM benefits from an extra flow network, while ours+Obj model takes additional supervisory signals of hands and hotspots.

5. Conclusion

In this paper, we report the model details of our entry on EPIC-Kitchens action anticipation challenge. Specifically, we have presented the first deep model that jointly predicts motor attention, interaction hotspots, and future action labels in FPV. We demonstrated that motor attention plays an important role in forecasting human-object interactions. Another key insight is that characterizing motor attention and interaction hotspots as probabilistic variables can account for the stochastic pattern of human intentional movement and human-object interaction. We believe that our model connects the findings in cognitive neuroscience to an important task in computer vision, thereby providing a solid step towards the challenging problem of visual anticipation.

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] Antonino Furnari, Sebastiano Battiato, and Giovanni Maria Farinella. Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation. In *ECCV Workshops*, 2018.

[3] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *ICCV*, 2019.

[4] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *CVPR*, 2019.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[6] Miao Liu, Siyu Tang, Yin Li, and James Rehg. Forecasting human object interaction: Joint prediction of motor attention and egocentric activity. *arXiv preprint arXiv:1911.10967*, 2019.

[7] Antoine Miech, Ivan Laptev, Josef Sivic, Heng Wang, Lorenzo Torresani, and Du Tran. Leveraging the present to anticipate the future in videos. In *CVPR Workshops*, 2019.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[9] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *ICCV*, 2019.

Semi-Supervised Object Detection with Sparsely Annotated Dataset

Jihun Yoon
hutom
Republic of Korea
yjh2020@hutom.io

Seungbum Hong
hutom
Republic of Korea
qbration21@hutom.io

Sanha Jeong
hutom
Republic of Korea
jeongsanha@hutom.io

Min-Kook Choi
hutom
Republic of Korea
mkchoi@hutom.io

Abstract

In training object detector based on convolutional neural networks, selection of effective positive examples for training is an important factor. However, when training an anchor-based detectors with sparse annotations on an image, effort to find effective positive examples can hinder training performance. When using the anchor-based training for the ground truth bounding box to collect positive examples under given IoU, it is often possible to include objects from other classes in the current training class, or objects that are needed to be trained can only be sampled as negative examples. We used two approaches to solve this problem: 1) the use of an anchorless object detector and 2) a semi-supervised learning-based object detection using a single object tracker. The proposed technique performs single object tracking by using the sparsely annotated bounding box as an anchor in the temporal domain for successive frames. From the tracking results, dense annotations for training images were generated in an automated manner and used for training the object detector. We applied the proposed single object tracking-based semi-supervised learning to the Epic-Kitchens dataset. As a result, we were able to achieve ***runner-up*** *performance in the* ***Unseen*** *section while achieving the* ***first place*** *in the* ***Seen*** *section of the Epic-Kitchens 2020 object detection challenge under IoU* > 0.5 *evaluation.*

1. Introduction

Thanks to the rapid development of CNN (Convolutional Neural Networks), the performance of object recognition networks using CNN has also been improved dramatically [16]. As the performance of the object detection network has been improved, the dataset for evaluating it was also started from a dataset with low complexity such as PASCAL VOC [8] and developed to have a high complexity such as MS-COCO [15]. Among the object detection datasets, the relatively recently released Epic-Kitchens dataset has the following characteristics different from other object detection datasets [7].

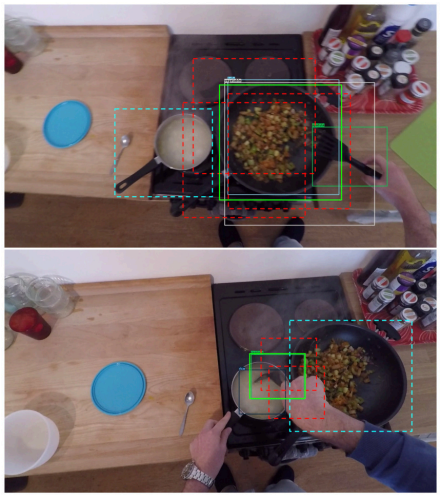


Figure 1. **An example of anchor-based detector training on a sparsely annotated dataset.** The solid green line represents the label information in each training image, and the red dotted line is an example of positive examples. Light blue dashed lines indicate objects that are included in the label in other training images (top), but are not labeled in the given image to train (bottom). As such, in the Epic-Kitchens object detection dataset, it is an object to learn when training an anchor-based detector, but training performance is impaired because label information is missing.

- Images for training detector are collected from the original video, and corresponding frame sequences are provided.
- In a training image, only some of the trainable objects are sparsely annotated.
- The difference in the amount of annotations between the few and many shot classes is large, depending on the distribution of the appearance of the objects in the training dataset.

As described above, the annotation of Epic-Kitchens for object detection is provided in a different way from the existing dataset, and has a characteristic that it is difficult to apply the method of training the existing object detection model as it is. Typically, in the case of detectors that train positive examples based on anchors [21, 17] or detectors

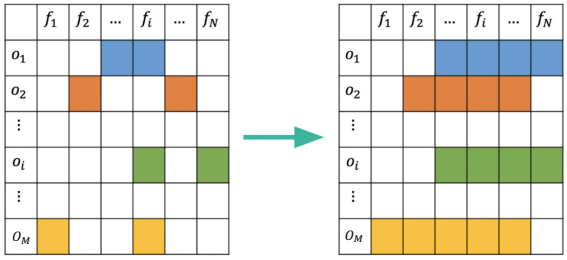


Figure 2. **The final goal of the proposed supervised learning.** f denotes an action clip composed of N frames, and o denotes a total of M objects present in the action clip. We performed semi-supervised learning through bidirectional tracking to obtain dense labels for all learnable objects present in the action clip.

that train the objectness of a candidate object with the structure of an RPN [23, 6], batch sampling is performed considering the Intersection on Union (IoU) with the ground truth bounding box for effective training. However, if an anchor-based hard example mining is performed on a sparsly annotated training image, the efficiency of training is hindered by the distribution of objects near the ground truth bounding box. Figure 1 shows the negative effects of trying to select a good positive example when training anchor-based object detectors with sparse annotation images.

We have trained the object detector through two approaches to solve this problem. First, a detector using different parameterization for estimating the bounding box was used instead of the anchor-based detection model. We were able to minimize the effect of sparsely annotated training images affected by anchor-based sampling by utilizing the Fully Convolutional One-Stage Object Detection (FCOS) network [24]. The second is to utilize the features of the Epic-Kitchens dataset, and the object tracking technique is used to semi-supervise all detectable objects in each training image. To this end, the bounding box label existing in a specific frame is set as an initial bounding box on the time domain and used as an input of a single object tracker. Subsequently, the predictive output exceeding the threshold from the tracker was assumed to be a pesudo annotation, and labels for all learnable objects were provided in the image for training. Figure 2 schematically shows the proposed goal of semi-supervised learning.

With the proposed approach, we were able to train an object detection network effectively with the Epic-Kitchens object detection dataset. Subsequently, joint NMS-based ensemble [11] was performed for FCOS models with trained inhomogeneous backbones. As a result, we were able to achieve ***first place*** in the *Seen* evaluation set and ***runnerup*** in the *Unseen* evaluation set under the IoU > 0.5 of the Epic-Ktichens 2020 object detection challenge.

2. Related Work

Object detection. CNN-based object detection models are largely divided into one-stage or two-stage models. In the one-stage model, the process of predicting the class and position of an object is performed in one structure, and examples include YOLO [21, 22], SSD [17, 9], and RetinaNet [14]. Additional structures such as Feature Pyramid Networks (FPN) [21] are often used to efficiently process the output features obtained from the backbone in the head structure. In general, it is known that the regression accuracy is lower than the two-stage model because classification and regression are performed in one structure. In the case of the two-stage model, the prior knowledge of the location of the object is estimated from the RPN (Region Proposal Network), which is a subnetwork in the detector [23]. RPN determines objectness by class-agnostic subnet, and performs class-aware detection through the subsequent head structure. Faster R-CNN [23], R-FCN [6], Cascade R-CNN [2], Cascade RPN [25], etc. are representative of various head structures, and are known to have relatively high regression accuracy. Models such as RefineDet [29] that combine the philosophy of one-stage and two-stage models have also been proposed, and detectors that utilize other parameterizations for bounding box regression rather than structural advantages, such as FCOS [24], have also been proposed.

Semi-supervised learning for object detection. Object detection using semi-supervised learning is used in situations where it is difficult to manually acquire a sufficient number of annotations to learn, or when pseudo labels are to be obtained from a relatively large number of unlabeled data. [18] proposed an iterative framework for evaluating and retraining pseudo labels using pre-trained object detectors and robust trackers to obtain good pseudo labels in successive frames. In [19], it was possible to achieve improved detection performance in Open Image Dataset V4 by utilizing part-aware sampling and RoI proposals to obtain good pseudo labels for sparsely annotated large-scale datasets. In [5], in order to efficiently use unlabeled data from the MS-COCO dataset, co-occurrent matrix analysis was used to generate good pseudo labels by using prior information of the labeled dataset. The proposed single object tracker-based semi-supervised learning is similar to [18] in that it uses a tracker, but has a difference in obtaining dense annotation information for a specific image by using the existing lean annotation information. At the same time, since the object detector is not used for the initial input for tracking, the training is not applied as an iterative training scenario.

Single object visual tracking. We used a single object tracking network to generate pseudo labels for sparsely an-

notated datasets. In single object tracking [1, 13, 27, 3, 12], the Siamese network-based visual tracker shows balanced accuracy and speed across various datasets. The Siamese network-based tracker basically trains with the similarity of the CNN feature for the target image and the input image for tracking. We used SiamMask [27] as a single object tracker, which uses box and mask information together with similarity of features to the tracking target.

3. Fully Convolutional One-Stage Object Detection (FCOS)

We used the FCOS model [24] to exclude the computational process for selecting a good positive example with an anchor from detector training. FCOS defined the parameters for regression of the bounding box differently, and presented an anchor-free detector. The loss function of FCOS is defined as follows.

$$L(a_p, m_p) = \frac{1}{N} \sum_{\{u,v\} \in p} L_{cls}(a_p, c_p) + \frac{1}{N} \sum_{\{u,v\} \in p} [c_p > 0] L_{reg}(m_p, \hat{m}_p), \quad (1)$$

where p denotes a position (u, v) on the feature map, and a_p denotes a prediction vector for class estimation. c_p denotes the class label for the input example, and m_p denotes the specific spatial location of the feature map and the distance from the ground truth bounding box $m_p = (l, t, r, b)$. Given the label $B = (x_0, y_0, x_1, y_1, c)$ for the bounding box, FCOS parameterizes to find the bounding box by $l = u - x_0$, $t = v - y_0$, $r = u - x_1$, $b = v - y_1$. We used pretrained ResNet [10], ResNeXt [28], HRNet [26] as the backbone network for FCOS to perform inhomogeneous ensemble.

4. Semi-Supervised Learning with Single Object Tracker

The Epic-Kitchens dataset simultaneously provides a bounding box label for a particular object and a sequence frame for the *action clip* in which the object appears. The bounding box for the object is not densely given every frame, but rather sparsely in the action sequence. We used a single object tracker to achieve the goal in Figure 2 with an automated procedure. Among various single object trackers, SiamMask [27] was used, which shows a balanced performance for tracking accuracy and speed. We performed a bidirectional tracking using the SiamMask model trained from the DAVIS dataset, using each bounding box as the initial value for a single object. The details of forward tracking with SiamMask for one action clip input are described in Algorithm 1. Algorithm 1 is used in the same way for backward tracking to complete bidirectional tracking. Figure 3 shows an example of the start and end of tracking according to Algorithm 1 on a single object, and Figure 4 shows training images with pseudo labels generated after tracking an object.

Algorithm 1: Forward tracking

Input: Action clip (A), pretrained tracking model (T), a set of bbox for initial input (BB), threshold of tracking score (ρ_1), threshold of IoU between two pair of tracked bbox (ρ_2)
Output: BB in Q from T
Initialize an empty queue Q
while *bbox $b_{c,i}$ with class c at i-th frame available from BB* **do**
  Get a list of frames FF in forward from i-th frame in A;
  Initialize T with b_i from A;
  Initialize a variable pre_s with a size of $b_{c,i}$ to store a size of object from T at previous frame;
  while *each frame in FF* **do**
    Get a bbox $b_{c,k}$ from T at k-th frame;
    $crnt_s :=$ a size of $b_{c,k}$;
    if $IoU(prev_s, crnt_s) \geq \rho_1$ **then**
      $prev_s := crnt_s$;
      Add $b_{c,k}$ to Q;
    else
      break;
    end
  end
end

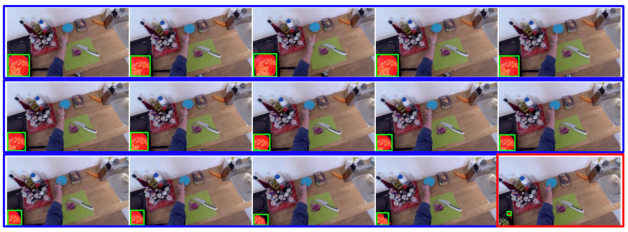


Figure 3. **Example of the result of Algorithm 1.** Frames marked with blue boxes are frames that have been tracked with the same object since tracking started, and a frame marked with a red box is a frame whose tracking is terminated due to the termination condition of algorithm 1.

Figure 4. **Changes in training images after tracking.** An example of the final annotations to be used for semi-supervised learning is shown on the training images indicated by the red dotted line.

Table 1. **Training detail of each detector.** 'lr' represents the learning rate. Learning schedule is indicated as (scheduler, drop rate) [drop epoch1:drop epoch2:max epoch].

Detector	Backbone	Optimizer (lr)	Learning schedule	Warmup(iter, ratio)
Faster R-CNN	ResNet-101	SGD (0.02)	(step, 0.1)[8:11:24]	linear(500, 1/3)
Cascade R-CNN	HRNet-V2P	SGD (0.02)	(step, 0.1)[16:19:30]	linear(500, 1/3)
FCOS	HRNet-V2P	SGD (0.01)	(step, 0.1)[8:11:20]	constant(500, 1/3)
FCOS	ResNet-50	SGD (0.01)	(step, 0.1)[8:11:16]	constant(500, 1/3)
FCOS	ResNet-101	SGD (0.01)	(step, 0.1)[16:20:22]	constant(500, 1/3)
FCOS	ResNeXt-101	SGD (0.01)	(step, 0.1)[16:22:39]	constant(500, 1/3)

Table 2. **Performance comparison of anchor-based and anchorless detectors.** If the model name has a '+', it is the result of evaluation using tracker-based semi-supervised learning. The highest performance in a single model and the highest performance in an entire model are shown in bold.

Detector	backbone	Seen			Unseen		
-	-	> 0.05	> 0.5	> 0.75	> 0.05	> 0.5	>0.75
Faster R-CNN	ResNet-101	37.54	28.64	6.92	32.83	23.16	5.55
Cascade R-CNN	HRNet-V2P	30.44	24.17	8.73	23.87	18.05	6.81
FCOS	HRNet-V2P	48.44	34.87	**11.02**	**43.88**	**30.68**	**9.27**
FCOS	ResNet-50	46.96	34.51	10.09	42.46	29.49	7.48
FCOS	ResNet-101	49.77	35.8	10.15	43.39	28.98	7.86
FCOS	ResNeXt-101	48.17	33.95	9.86	41.79	27.27	7.19
FCOS+	ResNet-101	**50.27**	**35.89**	10.57	43.14	29.82	7.76
FCOS Ensemble+	-	**58.27**	**44.48**	**15.36**	**55.72**	**41.12**	**12.5**

5. Epic-Kitchens Object Detection Results

Training details. We used Faster R-CNN [23] and Cascade R-CNN [2] as anchor-based detectors and FCOS [24] as an anchorless detector to compare performance in the Epic-Kitchens object detection dataset [7]. As the backbone CNN for training the detector, ResNet-50, ResNet-101, ResNeXt-101, and HRNet-V2p-W32 pretrained with ImageNet were used, and training details for each combination of backbone and head structure are shown in Table 1. All experiments were conducted using the MMDetection library [4].

Anchor-based vs. anchorless detector Table 2 shows the training performance in a single model of an anchor-based detector and an anchorless detector. According to Table 2, it can be seen that in the basic performance of training, the performance of the detector without an anchor is excellent and shows stable learning results. Figure 5 shows the loss change during training of an anchor-based detector and an anchorless detector. The detector without anchor shows a relatively stable loss curve. At the same time, Table 2 shows the performance change of the FCOS model according to different backbones. For a single model, it was confirmed that the FCOS model utilizing the ResNet-101 backbone achieved the best generalization performance in the Seen set, and the HRNet backbone model performed the best in the Unseen set.

Semi-supervised learning. We used the pretrained SiamMask model from the DAVIS dataset [20] to generate dense labels to train the FCOS models with dense annotations. Table 2 shows that the generalization performance of the FCOS models under $IoU > 0.5$ is consistently improved when using semi-supervised learning based on single object tracking.

Inhomogeneous backbone ensemble. We performed model ensemble for each trained model in Table 2 to achieve the best detection performance from trained detectors. We performed the ensemble using the joint NMS technique [11], where we can achieve an ensemble by applying NMS to a bounding box with up to 300 high prediction scores obtained from every detectors. Table 2 shows the performance change of the Seen and Unseen sets according to the ensemble combination, and Figure 6 shows the performance published on the Epic-Kitchen object detection challenge page. We were finally able to achieve the ***first rank*** in *Seen* set and ***runnerup*** performance in *Unseen* set through an inhomogeneous backbone ensemble under IoU > 0.5 evaluation.

Visualizations. We visualized the inference results of each model to confirm the effect of the inhomogeneous backbone ensemble. Figure 7 shows the visualization of the inference bounding box of the FCOS model with different backbones. As shown in Figure 7, the detectors have the same structure, but only different backbones can be used to obtain very different types of inference results. Through this, it was confirmed that the ensemble model can achieve a very large performance improvement compared to a single model.

6. Conclusion

We performed single object tracker-based semi-supervised object detection to effectively train datasets with sparse annotations on sequence images. The Epic-Kitchens object detection dataset was used to verify the utility of the proposed technique, and the proposed semi-supervised learning showed good performance in the ensemble as well as in the single model. However, it needs to be analyzed more closely with semi-supervised learning about the advantages and disadvantages of the anchor-based model, and there is a limitation that a simple rule-based engine is used to obtain a pseudo label. For future improvement, it is necessary to perform quantitative analysis on the effect of anchor and RPN on sparse annotation data training, and at the same time, it is possible to consider how to improve the tracking rules or utilize the results obtained in the tracking process during training.

References

[1] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In *In Proc. of ECCV*, 2016.

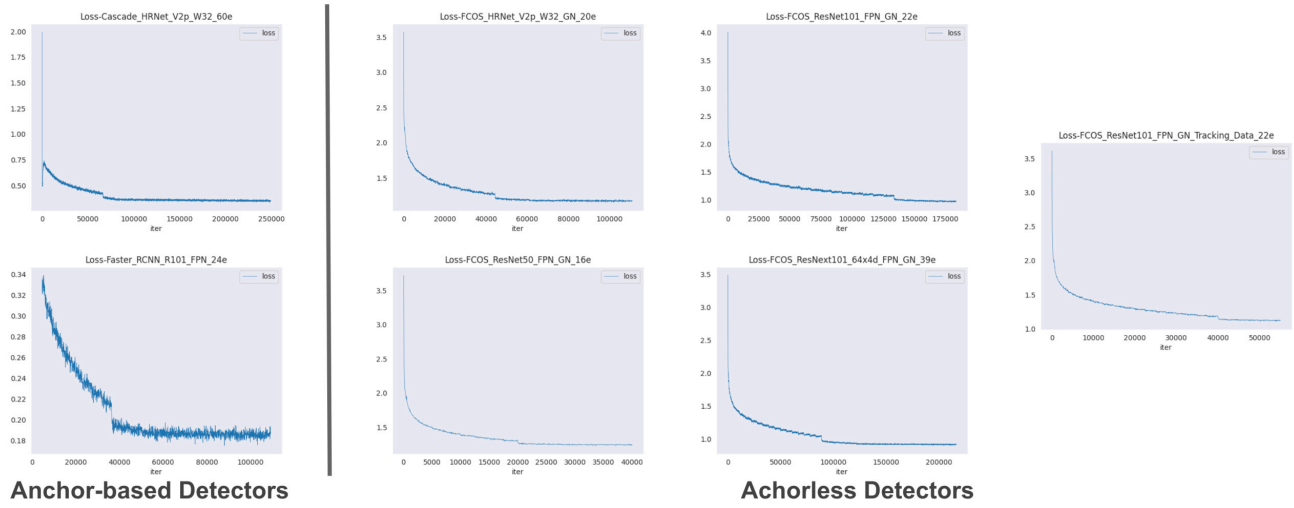


Figure 5. **Training losses for anchor-based and anchorless detectors.** It shows that the loss curve of the anchorless detectors is relatively stable.

Seen Kitchens (S1)

#	User	Entries	Date of Last Entry	Team Name	Few Shot Classes (%)			Many Shot Classes (%)			All Classes (%)		
					IoU > 0.05 ▲	IoU > 0.5 ▲	IoU > 0.75 ▲	IoU > 0.05 ▲	IoU > 0.5 ▲	IoU > 0.75 ▲	IoU > 0.05 ▲	IoU > 0.5 ▲	IoU > 0.75 ▲
1	killerchef	51	05/30/20	hutom	47.44 (3)	35.75 (1)	14.32 (2)	60.77 (2)	46.50 (1)	15.60 (2)	58.27 (2)	44.48 (1)	15.36 (1)
2	gongtao	35	05/19/20		49.95 (2)	32.63 (2)	6.64 (6)	60.03 (3)	44.39 (3)	9.71 (8)	58.13 (3)	42.18 (2)	9.13 (6)
3	kide	27	05/29/20	DHARI	54.98 (1)	32.40 (3)	14.55 (1)	68.74 (1)	43.88 (4)	15.38 (3)	66.15 (1)	41.72 (3)	15.23 (2)
4	gb7	69	04/01/20	FB AI	26.55 (8)	19.01 (8)	8.22 (4)	58.44 (4)	46.22 (2)	15.61 (1)	52.44 (4)	41.10 (4)	14.22 (3)
5	cvg_uni_bonn	23	05/12/20	CVG Lab Uni Bonn	39.36 (4)	26.66 (4)	7.89 (5)	53.50 (5)	41.28 (5)	12.46 (4)	50.84 (5)	38.53 (5)	11.60 (4)

Unseen Kitchens (S2)

#	User	Entries	Date of Last Entry	Team Name	Few Shot Classes (%)			Many Shot Classes (%)			All Classes (%)		
					IoU > 0.05 ▲	IoU > 0.5 ▲	IoU > 0.75 ▲	IoU > 0.05 ▲	IoU > 0.5 ▲	IoU > 0.75 ▲	IoU > 0.05 ▲	IoU > 0.5 ▲	IoU > 0.75 ▲
1	gb7	69	04/01/20	FB AI	13.70 (8)	10.41 (8)	2.88 (7)	59.21 (2)	45.42 (1)	16.24 (1)	54.57 (4)	41.85 (1)	14.88 (1)
2	killerchef	51	05/30/20	hutom	29.81 (3)	20.87 (4)	8.09 (1)	58.66 (3)	43.42 (2)	13.00 (3)	55.72 (2)	41.12 (2)	12.50 (3)
3	kide	27	05/29/20	DHARI	35.75 (1)	22.31 (2)	7.33 (4)	67.92 (1)	41.92 (3)	14.29 (2)	64.64 (1)	39.93 (3)	13.58 (2)
4	gongtao	35	05/19/20		35.72 (2)	25.60 (1)	7.78 (3)	56.93 (4)	41.19 (4)	8.75 (7)	54.77 (3)	39.60 (4)	8.65 (6)
5	cvg_uni_bonn	23	05/12/20	CVG Lab Uni Bonn	25.34 (4)	21.54 (3)	7.81 (2)	52.18 (5)	38.24 (5)	11.41 (4)	49.45 (5)	36.54 (5)	11.04 (4)

Figure 6. **Epic-Kitchens 2020 object detection challenge evaluation page.** The entry marked with a red box is the final performance evaluated by the our proposed approach. Each entry is ranked under IoU > 0.5 evaluation.

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *In Proc. of CVPR*, 2018.

[3] Bao Xin Chen and John K. Tsotsos. Fast visual object tracking with rotated bounding boxes. In *In Proc. of ICCVW*, 2019.

[4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv:1906.07155*, 2019.

[5] Min-Kook Choi, Jaehyeong Park, Jihun Jung, Heechul Jung, Jin-Hee Lee, Woong Jae Won, Woo Young Jung, Jincheol Kim, and Soon Kwon. Co-occurrence matrix analysis-based semi-supervised training for object detection. In *In Proc. of ICIP*, 2018.

[6] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *In Proc. of NIPS*, 2016.

[7] 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 *In Proc. of ECCV*, 2018.

[8] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111:98–136, 2015.

[9] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C. Berg. Dssd : Deconvolutional single shot detector. *arXiv:1701.06659*, 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *In Proc. of CVPR*, 2016.

[11] Heechul Jung, Min-Kook Choi, Jihun Jung, Jin-Hee Lee, Soon Kwon, and Woo Young Jung. Resnet-based vehicle classification and localization in traffic surveillance systems. In *In Proc. of CVPR*, 2017.

[12] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *In Proc. of CVPR*, 2019.

[13] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *In Proc. of CVPR*, 2018.

[14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *In Proc. of ICCV*, 2017.

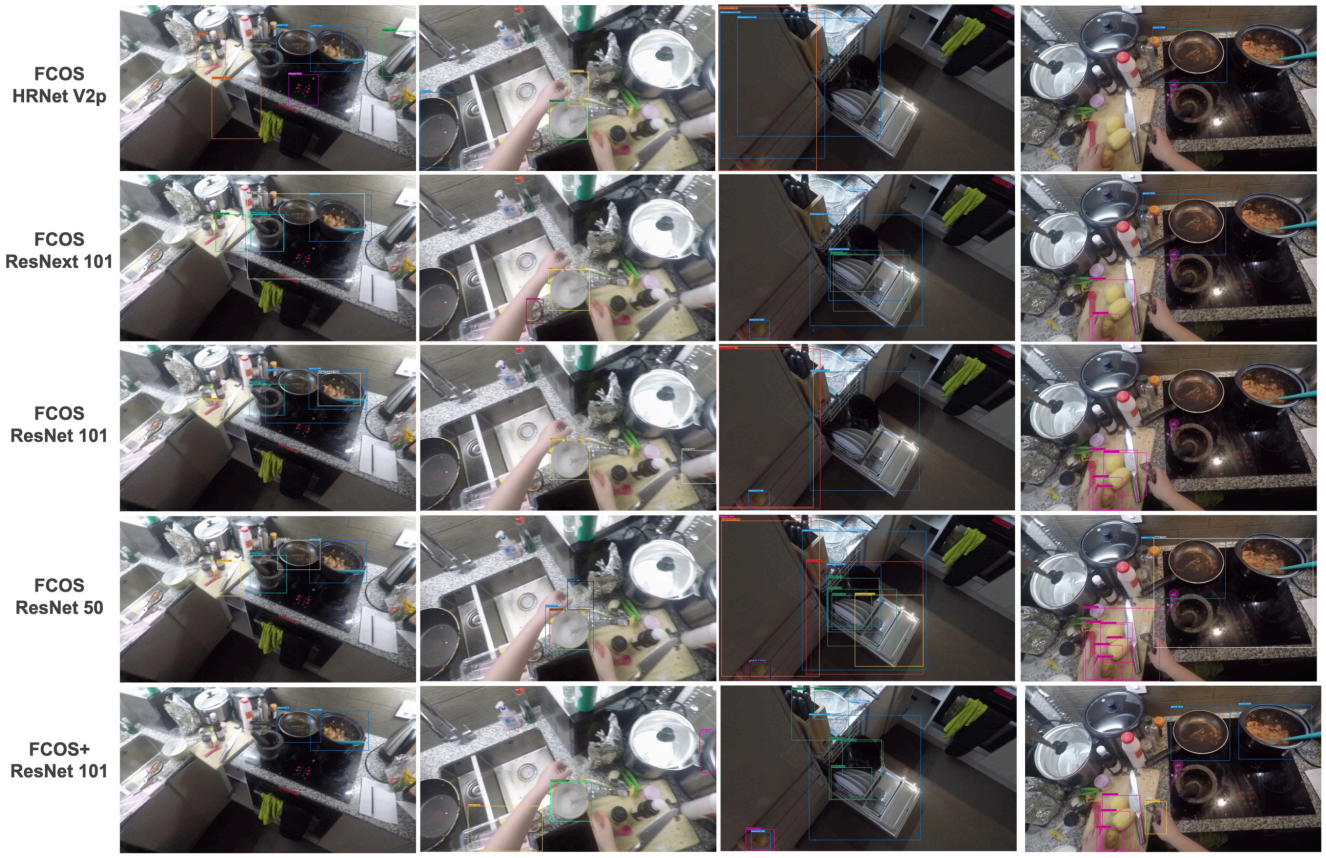


Figure 7. **Visualization for each network prediction.** It shows that the prediction scores can be variously distributed for each network when they have the same structure but have a heterogeneous backbone. The predicted score threshold for visualization was set at 0.5.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro PeronaDeva, Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *In Proc. of ECCV*, 2014.

[16] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *arXiv:1809.02165*, 2018.

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *In Proc. of ECCV*, 2016.

[18] Ishan Misra, Abhinav Shrivastava, and Martial Hebert. Watch and learn: Semi-supervised learning for object detectors from video. In *In Proc. of CVPR*, 2015.

[19] Yusuke Niitani, Takuya Akiba, Tommi Kerola, Toru Ogawa, Shotaro Sano, and Shuji Suzuki. Sampling techniques for large-scale object detection from sparsely annotated objects. In *In Proc. of CVPR*, 2019.

[20] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbelaez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[21] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *In Proc. of CVPR*, 2017.

[22] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv:1804.02767*, 2018.

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *In Proc. of NIPS*, 2015.

[24] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *In Proc. of ICCV*, 2019.

[25] Thang Vu, Hyunjun Jang, Trung X. Pham, and Chang Yoo. Cascade rpn: Delving into high-quality region proposal network with adaptive convolution. In *In Proc. of NIPS*, 2019.

[26] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *arXiv:1908.07919*, 2019.

[27] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H.S. Torr. Fast online object tracking and segmentation: A unifying approach. In *In Proc. of CVPR*, 2019.

[28] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *In Proc. of CVPR*, 2017.

[29] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z. Li. Single-shot refinement neural network for object detection. In *In Proc. of CVPR*, 2018.

FB AI Submission to the EPIC-Kitchens 2020 Object Detection Challenge

Gedas Bertasius, Lorenzo Torresani
Facebook AI

Abstract

We leverage the framework of the recently proposed video instance segmentation system MaskProp [1] for object detection in egocentric videos. While the original MaskProp system is trained to track and segment objects, we adapt it to use long-range temporal cues for improved object detection. Given an input video frame, our system performs object detection in that frame by incorporating relevant features from other frames in a video. Using such temporal cues allows our model to handle challenging cases of object occlusions and motion blur, which are particularly common in egocentric videos. Our method does not rely on the expensive optical flow methods, it can be easily trained end-to-end, and it achieves state-of-the-art results on the unseen test set of EPIC-Kitchens object detection challenge.

1. Introduction

In recent years, the field of object detection has witnessed rapid progress for the domain of images [7, 12, 6, 8]. However, directly applying these image-level models to egocentric videos is challenging as these models are not designed to handle motion blur, or object occlusions, which are quite common in an egocentric setting.

Consider an example in Figure 1, for which our main goal is to detect an active object in frame t (i.e. an instance of tongs). If we only had access to frame t, this would be a very challenging task because in frame t, the tongs are heavily occluded. However, if we could also leverage frames $t-5$ and $t+5$, this task would be much easier because the tongs are much more clearly visible in these other frames. Thus, the ability to leverage temporal information can be highly beneficial for accurately detecting objects in egocentric videos.

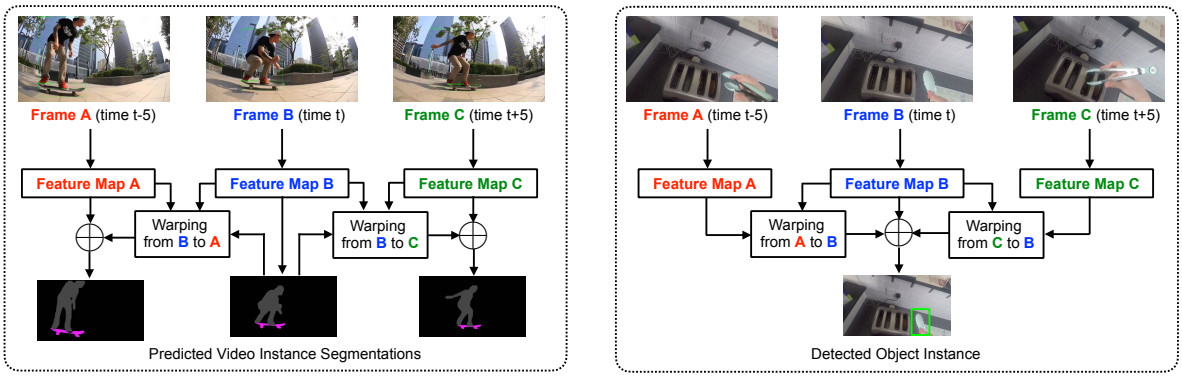
For this challenge, we leverage the framework of the recent video instance segmentation system MaskProp [1], which uses deformable convolutions [4] across space and time to learn temporal feature correspondences between objects in different frames. In the original paper [1], these temporal correspondences are used for tracking, whereas we use them as means to incorporate long-range temporal cues for improved object detection. Specifically, our model learns to warp useful feature points from other video frames such that object detection accuracy in a given video frame is maximized. This naturally renders our approach robust to occlusion or motion blur in individual frames of egocentric videos. We also demonstrate that our model obtains state-of-the-art results on the unseen test set of EpicKitchens object detection challenge.

Figure 1: An illustration of the common challenges associated with object detection in egocentric video. The bounding box denotes an object instance that we want to detect (an instance of tongs). Due to occlusions, it is difficult to detect an object in frame t by only leveraging information from frame t. However, leveraging additional object cues from frames $t-5$, and $t+5$ makes the object detection task considerably easier as the same object is more clearly seen in those frames.

2. Our Approach

Backbone Architecture. As our backbone, we use a ResNeXt-101 [13] with a Feature Pyramid Network [9].

Temporal Feature Warping. As in the original MaskProp system [1], we first consider a pair of frames, which we denote as I_t and I_{t+5}. Our goal is to warp relevant features from frame I_{t+5} such that object detection accuracy in frame I_t is maximized. To do this, we first feed both frames through our backbone CNN, which outputs feature tensors f_t and f_{t+5}. We then compute the feature difference $\psi_{t,t+5} = f_t - f_{t+5}$, and provide it as input to a simple residual block, which outputs offsets $o(p_n)$ at all pixel locations p_n. The predicted offsets are used to spatially rewarp the feature tensor f_{t+5} into frame t. The warping mechanism is implemented using a deformable convolution [4],



a) MaskProp **b)** Temporal Feature Aggregation

Figure 2: We adapt the framework of the recent video instance segmentation system MaskProp [1] for object detection in egocentric videos. The original MaskProp is trained to segment and track object instances in the video (see subfigure on the left). Instead, we we want to incorporate long-range temporal cues for improving object detection. Given a particular video frame (i.e., the middle frame of a video in this case), our model learns to warp useful features from other video frames such that object detection accuracy in a given video frame is maximized. All temporally warped features are then aggregated into a single feature tensor, which is subsequently used to localize and classify object instances in a given video frame (see subfigure on the right). Our model's ability to incorporate long-range temporal cues allows it to produce accurate detections even if objects in individual frames appear occluded or blurry.

which takes 1) the offsets $o(p_n)$, and 2) the feature tensor f_{t+5} as its inputs, and then outputs a newly sampled feature tensor $g_{t,t+5}$.

Temporal Feature Aggregation. As is done in [1, 2], we apply the temporal feature warping scheme for all pairs of frames I_t, and $I_{t+\delta}$ where $\delta = -K, -K+1, ..., K-1, K$. Doing so, produces a set of feature tensors $g_{t,t+\delta}$ for $\delta = -K, -K+1, ..., K-1, K$. We then want to aggregate all these feature tensors into a single tensor g_t^{agg}, which effectively captures information from its K preceding and K subsequent frames. The aggregation is done using a weighted summation:

$$g_t^{agg} = \sum_{\delta=-K}^{K} w_{t,t+\delta} \; g_{t,t+\delta} \qquad (1)$$

Here, the temporal feature aggregation weights $w_{t,t+\delta}$ are defined as $w_{t,t+\delta} = a \exp(-|\delta|/b)$. The rationale behind this definition is simple. In this case, our goal is to detect object instances in frame t. Thus, it is reasonable to assume that some of the most informative features will come from the feature tensors that are temporally close to frame t. Therefore, the aggregation weight associated with such frames should be large. On the other hand, the warped feature tensor $g_{t,t+\delta}$ that is far away from frame t may not even contain the object of interest. Thus, its corresponding weight $w_{t,t+\delta}$ should be small.

Our definition of w captures this intuition, which is why we adopt it throughout our experiments. We also note that all all weights w are fed into the softmax layer, to ensure that they sum up to 1 at each pixel location.

Detection Network. Lastly, the aggregated feature tensor g_t^{agg} is used as input to the Cascaded R-CNN detection network [3], with 3 stages. This last step yields the final bounding box predictions and their object class probabilities.

2.1. Implementation Details

Training. We train our method for object detection on the EPIC-Kitchens [5] dataset for 1 epoch with a learning rate of 0.004. Our model is initialized with a ResNeXt-101 [13], which is first pretrained on Instagram [11] for image classification, and is then also trained on COCO [10] for object detection. The hyperparameters of RPN, FPN, and Cascade R-CNN are the same as in [3].

Inference. During inference, we leverage temporal cues from 23 frames. This means that for a given video frame, we incorporate relevant features from 11 previous and subsequent video frames. The hyper-parameters for computing temporal feature aggregation weights are set to $a = 5, b = 10$. For the detection stage, we run the bounding box prediction branch on 1000 proposals, apply non-maximum suppression, and use 300 top ranked boxes with a score higher than 0.0001 as our final detections.

3. Experimental Results

Leaderboard Results. In Table 1, we present leaderboard results on the unseen test set of EPIC-Kitchens object detection challenge [5]. The results are evaluated according to the mAP metric from PASCAL VOC using an Intersection over Union (IoU) threshold of 0.5. Our challenge entry, named "gb7" outperforms all other challenge entries demonstrating its effectiveness on this task.

Ablation Studies. In Table 2, we also investigate the rea-

Method Name	mAP
VCL-ITI	33.55
cvg_uni_bonn	36.54
gongtao	39.60
kidef	39.93
killerchef	41.12
gb7	**41.85**

Table 1: Here, we present leaderboard results on the unseen test set of EPIC-Kitchens object detection challenge [5]. The performance is evaluated according to the mAP metric from PASCAL VOC using an IoU threshold of 0.5. Our challenge entry, named "gb7", outperforms all other challenge entries.

sons behind our method's success. Specifically, we study 1) the importance of large-scale Instagram pretraining [11], and 2) the effectiveness of long-range temporal cues.

We note that using a backbone network that was pretrained on the Instagram dataset boosts our method's accuracy by 2.86 mAP. Furthermore, we observe that temporally aggregating information from 23 video frames for detection improves our system's performance by 4.55 mAP over the single-frame baseline, which suggests that long-range temporal cues are indeed beneficial for this task.

4. Discussion

Lastly, we would like to discuss several potential issues related to the current evaluation protocol of the EPIC-Kitchens object detection challenge. In the description of the benchmark [5], the authors mention that: "for each class, we only report results on images where class has been annotated". To understand why this is problematic consider a following example. Suppose that we had 1 image containing an annotated instance of a knife and 99 images containing no annotated knife instances. Now consider a system A, which *correctly* detects a knife instance in the first image, but *incorrectly* predicts that there are also knife instances in the 99 other images. In other words it's predicting knife instances in all 100 images. Now let us consider a system B that *correctly* detects a knife instance in the first image, and also *correctly* predicts that there are no knife instances in the 99 other images. Based on our understanding of the current evaluation protocol, both systems A and B would yield the same evaluation score for a category "knife", even though system B is objectively much better. Thus, we believe that the current evaluation protocol favors systems with high recall over the systems that are well balanced in terms of precision and recall metrics. In other words, a system predicting many false positives but few false negatives will generally perform very well according to the current evaluation protocol.

To investigate this, we constructed our own object detection benchmark from the EPIC-Kitchens training set, and evaluated a series of strong models on it using a standard COCO mAP detection metric. We observed that models that performed very well on our own benchmark would not necessarily perform well on the EPIC-Kitchens object detection challenge. Similarly, our best model on the EPIC-Kitchens object detection challenge performed quite poorly on our own benchmark. Thus, we believe that such a trend might indicate potential issues in the current EPIC-Kitchens object detection evaluation protocol. We are also confident that addressing these evaluation issues would allow us to advance state-of-the-art in egocentric object detection even further.

IG Pretraining	# of Frames	mAP
✓	1	37.30
✗	23	39.99
✓	23	**41.85**

Table 2: Our ablation studies investigating 1) the importance of large-scale Instagram pretraining [11], and 2) the effectiveness of long-range temporal cues for the detection task. Our results indicate that both of these components are critical to our system's good performance.

References

[1] Gedas Bertasius and Lorenzo Torresani. Classifying, segmenting, and tracking object instances in video with mask propagation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2

[2] Gedas Bertasius, Lorenzo Torresani, and Jianbo Shi. Object detection in video with spatiotemporal sampling networks. In *ECCV (12)*, 2018. 2

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018. 2

[4] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, volume 00, pages 764–773, Oct. 2017. 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

[6] Ross Girshick. Fast R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 1

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 1

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017. 1

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, Zürich, September 2014. 2

[11] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018. 2, 3

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, 2015. 1

[13] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 1, 2

DHARI Report to EPIC-Kitchens 2020 Object Detection Challenge

Kaide Li, Bingyan Liao, Laifeng Hu, Yaonong Wang
ZheJiang Dahua Technology CO.,LTD, Hangzhou, China
{li_kaide, liao_bingyan, hu_laifeng, wang_yaonong}@dahuatech.com

Abstract

In this report, we describe the technical details of our-submission to the EPIC-Kitchens Object Detection Challenge. Duck filling and mix-up techniques are firstly introduced to augment the data and significantly improve the robustness of the proposed method. Then we propose GRE-FPN and Hard IoU-imbalance Sampler methods to extract more representative global object features. To bridge the gap of category imbalance, Class Balance Sampling is utilized and greatly improves the test results. Besides, some training and testing strategies are also exploited, such as Stochastic Weight Averaging and multi-scale testing. Experimental results demonstrate that our approach can significantly improve the mean Average Precision (mAP) of object detection on both the seen and unseen test sets of EPIC-Kitchens.

1. Introduction

EPIC-Kitchens dataset was introduced as a large scale first-person action recognition dataset. In this Object detection task, the annotations only capture the 'active' objects pre-, during- and post- interaction [3] [2]. It is challenging for object detection due to the influence of the sparse annotations and long-tail class distribution in this dataset. To address these challenges, we focus on the sampling methods in detection process and present the GRE-FPN and Hard IoU-imbalance Sampler methods to improve the robustness of location. Additionally, duck filling methods are applied to balance the influence of the long-tail class distribution and improve the diversity of few-shot classes. Experimental results demonstrate our approach can significantly improve the object detection performance and achieve a competitive result on the test set. The implementations details of the above are described in section 2 and section 3.

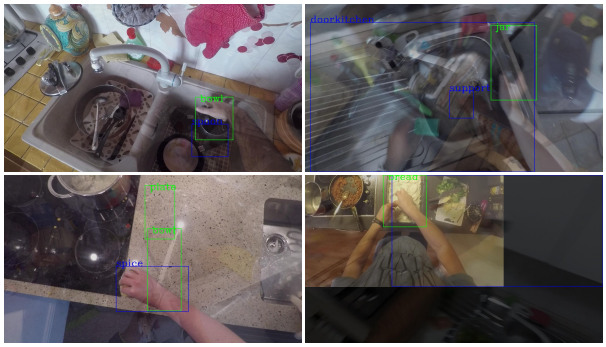


Figure 1. The mix-up images for few-shot classes.

Figure 2. The duck filling images for few-shot classes.

2. Proposed Approach

2.1. Data Preprocess

The EPIC-Kitchens training dataset for object detection contains 290 valid annotation categories and 326,064 bounding boxes. But the training dataset is extremely imbalanced, and it has the long-tail class distribution. Especially, there are only 8,136 bounding boxes for few-shot classes (118 classes) with less than 200 bounding boxes in total. To address the imbalance between few-shot classes and many-shot classes and improve the diversity of few-shot classes, we adopt some data augmentation preprocess, such as mix-up [13] and few-shot classes duck filling. For

mix-up, it fuses two visually coherent images together into one output image by using Beta random distribution to improve the diversity of training set. And the mixtures are present in the Fig 1. For few-shot classes duck filling, we first extract the few-shot bounding boxes from training set, and then fill the few-shot objects into the non-annotated images in training set. In filling processing, some skills such as random weighted average method, random rescale for bounding boxes, are utilized to realize the mixing between few-shot objects and non-annotated images. The result images for few-shot classes duck filling are shown in Fig 2. Besides the two specific data argument methods, some regular augment methods are applied, such as random scale, random flipping, channel shuffling and random brightness contrast and so on.

2.2. Proposed Methods

We exploit both one-stage (such as FCOS [10], ATSS [12]) and two-stage (Cascade-RCNN [1]) as the basic detectors and evaluate their performance on validation dataset. And several classification networks are chosen as the backbone, such as HRNet-w36, HRNet-w48 [9], ResNext101-64, ResNext101-32 [11] and ResNet101 [5]. Comparing the detection performance of the above methods, we select the Cascade-RCNN as base network framework. And The ResNet101 is chosen as the backbone with FPN and deformable convolution (DCN). Besides, two other improvement skills are introduced to obtain a better performance — GRE-FPN and Hard IoU-imbalance Sampler.

GRE-FPN: At the regular FPN [7] stage, RoI features are usually extracted from one certain pyramid level according to the scales of RoIs. It ignores the importance of the adjacent scale features, which may contain more accurate location information. Therefore, we propose the Global RoI Extractor (GRE), which can extract RoI features from all pyramid levels and learn adaptive weights to balance the importance of features on different pyramid levels automatically. The detailed structure of GRE is illustrated in Fig 3. We first pool RoI features from all pyramid levels and concatenate them together. Then, these pooled features are convoluted with 1x1 convolution to reduce the channel dimension and obtain the final RoI features to predict objects.

Hard IoU-imbalance Sampler: We visualize annotations of objects on the respective images and find the annotations are sparse and coarse annotated. Besides, amounts of noise information are introduced and many targets should be labeled are missing. We call these lost targets unmarked targets. As for anchor-based network, the qualities of proposed anchors are significant for achieving an outstanding performance. Considering these images shown in Fig 4, we can observe that the marked target (the green box) may be surrounded by other unmarked targets (the red boxes).

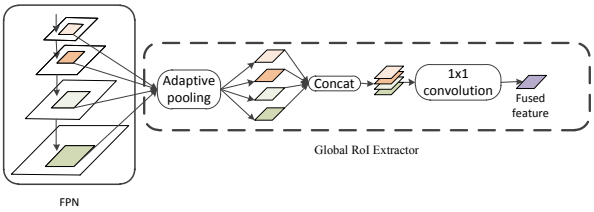


Figure 3. The details of Global RoI Extractor

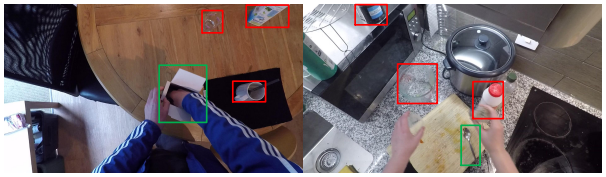


Figure 4. The annotation visualize examples

The regular sampling methods, such as Random Sampling, Online Hard Example Mining (OHEM), may generate false samples that take the unmarked target as negative samples and lead to the decreased robustness at the stage of RPN. To reduce the possibility of taking the unmarked targets as negative samples, we constrain negative sample regions that IoU ($IoU_{(S,T)}$) and center distance with marked target ($DIS_{(S,T)}$) are subject to

$$\begin{cases} IoU_{(S,T)} & < 0.3 \\ DIS_{(S,T)} & < (T_w, T_h)_2 \end{cases} \tag{1}$$

S is the proposal anchor, and T is the annotated target bounding box. T_w, T_h are the width and height of annotated target respectively. In this methods, the negative sample region always surround the marked targets rather than the whole input image, which can significantly reduce the influence of false sample. Besides, inspired by [8], we also improve the sampling ratio of hard negative examples with IoU in range $(0.05, 0.3)$ to further ensure the balance of sample processing.

3. Experiments

In this section, we conduct several experiments on EPIC-Kitchens object detection dataset. And the comparisons of detection performance are presented to verify the effectiveness of proposed method.

3.1. Experimental Settings

All the experiments are conducted by using the MMdetection toolbox which is developed with PyTorch by Multimedia Laboratory, CUHK. And we run our experiments on 8 NVIDIA P40 GPUs. The mini-batch Stochastic Gradient Descent (SGD) optimizer with momentum of 0.9 and

Cascade-rcnn	EPIC-Pretrain	CB	Multi-Test	SWA	IoU>0.05(%)	IoU>0.5(%)	IoU>0.75(%)
✓					71.11	43.22	17.12
✓	✓				72.01	43.67	18.01
✓		✓			72.77	44.67	18.55
✓			✓		72.45	44.51	18.32
✓				✓	71.97	43.78	17.89
✓	✓	✓			73.54	45.10	18.97
✓	✓	✓	✓		73.87	45.57	19.22
✓	✓	✓	✓	✓	**74.64**	**46.22**	**20.13**

Table 1. Comparison results on EPIC validation dataset. Performance measures contain mAP with ratio 0.05, 0.5, 0.75.

	method	Few shot classes(%)			Many shoy classes(%)			All classes(%)		
		IoU> 0.05	IoU> 0.5	IoU> 0.75	IoU> 0.05	IoU> 0.5	IoU> 0.75	IoU> 0.05	IoU> 0.5	IoU> 0.75
seen	base	47.52	26.01	7.56	36.10	39.59	10.79	60.17	37.03	10.18
	Mixup + duck	49.86	31.43	12.39	64.01	40.26	13.24	61.35	38.60	13.08
	Mixup + duck + Train tricks	47.07	27.01	11.79	65.68	41.72	13.92	62.18	38.95	13.52
	Mixup + duck + Hard IoU	52.48	**33.30**	13.29	67.27	41.16	12.91	65.23	39.68	12.98
	Mixup + duck + GRE-FPN	47.14	32.02	13.35	66.23	42.47	14.52	53.58	39.83	14.30
	Ensemble	**54.98**	32.40	**14.55**	**68.74**	**43.88**	**15.38**	**66.15**	**41.72**	**15.23**
unseen	base	20.07	13.42	1.93	54.83	31.92	8.11	51.29	30.04	7.48
	Mixup + duck	**42.95**	18.26	4.86	65.42	38.24	11.68	63.19	36.20	10.99
	Mixup + duck + Train tricks	39.44	**24.20**	**8.61**	66.44	38.85	12.79	63.69	37.36	12.36
	Mixup + duck + Hard IoU	40.99	22.66	6.82	66.62	39.13	12.42	64.01	37.45	11.85
	Mixup + duck + GRE-FPN	34.02	19.42	7.52	66.81	39.71	13.38	63.47	37.64	12.78
	Ensemble	35.75	22.31	7.33	**67.92**	**41.92**	**14.29**	**64.64**	**39.93**	**13.58**

Table 2. Comparison results on EPIC testing dataset on mAP. Ensemble results based on all designed variability. Challenge website details: https://competitions.codalab.org/competitions/20111. Note that our best model was submitted under the anonymous nickname DH-ARI

weight decay of $1 \times e^{-4}$ is utilized to solve the experiments. The input images are randomly sized to (1280×720) and (1394×764). The batch size is 32 and the maximum epoch for training is set to 12. The initial learning is fixed to 0.02. Then, it decays to $2 \times e^{-3}$ at epoch 8 and $2 \times e^{-4}$ at epoch 11. Meanwhile, we use 0.0067 to warm up the training until 500 iterations, then go back to 0.02 and continue training.

According to the number of bounding boxes for each category, the whole training set is divided into two parts : many shot (S1), few shot (S2). In the entire training phase, there are three stages: T1, T2, T3. In training phase T1, we choose datasets S1 to train and get a trained model (M1); In T2, we choose datasets S2 to fine-tune model M1 and obtain fine-tuning model (M2); And in the final phase T3, we use both datasets S1 and S2 to fine-tuning model M2 and obtain the final model.

3.2. Training Skills

In training and validation stage, we use several training skills to optimize the training procedure, such as Model Pre-training, Class Balance Sampling and Stochastic Weight Averaging. The test results on validation dataset is show in Table 1. The validation dataset is extracted from train dataset, and is divided into seen and unseen set.

Model Pre-training. We extract images based on bounding boxes and categories of objects from EPIC-Kitchens Object Detection datasets, and train a Classification model as our pre-trained model to replace default ImageNet pre-trained model.

Class Balance Sampling. During the training phase, image lists are randomly shuffled before the start of every epoch. Considering the long-tail class distribution and imbalanced categories, we randomly sample images in training list based on category probability as formula (2).

$$\begin{cases} W_i & = 1/S_{i_c} \\ S_{i_c} & = \sum_{j=0}^{m} \mathbb{R}_{(c,c_{i_j})} \end{cases} \tag{2}$$

Where W_i and S_{i_c} is the sampling possibility and the total numbers of category id c in the i^{th} image in training list respectively. c is the class ID that is annotated in image i and has the minimum proportion in the whole training dataset. m represents the total marked categories in the image i. c_{i_j} is the category of the j^{th} object in image i. And $\mathbb{R}_{(c,c_{i_j})} = 1$ only if $c = c_{i_j}$, otherwise $\mathbb{R}_{(c,c_{i_j})} = 0$. This

sampling methods can increase the sampling possibility of the few shot class and effectively solve the imbalanced categories problem.

Stochastic Weight Averaging (SWA). SWA [6] is based on averaging the samples proposed by SGD with a learning rate schedule that allows exploration of the region of weight space corresponding to high-performing networks. Using SWA, we achieve notable improvement over conventional SGD training on our base model and this method has almost no computational overhead.

3.3. Testing

In testing phase, we adopted the fusing results of multi-scale [4], flipping and Gaussian fuzzy transformation as the test output, which can further improve the detection performance. The detection results with different methods are presented in Table 2. Specifically, by using data augmentation proposed in Sec 2.1, proposed methods in Sec 2.2 and training tricks in Sec 3.2, the best performance obtained for seen and unseen results in a mAP of 39.93% with $IoU>0.5$. Using the duck filling technique, the recognition accuracy improved by 6.16% (30.04% vs 36.20%) on unseen set. By combining the data argument with training tricks, Hard IoU and GRE-FPN models separately, improvements of 1.16%, 1.25%, 1.44% are obtained respectively. With an ensemble of all the methods, the mAP of $IoU>0.5$ is further improved by 3.73% (36.20% vs 39.93%).

4. Conclusion

The proposed method for the EPIC-Kitchens object detection task is demonstrated in detail in this paper. Our main concerns are to moderate the long-tail class distribution of training set and extract more effective features.

The duck filling, mix-up and class balance sampling are introduced to expand the training set and moderate the long-tail distribution. And The Hard IoU-imbalance Sampler and the reconstructed GRE-FPN are also utilized to help extracting more representative object features. Experimental results demonstrate that our methods are effective and useful. By assembling these main methods, our detection framework can obtain a competitive performance on both the seen and unseen test data.

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. pages 6154–6162, 2018.

[2] D. Damen, H. Doughty, G. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[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] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 37(9):1904–16, 2014.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 2016.

[6] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. 2018.

[7] Tsungyi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. pages 936–944, 2017.

[8] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. 2019.

[9] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. pages 5693–5703, 2019.

[10] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. *arXiv: Computer Vision and Pattern Recognition*, 2019.

[11] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. pages 5987–5995, 2017.

[12] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. *arXiv: Computer Vision and Pattern Recognition*, 2019.

[13] Zhi Zhang, Tong He, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of freebies for training object detection neural networks. *arXiv: Computer Vision and Pattern Recognition*, 2019.

EPIC-Kitchens Object Detection Challenge 2020
VCL-Team Technical Report

Ioannis Athanasiadis, Lazaros Lazaridis, Athanasios Psaltis, Apostolos Axenopoulos, Petros Daras
The Visual Computing Lab - Centre for Research and Technology Hellas/Information Technologies Institute
{athaioan@, lazlazari@, at.psaltis@, axenop@, daras@} iti.gr

Abstract

In this report, the technical details of the VCL-team submission to the EPIC-Kitchens object detection challenge 2020 are presented. Although the modern two-stage object detection approaches can be trained to detect a huge variety of objects under conditions of high diversity, they rely completely on visual information. On the contrary, humans are capable of recognizing rarely seen and occluded objects by utilizing high-level information based on the reasoning mechanism. In order to evaluate the effect that prior-knowledge has in object detection, a set of experiments was conducted using a different kind of object-to-object relationship knowledge that was efficiently complemented by the understanding derived from the objects' affordance information. With the term affordance of an object, we are referring to all the possible ways the object's category can be manipulated during its usage.

1. Introduction

Object detection is the process of localizing and classifying the objects that appear in an image. Moreover, its numerous applications, ranging from self-driving cars to medical image processing, along with its challenging nature have attracted many researchers to this domain. More specifically, an object detection sub-case referring to egocentric viewing has been recently gaining popularity due to its immediate relation to the HCI field. In the EPIC-Kitchens dataset [2] a number of frames have been annotated in the form of bounding boxes, offering a large-scale egocentric view object detection benchmark. Compared to other frequently used object detection datasets, further difficulties are introduced due to annotations being made, based upon the action taking place rather than the objects' visual appearance explicitly. In this work, we address the EPIC-Kitchens object detection challenge by embedding a prior-knowledge graph in the object detection process, which imitates how humans utilize both their reasoning and the visual signal when recognizing objects.

2. Our Approach

2.1. Overview

Inspired by Reasoning-RCNN [5], in which relationship and attribute graphs are employed with the purpose of interpreting the human prior-knowledge, we investigate various ways of incorporating the objects' *affordance* combined with different kind of relationship graphs. The intuition behind this approach is to not only boost the object detection performance through exploiting additional high-level information, but also to target the particularity of annotations being dependent on the occurring actions by grasping the annotation pattern.

2.2. Faster R-CNN

Object detection in Faster R-CNN [4] is carried out through two discrete stages, one for region proposing and the other for region classification respectively. At first, a number of densely distributed regions of fixed scale and ratio, called *anchors*, are generated uniformly across the image. Thereafter, the Region Proposal Network (RPN) distinguishes the generated regions between foreground and background while also refining their size, shape, and location in a class agnostic manner. The regions falling in the foreground category are considered possible of containing a ground truth object and are proposed as such. In the second stage, each proposed region has a discriminative representation extracted by applying the RoI-Pool technique. Finally, based on their extracted feature descriptors, the regions are classified into one of the available classes as well as refined based on their predicted class.

2.3. Cascade R-CNN

Found on the observation that a classifier is trained to refine its input regions in such a way to better fit their corresponding ground truth targets, additional classifiers of increasing *quality* are introduced in [1]. More specifically,

Cascade R-CNN achieves increased object detection performance by simultaneously training multiple classifiers to progressively increase the proposed regions' alignment with the ground truth objects. By adopting the approach described above, each classification level is optimized at different IoU, while having the required thresholds increasing gradually, which guarantees that every classifier will have sufficient positive samples to be trained with.

2.4. Reasoning R-CNN

Reasoning R-CNN consists of two classification stages. In the first one, only visual information is considered for region refinement and classification, similar to how Faster R-CNN operates. Afterwards, the refined regions are fed to a second classifier of higher quality, as in Cascade R-CNN, which classifies and further refines its input regions by capitalizing on more informative feature descriptors complemented by high-level information as encoded by the reasoning module. Bellow a brief description is given, summarizing how the reasoning module computes the enhanced features as described in [5].

Reasoning module: At first, the initial classification layer's weight and bias parameters form the Global Semantic Pool $M \in \mathbb{R}^{C\times(D+1)}$, where C and D are the number of available categories and the length of the visual feature descriptors respectively. The M array contains the feature activation values for each category, thus comprises of high-level semantic information. An attention-like mechanism is implemented in order to focus on the relevant categories based on the input image. The image feature map is propagated through a *Squeeze-and-Excitation Network* [3] while average global pooling is applied resulting in an array $J \in \mathbb{R}^{1\times(D+1)}$. Afterwards the category-wise attention array $a \in \mathbb{R}^{1\times C}$ is calculated as shown in 1.

$$a = softmax(JM^T) \tag{1}$$

Let $G \in \mathbb{R}^{C\times C}$ be the knowledge graph depicting the class-to-class relations through different intensities and $W_g \in \mathbb{R}^{(D+1)\times E}$ be a trainable fully connected layer, where E defines the length of reasoning-based feature representation. Given the classification probability distribution $P \in \mathbb{R}^{N_r\times C}$ over the available classes for each of the N_r proposed regions, as predicted in the first classification stage, the enhanced descriptors $f^{'}$ are calculated as shown in 2. Finally the reasoning-based $f^{'}$ are concatenated with the regular visual descriptors which are then fed to the second stage classifier.

$$f^{'} = P(a\otimes GM)W_g \tag{2}$$

2.5. Cascade enhanced by Reasoning

Build upon the Reasoning R-CNN success in integrating the human-prior knowledge in the object detection process, we experiment with different kinds of relationship graphs as well as investigate different ways of assimilating the objects' affordances. In contrary to Reasoning R-CNN, where the various knowledge graphs are generated by manual annotation, we construct the graphs considering only the annotation of the training set. In this work, two variations of relationship graphs are considered, the *co-occurrence* graph (G^c) and *intersection* graph (G^i). The G^c contains information regarding the frequency with which various objects' categories have been annotated in the same frame. On the other hand G^i represents relations with respect to the overlap proportion exhibited between objects of different category. It is evident that the values in the diagonal of G possess no actual information. To address the uninformative diagonal we copy the values of an $A \in \mathbb{R}^{1\times C}$ array, which was generated based on the affordance intensities of each category. To evaluate the effect, affordance-based knowledge has in the object detection, we use it both separately and in combination with the relation-based knowledge graph on a third level of cascade classification.

3. Experiments

3.1. Implementation Details

Single-stage object detection: We consider Faster R-CNN as our baseline due to its expandability and its satisfactory object detection performance. The input images are resized to have their shortest dimension be 600 pixels wide. The anchors scales and ratios were set to $[32, 64, 128, 256, 512]$ and $[0.5, 1, 2]$ respectively. For the purpose of extracting the region feature descriptors, ROI-Pool is replaced with ROI-Crop as implemented in [6]. Additionally, *Non-maximum Suppression* (NMS) is applied both after the RPN and the classification stages with thresholds of 0.7 and 0.6 respectively. The network weights are initialized using the pre-trained model provided by the challenge's coordinators. Furthermore, only the heads of the network are trained for 7 epochs using a batch size of 32 images with an initial learning rate of 0.02 reduced by a factor of ten every other epoch.

Two-stage object detection: When embedding the knowledge graphs into the training process, an additional classification layer is introduced which requires the proposed regions to have an overlap of 0.6 IoU in order to be considered as a foreground object, contrary to the first stage in which 0.5 IoU was needed. The length of reasoning-based descriptor E is set to 512 while the visual-based one D is set to 2048. The rest of the setup is identical to the one described previously.

Three-stage object detection: In order to jointly exploit the knowledge derived from the objects' affordance along with the category-to-category graph, an extra classification level is added in which the enhanced reasoning

Methods	Seen			Unseen		
	$AP_{0.05}$	$AP_{0.5}$	$AP_{0.75}$	$AP_{0.05}$	$AP_{0.5}$	$AP_{0.75}$
Faster R-CNN	45.99	34.11	8.57	45.10	31.85	7.59
Reasoning G^c	**47.48**	**35.13**	8.89	**47.26**	**33.55**	8.64
Reasoning G^i	46.56	34.53	**10.00**	44.75	32.52	**9.54**
Affordance A	46.92	34.29	9.50	45.90	32.63	9.00
Affordance-based Reasoning G^A	39.37	29.24	7.45	43.69	31.56	7.85

Table 1. Object detection results.

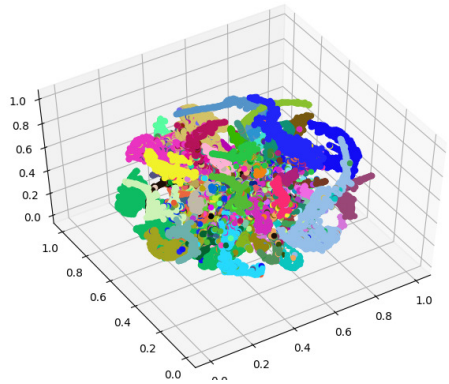


Figure 1. The enhanced features as extracted by the reasoning module with different colorization based on their class.

features are calculated by utilizing a G^A graph, which is created by copying the A affordance values on the diagonal of G.

3.2. Results

To evaluate the impact each knowledge graph has in the object detection performance, we report the *Average Precision* (AP) metrics in Table 1 as computed by the challenge's leaderboard. It is observable that the performance is marginally better when evaluated on the Seen test-tet compared to the Unseen one. Moreover, both relationship and affordance based knowledge seem to be boosting the baseline approach with the G^c outperforming the baseline's $AP_{0.5}$ by 1.7%. However, when G^c graph was used in combination with the A, on a third cascade level, the results were slightly below the baseline, probably because further training was required due to having more classifiers to train. In Figure 1 the enhanced feature have been projected to the R^3 space offering an intuitive visualization indicating that the features produced based on high-level knowledge have discriminative capabilities.

4. Conclusions

In the context of EPIC-Kitchens object detection challenge, a number of different prior-knowlegde graphs were evaluated based on their ability of enhancing the regularly used two-stage object detection approach. The results that have been yielded, suggest that prior-knowlegde, as derived by the training annotations, can be used to extract additional information that coupled with the visual one can provide detection results of increased quality. Additionally, even though the objects' affordances are indicative of their category, the performance achieved by their usage shows that more sophisticated techniques might be required in order to be better taken advantage of. Finally, generating the knowledge graphs solely based on a partially annotated dataset might introduces ambiguities limiting its potential.

Acknowledgment

This work was supported by the European Commission under contract H2020-820742 HR-Recycler.

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[2] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[3] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[5] Hang Xu, ChenHan Jiang, Xiaodan Liang, Liang Lin, and Zhenguo Li. Reasoning-rcnn: Unifying adaptive global reasoning into large-scale object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6419–6428, 2019.

[6] Jianwei Yang, Jiasen Lu, Dhruv Batra, and Devi Parikh. A faster pytorch implementation of faster r-cnn. *https://github.com/jwyang/faster-rcnn.pytorch*, 2017.