Quasi-Online Detection of Take and Release Actions from Egocentric Videos

Rosario Scavo[1,2(✉)], Francesco Ragusa[1,2], Giovanni Maria Farinella[1,2], and Antonino Furnari[1,2]

[1] FPV@IPLAB, DMI - University of Catania, Catania, Italy
rosario.scavo@studium.unict.it
[2] Next Vision s.r.l. - Spinoff of the University of Catania, Catania, Italy
https://iplab.dmi.unict.it/fpv/, https://www.nextvisionlab.it/

Abstract. In this paper, we considered the problem of detecting object take and release actions from untrimmed egocentric videos in an industrial domain. Rather than requiring that actions are recognized as they are observed, in an online fashion, we propose a quasi-online formulation in which take and release actions can be recognized shortly after they are observed, but keeping a low latency. We contribute a problem formulation, an evaluation protocol, and a baseline approach that relies on state-of-the-art components. Experiments on ENIGMA, a newly collected dataset of egocentric untrimmed videos of human-object interactions in an industrial scenario, and on THUMOS'14 show that the proposed approach achieves promising performance on quasi-online take/release action recognition and outperforms methods for online detection of action start on THUMOS'14 by +8.64% when an average latency of 2.19s is allowed. Code and supplementary material are available at https://github.com/fpv-iplab/Quasi-Online-Detection-Take-Release.

Keywords: Quasi-Online Action Detection · Low Latency · Take/Release Actions · Egocentric Untrimmed Videos

1 Introduction

Egocentric vision aims to analyze images and videos acquired from the user's point of view through a wearable device equipped with a camera (e.g., smart glasses) to understand how the user interacts with the environment and possibly provide assistance. Understanding users' activities and interactions with objects from egocentric visual signals allows to provide services to support humans in different domains such as homes, kitchens, museums, and industrial workplaces [2,4,13]. Since humans interact with objects using their hands, recognizing actions such as "take an object" and "release an object" can offer crucial insights into the user's intentions, especially in industrial environments.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-43153-1_2.

G. L. Foresti et al. (Eds.): ICIAP 2023, LNCS 14234, pp. 13–24, 2023.
https://doi.org/10.1007/978-3-031-43153-1_2

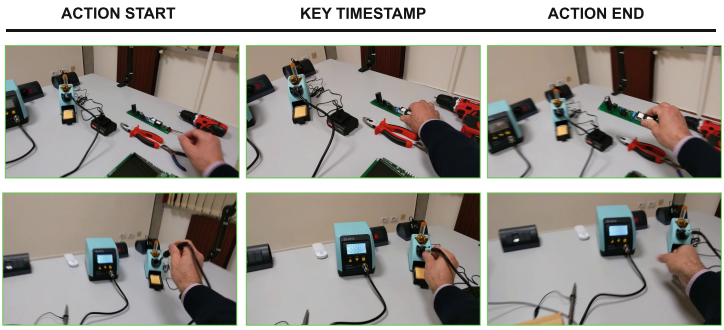

Fig. 1. Two examples of take (top) and release (bottom) actions. For each action, we define a key timestamp that corresponds to the first frame of contact (contact frame) in the take examples and to the last frame of contact (end-of-contact frame) in the release examples. These timestamps are distinct from classic action start and end time which are often ambiguous to annotate [11].

For instance, predicting the user's next action based on the tool they have taken, allows to provide assistance offering feedback to correct erroneous actions (e.g., "Please complete step X before picking up the pliers"). Additionally, this knowledge can be used to suggest ways to improve efficiency and reduce errors, such as recommending the optimal use of the object that was just picked up (e.g., triggering AR contents). Understanding take/release interactions taking place between humans and objects also allows to estimate the usage time of an object, possibly enabling predictive maintenance applications. Critically, such actions should be predicted in a timely fashion in order to provide useful assistance to workers as soon as possible.

In this paper, we consider the problem of detecting two key actions from egocentric videos: "take" and "release". These two actions occur respectively when the user takes an object and when they put it down. We assume a low-latency, quasi-online scenario in which take and release actions can be detected as soon as possible (e.g., in a few seconds) after they are observed from an input streaming video while aiming to keep a low latency to allow making decisions, such as sending an alarm, or notifying that a wrong action occurred in a maintenance procedure. We would like to note that the considered scenario is realistic and of practical relevance in contexts where the system aims to support the user, such as industrial environments. Indeed, in this context, an "after-the-fact" detection of actions with a low latency is useful for the verification of incomplete procedural tasks performed by workers.

To study the considered problem in the industrial domain, we collected and labeled ENIGMA, a dataset of egocentric videos in which several users performed repair and maintenance procedures on electrical boards in an industrial laboratory. Previous literature highlights how labeling start and end times, even for simple take and release actions, can lead to inconsistencies due to the limited agreement between different annotators [11]. In order to avoid bias due to these inconsistencies, each take and release action has been labeled by

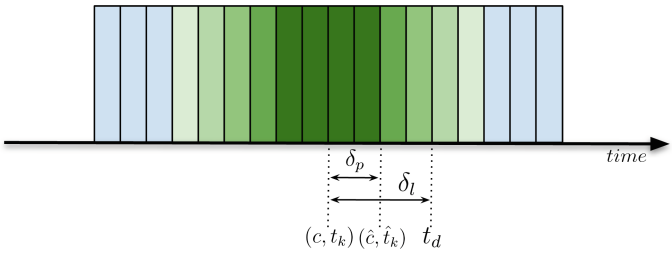

Fig. 2. Given an action of class c executed at the ground truth timestamp t_k, our goal is to estimate its key timestamp $\hat{t}_k$ and its class $\hat{c}$ while keeping a low latency $\delta_l = \max(0, t_d - t_k)$, where t_d is the timestamp in which the prediction is made. For a prediction to be deemed valid, we also expect the distance $\delta_p = |\hat{t}_k - t_k|$ between the estimated key timestamp and the ground truth one to be under a given temporal threshold ϕ.

marking a single timestamp indicating the first frame in which the hand touches an object (contact frame) in the case of take actions, or the first frame in which there is no more contact between the hand and the object (end-of-contact frame) for the case of release actions. The timestamps related to these frames are also referred to as "key timestamps" (see Fig. 1). We hence treat the problem of quasi-online take/release action detection as the one of processing an untrimmed input video and predicting a set of take/release action events with a low latency (see Fig. 2). Based on this problem definition, we designed an evaluation protocol aimed to assess the ability of the models to accurately detect the occurrence of take and release actions, as well as their latency. Therefore, we propose an approach to tackle this task based on a state-of-the-art transformer model for online action detection [19] in conjunction with a post-processing technique that aims to identify action occurrences from the analysis of time series of online prediction scores. Experiments on the collected dataset of egocentric videos show the feasibility of the proposed approach, which achieves promising results despite the task being challenging. Moreover, experimental results on THUMOS'14 demonstrate that our framework outperforms state-of-the-art online detection of action start methods by +8.64% when a quasi-online formulation is considered and an average latency of 2.19s is allowed, which shows the flexibility of the proposed problem formulation and approach in realistic scenarios.

The main contributions of this paper are as follows: 1) We investigate the problem of detecting take and release actions in egocentric videos in a quasi-online manner. 2) We designed an evaluation protocol to assess the accuracy and low-latency performance of the proposed models in predicting take/release actions. 3) We introduced a novel approach to address this task, utilizing cutting-edge transformers and a time series post-processing technique to refine the detection scores. Our code is publicly available to the research community to facilitate future research.

2 Related Works

Our investigation is related to previous works on online action detection, online detection of action starts, and human-object interaction detection.

Online Action Detection. Online action detection aims to detect an action as it happens from video and, ideally, even before it is fully completed [3]. While offline methods assume the entire video is available, online methods process the data up to the current time, predicting the action observed in each frame without looking at future frames. Different approaches over the years have been proposed to solve the online action detection task. Recent works employ Transformer architectures [15], which provide a more effective way to represent and model long data sequences than RNN architectures [10]. OadTR [17] is an encoder-decoder framework based on Transformers that recognize current actions by encoding historical information and predicting future context simultaneously. LSTR [18] explicitly divides the entire history into long- and short-term memories. TeSTra [19] is a state-of-the-art approach that uses the same strategy of LSTMs. TeSTra improves the computational efficiency of video transformers by applying temporal smoothing kernels to the cross-attention, resulting in streaming attention that only requires a constant time to update each frame.

The problem formulation proposed in this study is different from online action detection. Indeed, rather than expecting predictions to be made in real-time, we allow models to fully observe actions before determining whether an action should be predicted. While this formulation simplifies the prediction problem, we still assess that models have a low latency to ensure they are practically useful. We build on previous literature on online action detection by incorporating TeSTra into our framework to generate confidence scores for observed actions on a frame-by-frame basis.

Online Detection of Action Start. Some works have explored the problem of Online Detection of Action Start (ODAS) [14] from a third-person perspective. These approaches differ from online action detection in that its primary goal is to detect the start of an action as precisely as possible. Previous authors have employed methodologies such as 3D convolutions [14], a combination of LSTM and reinforcement learning [6], and weakly-supervised learning with video-level labels [7].

The ODAS formulation shares some similarities with the formulation of our problem but it is distinct. Indeed, while ODAS aims to detect the action start by observing the video immediately preceding the action, we aim to detect the key timestamp indicating the execution of a take or release action, after the complete observation of the action, but with a low latency.

Human-Object Interaction Detection. Human-Object Interaction (HOI) detection aims to identify and locate both the human and the object in an image or video and to recognize their interactions. The main HOI detection methods are based on the use of GCN after detecting humans and objects in the scene [12], on human-centric approaches [8], on the detection of the interactions

between human-object pairs [16], or on and grasp analysis [1]. In recent years, HOIs have also been studied in Egocentric Vision (Egocentric Human-Object Interaction - EHOI) [13]. However, only few works specifically considered industrial scenarios [13]. Moreover, while the aforementioned works have focused on understanding human-object interactions in still frames, we focus on detecting when take/release actions occur in a streaming video. We expect that our approach could be integrated with HOI analysis to provide a more accurate and grounded output.

3 Problem Definition and Evaluation Protocol

Let (c, t_k) represent a ground truth action, where c is the action class (take or release) and t_k is the related key timestamp (either contact or end-of-contact). Each prediction is represented as a $(\hat{c}, \hat{t}_k, t_d, s)$ tuple, where $\hat{c}$ and $\hat{t}_k$ are respectively the predicted class and key timestamp, t_d is the timestamp in which the prediction is actually made, and s is a confidence score. We define the temporal distance between the correct key timestamp, t_k, and the predicted one, $\hat{t}_k$, as $\delta_p = |\hat{t}_k - t_k|$. We will consider a prediction as correct if δ_p is under a given temporal threshold ϕ. Given a correct prediction, we define its latency as the difference between the timestamp in which the prediction is made, t_d, and the corresponding ground truth key timestamp, t_k: $\delta_l = \max(t_d - t_k, 0)$. Note that the $t_d - t_k$ difference is in general a positive number, but it may assume negative values in rare cases in which the action is predicted a few moments before it happens. In such cases, we will consider a latency equal to zero, hence the max operator in our definition of latency. We do not define latency for incorrect predictions. It is worth noting that, while in an online prediction scenario we impose $t_d = \hat{t}_k$, in the considered quasi-online scenario, we allow the two timestamps to differ, but expect a low latency δ_l. Figure 2 illustrates the considered problem.

Evaluation Protocol. Metrics generally used to evaluate action detection, such as mAP are not suitable in our scenario, as they assume that both action start and end are predicted, while in our case, an action is associated to a single timestamp. Instead, we adopt the point-level detection mAP (**p-mAP**) defined in [14]. Predicted actions are deemed to be correct only when 1) the predicted action class is correct ($c = \hat{c}$), 2) the temporal offset δ_p is smaller than a given evaluation temporal threshold ϕ. Predictions are matched to ground truth actions in a greedy fashion, prioritizing predicted actions with higher confidence scores, checking whether $\delta_p \leq \phi$, and imposing that ground truth actions and predictions are matched at most once. Based on these matches, the point-level Average Precision (AP) for each action class is evaluated and averaged over all action classes to determine the point-level mAP. Consistently with prior works [6,7], we evaluated the p-mAP at temporal offset thresholds ϕ ranging from 1 to 10 s in one-second increments. We defined **mp-mAP** as the average of p-mAP values calculated at different temporal offset thresholds ϕ.

Given our quasi-online problem definition, we complement point-level mAP with an evaluation of the average latency of a given approach. Specifically, we

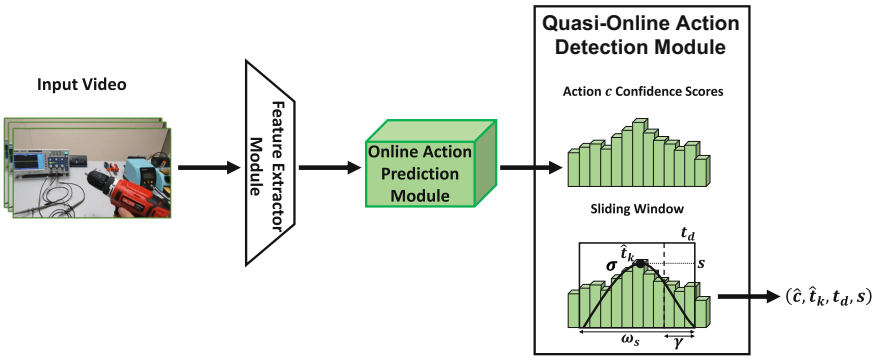

Fig. 3. Overview of the proposed method. Firstly, the input video is processed using a feature extractor. The resulting features are then fed into an online action prediction module, which outputs action confidence scores. Confidence scores are then passed to the quasi-online action detection module which processes them with a sliding window of size ω_s. Confidence scores within the window are smoothed using a Gaussian filter, and a peak detector is employed to temporally detect the action.

compute the average latency as follows: $\overline{\delta}_l = \frac{\sum_{i=1}^{N} tp(i) \cdot \delta_l(i)}{\sum_{i=1}^{N} tp(i)}$, where i is the index of the i-th prediction made by the model, N is the total number of predictions, $tp(i) = 1$ if prediction i is a true positive and $tp(i) = 0$ otherwise, and $\delta_l(i)$ denotes the latency of prediction i.

4 Proposed Approach

Our method comprises three main modules: 1) a feature extraction module that processes the input video and outputs per-frame features, 2) a transformer-based online action prediction module that takes the features as input and predicts actions along with their confidence scores frame-by-frame, in an online fashion, and 3) a quasi-online action detection module which takes as input a window of frame-by-frame action confidence scores and predicts the occurrence of actions in the considered window. Figure 3 shows the overall architecture of the proposed method. The following sections detail each component.

Feature Extraction Module. The feature extraction module takes as input the streaming video and produces per-frame high-level representations. This module may analyze input video clips or single frames. In this work, following [5], we extract per-frame features using a Two-Stream (TS) CNN model. In particular, the TS-features comprise appearance features that focus on the video's visual appearance information and motion features that rely on the user's and objects' movement during the actions. Features are then fused to obtain a unique representation for each video chunk, which is passed to the transformer-based online action prediction module.

Transformer-Based Online Action Prediction Module. We base this module on the state-of-the-art online action prediction approach TeSTra [19]. TeSTra takes pre-extracted features as input and outputs per-frame probability distributions of action classes. In practice, this module allows to predict two confidence scores at each frame: the probability of take actions, and the probability of release actions.

Quasi-Online Action Detection Module. This module takes as input time series of confidence scores produced by the online action prediction module and analyzes them to predict the presence of take/release actions, along with the estimated timestamps in which they take place and associated confidence scores. Rather than requiring the system to predict whether an action is performed in the current frame, we process the time series of predicted confidence scores with a sliding window of size ω_s up to the current timestamp t, and allow the prediction of actions observed within the window. We expect the predicted confidence score of an action to increase before the occurrence of the key timestamp and to decrease after it, as shown in Fig. 4-left. Hence, we aim to predict actions by detecting peaks in the confidence score time series. In practice, confidence score time series tend to be noisy due to the uncertainty of the model and the ambiguity of observations (Fig. 4-middle). To account for this, a Gaussian filter is applied within the current window with a fixed standard deviation σ in order to provide a smooth time series of confidence scores. We hence use a peak detector[1] to find the occurrence of each take/release action. Each peak is considered as a prediction, and the peak height is the associated confidence score (see Fig. 4-right). To avoid predictions due to incomplete observations, we ignore all predictions made in the last γ seconds of the sliding window. We refer to γ as "inhibition time". Note that predictions suppressed at time t because they fall in the inhibition time interval can be recovered later by the model when the sliding window has moved forward and the same prediction does not fall within the inhibition time segment anymore. By iterating over the video with the proposed sliding window approach, the same action may be detected more than once in the video, due to the natural overlap between the considered prediction windows. To avoid multiple detections of the same action, we discard new detections whose difference with previously made predictions is under a given threshold ξ.

5 Experimental Settings and Results

In this section, we report the settings and results of our experimental analysis aimed at evaluating the proposed problem and approach. Please also see the supplementary material for the implementation details and additional analysis of the results.

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html.

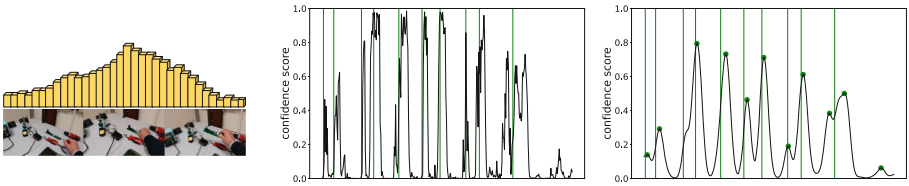

Fig. 4. Left: Ideally, predicted confidence scores of an action should increase before the occurrence of the key timestamp and decreases after it. Middle: In practice, the confidence scores tend to be noisy due to the uncertainty of the model and the ambiguity of observations. Green vertical lines represent ground truth key timestamps. Right: To mitigate the effect of noise, we apply Gaussian smoothing before performing peak detection. Green circles represent the detected peaks. As can be noted, Gaussian smoothing greatly improves the ability to detect key timestamps. (Color figure online)

5.1 Datasets

We perform our experiments on two datasets: ENIGMA, a dataset of egocentric videos collected and labeled for this study, and THUMOS'14 [9].

ENIGMA. In order to study the problem, we collected and labeled a set of egocentric videos of subjects simulating testing and repairing procedures on electrical boards using different laboratory tools. While performing the procedures, the subjects naturally interacted with 13 different objects. We labeled each take and release action with a single key timestamp, denoted with t_k. In the case of take actions, the key timestamp corresponds to the first frame in which the hand touches an object (contact frame), whereas, in the case of release actions, it corresponds to the first frame in which there is no more contact between the hand and the object (end-of-contact frame). See Fig. 1 for two examples. We acquired and labeled a total of 53 videos, which account for 23 hours, 14 minutes, and 8 seconds of video at a resolution of 2272×1278 pixels with a framerate of $30fps$. We randomly divided the videos into a train, validation, and test set, keeping balanced numbers of take and release actions.[2]

THUMOS'14. We also report results on the THUMOS'14 [9] dataset to assess the ability of the proposed approach to generalize to the problem of online detection of action start, which is closely related to ours. Following prior works [6,7], we evaluated on a test set of 213 untrimmed videos.

5.2 Quasi-Online Detection of Take and Release Actions Results

As previously discussed, the proposed approach relies on a set of parameters, and more precisely: ω_s, the window size, γ the inhibition time, σ the standard deviation of the Gaussian used to smooth the predicted confidence scores, and ξ, the minimum distance at which predictions should be made to avoid multiple

[2] For detailed statistics regarding the dataset, please refer to the supplementary material.

Table 1. Evaluations of mp-mAP on ENIGMA for different choices of parameters. Best results per column are reported in **bold**.

mp-mAP (%)	$\overline{\delta}_l$ (s)	w_s (s)	γ (s)	σ (s)
16.46	1.48	2	0.8	0.6
13.72	**1.16**	1	0	0.2
16.08	1.35	2	0	0.6

predictions of the same action. We performed 72 unique experiments considering different combinations of the aforementioned parameters, for ω_s varying in the range of [1, 2, 3, 4, 5] seconds, γ varying in the range of [0, 0.2, 0.4, 0.6, 0.8, 1] seconds, and σ varying in the range of [0.2, 0.4, 0.6] seconds. We set $\xi = 2s$. Figure 5-left report boxplots summarizing the distributions of mp-mAP% and average latency $\overline{\delta}_l$ values obtained in the different experiments. As can be noted, while some parameter combinations allow to achieve better results than others, detection performance and average latency values tend to be robust to the choice of parameters. Table 1 reports the performance of the proposed approach for some selected choices of the considered parameters. As can be noted, there is a trade-off between optimizing mp-mAP and reducing latency, but we obtain balanced results setting $\omega_s = 2s$, $\gamma = 0s$, and $\sigma = 0.6s$, with $mp - mAP = 16.08\%$ and $\overline{\delta}_l = 1.35s$. As can be noted, the proposed approach achieves promising results, but the problem is challenging and there is still room for improvement. Based on these experiments, it is clear that extending the inhibition time leads to better overall mp-mAP outcomes. However, this improvement results in physiologically higher latency. On the other hand, choosing a shorter inhibition time provides a better balance between mp-mAP and latency. This can be explained by observing that the confidence score distribution is almost uniform when far from the start of an action, and the peak detector does not detect a peak due to the Gaussian smoothing.

Average latency gives an indication of the ability of the model to make predictions on time. We further explore how detection performance changes when a given latency threshold is considered. Specifically, given a latency threshold ϵ, we deem as incorrect all predictions with a latency $\delta_l > \epsilon$ and re-compute mp-mAP. Figure 5-right reports mp-mAP% values for different latency thresholds. As can be noted, imposing a practical threshold of about $1s$ reduces mp-mAP% only by about 5% points, leading to a value of about 11%.

5.3 Generalization of the Proposed Approach to the Online Detection of Action Start Problem

The considered problem is closely related to previous investigations on the Online Detection of Action Start problem [6,7,14]. We hence assess how our approach generalizes to such a problem on the THUMOS'14 dataset. Figure 6-left reports the boxplots of the distributions of mp-mAP% and average latency for different parameter choices on THUMOS'14. Also in this case, results are stable for

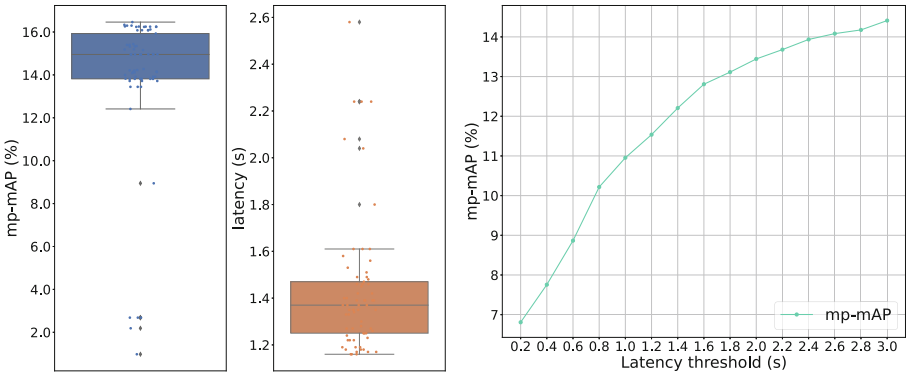

Fig. 5. Left: Boxplots showing the distributions of mp-mAP and average latency. Right: plots showing how mp-mAP% varies using different latency thresholds. Both plots are obtained on ENIGMA.

Table 2. Comparisons of p-mAP under different temporal offset thresholds on THUMOS'14 for online detection of action start. **Best** and second-best per column are highlighted.

	p-mAP temporal offset threshold ϕ (s)										mp-mAP (%)
	1	2	3	4	5	6	7	8	9	10	
StartNet [6]	21.9	33.5	39.6	42.5	46.2	46.6	47.7	48.3	48.6	49	42.39
WOAD [7]	**28.0**	**40.6**	45.7	48.0	50.1	51.0	51.9	52.4	53.0	53.1	47.38
Ours	17.15	37.82	**48.62**	**55.70**	**60.75**	**64.37**	**67.17**	**68.33**	**69.69**	**70.63**	**56.02**

different parameter choices. We set $\omega_s = 4s$, $\gamma = 0s$ and $\sigma = 0.25s$ for these experiments.[3] Table 2 compares the proposed approach to StartNet [6] and the state-of-the-art WOAD [7], both in terms of p-mAP at different temporal offset thresholds ϕ and mp-mAP. It is worth noting that both competitors aim to perform online detection of action start, while our method focuses on quasi-online detection. As can be noted, the quasi-online relaxation allows our approach to obtain improved performance (+8.64%). The average latency of our approach is $2.19s$. Figure 6-right finally compares mp-mAP performance for different latency thresholds. It is worth noting that, when low thresholds are considered, forcing the model to make online predictions, the mp-mAP performance of the model is strongly reduced to about 9%. However, a threshold of $1.5s$ still allows to achieve an mp-mAP performance of about 23% despite the model not explicitly being designed to tackle this task.

[3] See the supplementary material for a study on the influence of the different parameters on THUMOS.

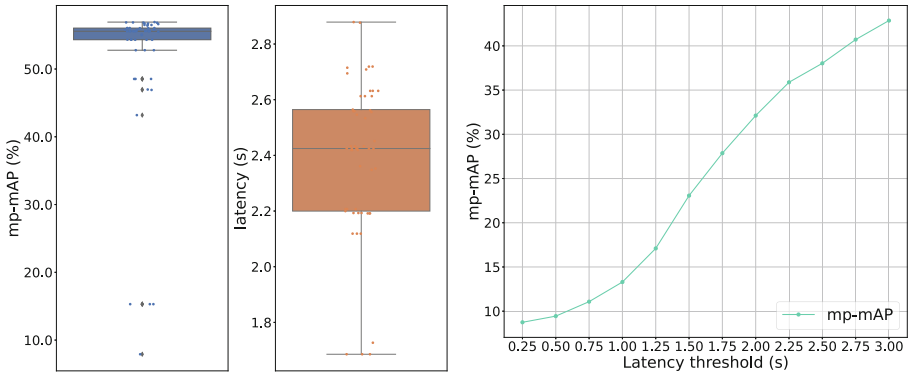

Fig. 6. Left: Boxplots showing the distributions of mp-mAP and average latency. Right: plots showing how mp-mAP% varies using different latency thresholds. Both plots are obtained on THUMOS'14.

6 Conclusion

We study the detection of take and release actions from egocentric videos in a quasi-online fashion. We propose a problem formulation and an initial approach to tackle the task. Experiments show promising results, but the problem is challenging and there is still space for improvement.

Acknowledgements. This research has been supported by Next Vision s.r.l., by the project MISE - PON I&C 2014-2020 - Progetto ENIGMA - Prog n. F/190050/02/X44 - CUP: B61B19000520008, and by Research Program Pia.ce.ri. 2020/2022 Linea 2 - University of Catania.

References

1. Besari, A.R.A., Saputra, A.A., Chin, W.H., Kubota, N., et al.: Feature-based egocentric grasp pose classification for expanding human-object interactions. In: 2021 IEEE 30th International Symposium on Industrial Electronics (ISIE), pp. 1–6. IEEE (2021)
2. Damen, D., et al.: Scaling egocentric vision: the dataset. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11208, pp. 753–771. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01225-0_44
3. De Geest, R., Gavves, E., Ghodrati, A., Li, Z., Snoek, C., Tuytelaars, T.: Online action detection. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9909, pp. 269–284. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46454-1_17
4. Farinella, G.M., et al.: VEDI: vision exploitation for data interpretation. In: Ricci, E., Rota Bulò, S., Snoek, C., Lanz, O., Messelodi, S., Sebe, N. (eds.) ICIAP 2019. LNCS, vol. 11752, pp. 753–763. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30645-8_68

5. Gao, J., Yang, Z., Nevatia, R.: RED: reinforced encoder-decoder networks for action anticipation. arXiv preprint arXiv:1707.04818 (2017)
6. Gao, M., Xu, M., Davis, L.S., Socher, R., Xiong, C.: StartNet: online detection of action start in untrimmed videos. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5542–5551 (2019)
7. Gao, M., Zhou, Y., Xu, R., Socher, R., Xiong, C.: WOAD: weakly supervised online action detection in untrimmed videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1915–1923 (2021)
8. Gkioxari, G., Girshick, R., Dollár, P., He, K.: Detecting and recognizing human-object interactions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8359–8367 (2018)
9. Idrees, H., et al.: The THUMOS challenge on action recognition for videos "in the wild". Comput. Vis. Image Underst. **155**, 1–23 (2017)
10. Karita, S., et al.: A comparative study on transformer vs RNN in speech applications. In: 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 449–456. IEEE (2019)
11. Moltisanti, D., Wray, M., Mayol-Cuevas, W., Damen, D.: Trespassing the boundaries: labeling temporal bounds for object interactions in egocentric video. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2886–2894 (2017)
12. Qi, S., Wang, W., Jia, B., Shen, J., Zhu, S.-C.: Learning human-object interactions by graph parsing neural networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11213, pp. 407–423. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01240-3_25
13. Ragusa, F., Furnari, A., Farinella, G.M.: MECCANO: a multimodal egocentric dataset for humans behavior understanding in the industrial-like domain. arXiv preprint arXiv:2209.08691 (2022)
14. Shou, Z., et al.: Online detection of action start in untrimmed, streaming videos. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11207, pp. 551–568. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01219-9_33
15. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
16. Wang, T., Yang, T., Danelljan, M., Khan, F.S., Zhang, X., Sun, J.: Learning human-object interaction detection using interaction points. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4116–4125 (2020)
17. Wang, X., et al.: OadTR: online action detection with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 7565–7575 (2021)
18. Xu, M., et al.: Long short-term transformer for online action detection. Adv. Neural. Inf. Process. Syst. **34**, 1086–1099 (2021)
19. Zhao, Y., Krähenbühl, P.: Real-time online video detection with temporal smoothing transformers. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision-ECCV 2022: 17th European Conference, Tel Aviv, Israel, 23–27 October 2022, Proceedings, Part XXXIV, vol. 13694, pp. 485–502. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19830-4_28