Multi-Level Confidence TAL for Naturalistic Driving Behavior

Jared Chan, Ziming Zhang
Worcester Polytechnic Institute
{jchan3, zzhang15}@wpi.edu

Abstract

Distracted driving is a common danger that threatens the safety of many and results in numerous deaths every year. As a means to further study such behavior and develop techniques that improve temporal annotations and action detection, AI City has released the 2023 Track 3 (Naturalistic Driving Action Recognition) challenge, which provides synthetic distracted driving data in the form of untrimmed long videos. The goal of the challenge is to identify the start and end times of different distracted driving actions, recorded from three distinct camera locations inside a vehicle. In this paper, we propose a framework for performing temporal action localization (TAL) over the challenge dataset, which leverages the general characteristics of the data to improve action localization. Specifically, we make use of a fine-grained action recognition model to generate prediction probabilities over fixed length temporal segments and then utilize our unique post-processing algorithm to analyze temporal action boundary candidates at various confidence levels. Our solution ranks 6th on the Track 3 public leaderboard and achieves a score of 0.6010 on the A2 test dataset. The source code is available at https://github.com/CarrotPeeler/AI_City_2023_Release.

1. Introduction

Distracted driving claims a significant amount of lives each year, having killed over 3,100 people and leaving 424,000 injured in just 2019 alone [1]. Everyday tasks, such as texting, calling, talking, or listening to music may appear to be harmless, but in reality, they are the most common reasons for why vehicular accidents occur [1]. To counteract such safety hazards, Computer Vision has become a popular and widely explored tool for developing real-time solutions that identify and prevent distracted behavior. However, a lack of sufficient untrimmed video data and label quality has led to obstacles in producing effective solutions.

In response to this, AI City has set up and released the Track 3 (Naturalistic Driving Action Recognition) competition, which is part of the broader 2023 AI City Challenge [2]. The aim of the challenge is to develop a system that localizes the temporal boundaries of various distracted driving actions, given an untrimmed long video. The data provided, SynDD2 [3], is synthetic and is collected from three different camera locations inside of a vehicle. The data is split into two datasets, A1 for training and A2 for validation, both of which total to 180 videos (27 hours in length). Additionally, each untrimmed video contains 16 unique actions according to their annotations, with a few rare exceptions, such as duplicate action occurrences in some videos. One key aspect of the challenge dataset is that action segments range anywhere from approximately 2 to 30 seconds in length, which is much longer than clips in popular benchmark datasets, such as Kinetics [4]. Furthermore, some videos have the driver's face blocked with a hat or sunglasses while others do not. On top of the aforementioned points, the action categories for classification are fine-grained such that some classes may have the driver performing identical actions with subtle differences. Therefore, part of the problem involves strategizing solutions for how to prepare and utilize the data to maximize the accuracy of the system's recognition and localization capabilities.

To effectively capture the temporal boundaries of actions, we use overlapping proposal generation and develop a temporal action localization framework that leverages the characteristics of the data to apply fine-grained post-processing.

1.2. Contributions

The main contributions of this paper are as follows:

- We propose a unique fine-grained post-processing approach which generates temporal action boundary candidates and narrows down the candidate search space via thresholding, merging, selection, and linking steps.

- We present a thorough study and analysis of the synthetic driving data to gain insight into how action localization can be improved for future works.

2. Related Work

2.1. Video Recognition

Video recognition refers to a wide variety of computer vision tasks which involve understanding videos and their content. Within video recognition are more specific tasks, such as object detection, action recognition, and many more [5]. Action recognition is a computer vision task that aims to classify sequences of human actions within videos. Temporal action localization (TAL) is another field under action recognition which not only deals with classifying different categories of actions but extracting temporal information as well. When developing action recognition models, some of the most popular approaches have been to apply Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), attention mechanisms, or a combination of the aforementioned concepts [6]. In recent years, however, new research has proposed the idea of purely attention-based models for action recognition, Vision Transformers, which omit the usage of CNNs and RNNs and even outperform large-scale CNN-based architectures [5].

2.2. Vision Transformers

Purely attention-based Vision Transformers were first proposed in [6]. Attention mechanisms increase the robustness and efficiency of model understanding by redirecting a model's 'attention' towards only learning the most important parts of input data. By using multiple attention mechanisms to form a multi-head self-attention layer, the representation of input is better defined and is less computationally intensive. Therefore, Vision Transformers can capture long-term human activities while ignoring redundant information in the process [5].

Current SOTA transformers for video recognition tasks include but are not limited to VideoMAEv2 [7], MViTv2 [8], SlowFast [9], and X3D [10]. In particular, VideoMAEv2 has shown not only to outperform all the aforementioned models on popular action recognition benchmarks such as Kinetics 400 and 600, but it also outperforms all current models benchmarked on the FineAction [22] dataset, which contains fine-grained action clips. VideoMAEv2 uses dual-masked autoencoding to reduce overfitting and effectively learn downstream tasks with minimal computation cost. All backbone models also undergo universal self-supervised pre training on a large-scale unlabeled dataset with 0.66M clips [7]. Overall, VideoMAEv2's architecture and implementation make it highly suitable for fine-grained recognition tasks, especially over small datasets where overfitting is prone to occur.

2.3. Temporal Action Localization

Current solutions for TAL consist of one-stage and two-stage methods. Two-stage methods tend to be more complex and work by first generating candidate intervals (proposals) where actions are likely to occur and then classifying these segments to refine temporal boundaries. In contrast, one-stage approaches perform both steps, simultaneously, to localize temporal boundaries and classify actions, forgoing the use of proposals. Recent one-stage methods mainly consist of boundary detection networks [11, 12, 13].

Among past top performers [14, 15] in the AI City challenge, the only effective one-stage method used has been ActionFormer [16], which uses a transformer to classify moments of an untrimmed video and then regresses action boundaries in a single shot. While the 2023 7th place challenge submission [14] uses ActionFormer to achieve top results, their method employs the usage of three separate models (X3D and MViTv2 for classification fusion, and ActionFormer for TAL), limiting the efficiency and scalability of the method when applied to other datasets.

Other one-stage solutions for TAL utilize aggregation of different contexts. For example, DCAN [12] opts to aggregate boundary and proposal level contexts to generate high-quality candidate intervals where actions might occur. Another method is DAPs [17], which aggregates temporal context (video frames) across a sliding window to generate variable length proposals of different temporal scales. However, top performers [18] have noted the downsides to one-stage methods in general, such as lower accuracy compared to two-stage solutions and the low volume of data provided by the challenge to effectively train boundary detection networks.

Overall, a large portion of the winners from past challenges [18, 19, 20, 21, 24] opted to use two-stage methods, which generate overlapping proposals every 16 frames with a temporal resolution of approximately 2 seconds or 64 frames at 30 FPS. Using anchor windows, their solutions favored exploring ways to improve classification and post-processing techniques rather than proposal generation. MViTv2 and X3D have been commonly employed as the backbone for classification.

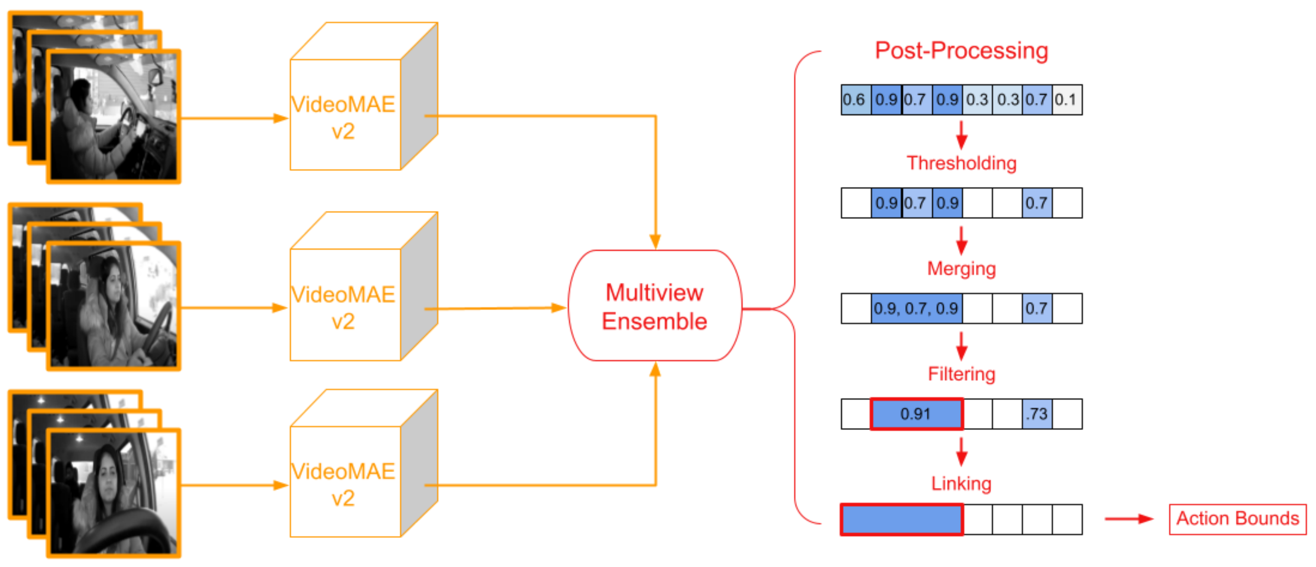

Figure 1. Proposed TAL Framework. For inference over a single video, all frames are grouped and split into fixed length action proposal segments. We generate 64 frame action proposals every 16 frames to create overlap between adjacent proposals, ensuring more precise prediction results. This process is performed three times, as the dataset provides three videos per video ID, each of which captures the same set of events from a different camera location. Following proposal generation and preprocessing, VideoMAEv2 is used to obtain prediction results from proposals. In post-processing, a multiview ensemble is produced from different camera view results and then a series of steps—thresholding, merging, selection, and linking—are applied over the ensemble to obtain action bounds.

3. Methodology

3.1. Classification

The A1 training dataset provided by the Track 3 challenge contains 150 untrimmed long videos (22.5 hours in length). However, after extracting meaningful clip segments from the videos according to annotated start and end times of labeled actions, we find the resulting dataset to be only 2409 clips in size. We then further divide the total pool of clips into three subsets based on the three camera locations used for recording videos inside a vehicle. Subsequently, the size of each subset of 811 clips, and is split into 80% training and 20% validation. Due to the fine-grained nature of the dataset, we train three separate models, one for each data subset, and perform a multiview ensemble to combine model results. As the data subsets are rather small in size, we utilize well known augmentation techniques and a suitable action recognition model to prevent overfitting.

3.1.1. Fitting the Data. Based on the success of past submissions [20] which have used VideoMAE [30] for action recognition, we explore the more recent VideoMAEv2. For the backbone, our model uses ViT-B/16 with pre-trained weights learned from Kinetics-710 [23]. To preprocess each subset of the A1 data, we apply spatio-temporal sampling methods to uniformly subsample action clips down to 16 frames and perform random spatial cropping. We also employ the use of several state-of-the-art augmentation techniques that are well known for reducing overfitting and boosting generalization: Mixup [25] and CutMix [26]. Mixup creates new synthetic samples from existing data by blending images and labels, while CutMix cuts and pastes image patches together. Additionally, we utilize other random data augmentation techniques as well, such as Random Erasing [27], which randomly erases a rectangular portion of an image and its pixels, and RandAugment [28], which performs random image distortions automatically using a multitude of hyperparameters.

To ensure the data is properly fitted using the above augmentation and preprocessing techniques, we perform 5-fold cross validation for each model trained on a particular camera view. After cross-validation, we ensemble the accuracy results across all folds for each model by calculating the mean accuracy. The multi-fold ensemble accuracies help determine the reliability of each trained camera view model and indicate the weight that should be given to each model in the computed multiview ensemble. Once each model is properly tuned and fitted, we retrain

3.2. Temporal Action Localization

To accomplish TAL, we propose the following architecture in Figure 1 which ensembles proposal prediction probabilities from different camera locations and then performs post-processing steps, such thresholding, merging, selection, and linking. Because the synthetic A1 and A2 datasets are designed to have 16 unique action class instances appear in each video, our approach assumes every action occurs once in a single video. Therefore, after we produce a multiview ensemble, we separate and analyze probabilities by action class to perform TAL. Our approach aims to gradually narrow down the candidate space for action bounds by incorporating prior knowledge of the characteristics of true-positive action segments.

3.2.1. Individual Class Analysis. Because the Track 3 A1 and A2 datasets attempt to mimic the realistic fine-grained nature of distracted driving behavior, the detection of actions is not a perfect process. We observe that while drivers may carry out the same action task, there are various nuances to how they can enact that particular action. Therefore, with such a small training dataset, there are occurrences where action recognition models detect true-positive action segments with low confidence. As such, the confidence levels which attribute to true-positive action instances may depend on not only the action class but the driver as well. Therefore, simply taking the argmax class with respect to probabilities for a single proposal is not sufficient. We find that the class with the highest probability for a prediction may still have relatively low confidence with respect to other probabilities corresponding to the same class. In turn, applying the argmax operation over each proposal's probabilities may discard and ignore other potential action candidates whose probabilities are lower than that of the argmax but relatively high with respect to their class. To solve this problem, we assess the probabilities in our multiview ensemble by class to ensure all action classes are fairly considered for a single proposal.

3.2.2. Thresholding. Since the confidence threshold at which actions are deemed true-positives depend on both the type of action and the driver's individualistic behavior, We create relative thresholds that account for such variations in driver behavior and action types. Eq. 1 describes the formulation for a relative threshold given a probability array for a single action class $p \in \mathbb{R}^t$, where t is the number of proposals in a video.

$$P_{th}(p) = max(p) * r \qquad (1)$$

The relative threshold is derived from the max probability given r, which is a fraction of the max probability. We set $r = 0.75$ to highlight only the most salient proposals with relatively high confidence. Other values such as 0.5 and 0.9 are either not high enough to remove most noise or are too severe for thresholding.

3.2.3. Merging. Following probability thresholding, we locate initial candidate segments where an action is most likely to occur via a merge operation. The merge operation will continuously group together temporally consecutive proposals where the difference between the end time of the previous proposal and the start time of the next proposal is within a designated linking threshold. Otherwise, if the difference surpasses the threshold, the next proposal will be the first in a new grouping while the previous proposal marks the end of the former grouping. After a proposal grouping is complete, the start time of the first proposal and the end time of the last proposal in the grouping become the start and end time of the corresponding merged segment. In the case where thresholding filters out all proposals for an action class, the merging operation will return an invalid value to indicate such an occurrence.

As thresholding may remove low confidence proposals from the middle or edges of an action segment, linking thresholds for the merging process are typically 2 seconds for most classes, with a few exceptions such as class 13 and 15. These classes have longer thresholds of 10 and 15 seconds, respectively, to account for common noise in their probabilities which causes wide gaps to form where their true-positive class instances typically occur.

3.2.4. Selection. To select the candidate segment where the true-positive instance is most likely to occur, we employ a selection process which considers two factors of a merged temporal segment: the mean probability of all proposals in the segment and the duration of the segment. For each merged segment, we compute a candidate score, which accounts for both of the previously mentioned aspects. Eq. 2 shows the formulation for the candidate score. $|T|$ is the duration of the merged segment in seconds; $\overline{p}_T$ is the mean probability of all proposal probabilities within the merged segment; β is a constant value, which is applied to the duration of a segment.

$$cs(|T|, \overline{p}_T) = \overline{p}_T + |T| \cdot \beta \qquad (2)$$

We take the merged segment with the highest candidate score as the true-positive action bound.

3.2.5. Linking. Using the selected action bound, we perform an endpoint refinement process, which may alter the start or end time of the action bound as necessary. Because of the relative thresholding process, there is a non-trivial possibility of losing temporal information which helps understand whether two merged segments should be a part of the same segment, or if the start or end time of a merged segment is slightly inaccurate. Therefore, to ensure such key proposals are not excluded in the final action bound, we devise a linking algorithm which examines all filtered proposals from the thresholding process and determines if any should be considered as a part of the final action bound.

The process is as follows: two search spaces are generated for linking. The left search space is composed of all the proposals that temporally occur before the start time of the final action bound and includes the *first* proposal within the final action bound. The right search space is made up of all the proposals that temporally occur after the end time of the final action bound and includes the *last* proposal within the final action bound. We then run the merging operation for each search space over multiple confidence levels, generated from 0.1 increments inclusively between 0.2 and the relative threshold. Note, the linking threshold for all action classes is set to be smaller than before (~0.5 seconds) since evaluation over lower confidence levels introduces the possibility of merging noisy proposals.

Within the left search space, if any proposals are merged into the first proposal within the final action bound, we denote the new final action bound start time as the start time of this newly merged segment. Similarly, within the right search space, the same process occurs and the new end time is denoted as the end time of the newly merged segment which includes the last proposal of the final action bound from before. If no merging occurs with the first or last proposal of the final action bound, the start or end time does not change. From the merge operation over different confidence levels, we obtain multiple start and end times which may or may not differ from one another.

As performing the merge operation on lower confidence levels may cause noisy proposals to be merged into the final action bound, we apply a Non-Maximum Suppression (NMS) algorithm with weighted interval fusion to compare how the final action bound was changed at different confidence levels. Eq. 3 describes the formulation for obtaining the refined start and end time over multiple confidence levels. $p_c \in \mathbb{R}^{|c|}$ is an array storing mean prediction probabilities for merged segments at different confidence levels. $c \in \mathbb{R}^{g}$ is an array storing the values of all confidence levels, where g is the number of 0.1 increments inclusively between 0.2 and the relative threshold P_{th}. $|c|$ denotes the number of confidence levels. $T \in \mathbb{R}^{|c|}$ is an array either storing the refined start or end times of the final action bound at multiple confidence levels.

$$nms(p_c, c, T) = \frac{\sum_{i=1}^{|c|} T_i \cdot p_{c,i} \cdot c_i}{\sum_{i=1}^{|c|} p_{c,i} \cdot c_i} \quad (3)$$

After performing NMS on both the merged segments from the left and right search spaces, we obtain the final fused start and end time of the temporal segment where an action is most likely present.

Class ID	Description
0	Normal Forward Driving
1	Drinking
2	Phone Call (Right)
3	Phone Call (Left)
4	Eating
5	Text (Right)
6	Text (Left)
7	Reaching Behind
8	Adjust Control Panel
9	Pick Up From Floor (Driver)
10	Pick Up From Floor (Passenger)
11	Talk to Passenger at the Right
12	Talk to Passenger at Backseat
13	Yawning
14	Hand on Head
15	Singing or Dancing With Music

Table 1. Class IDs and Descriptions. There are a total of 16 class actions that drivers perform throughout videos in the A1 and A2 datasets from Track 3 of the 2023 AI City Challenge.

4. Results and Discussion

4.1. Classification Results

We train three VideoMAEv2 models, each fitted to data for a specific camera location inside a vehicle. We perform both 5-fold cross validation and training over all fold data without validation. For hyperparameters, we use a base learning rate of 1e-3 and cosine decay scheduling that starts at 1e-8 and ends at 1e-6. Using the AdamW optimizer [29], we perform drastic weight decay of 0.05 and run warm-up for 5 epochs. We set the drop path to 0.1 and layer decay to 0.75. The input size of images is 224x224. Training for each model is run for 200 epochs in batch sizes of 6 with two RTX A5000 GPUs.

Fold	Dash (%)	Rear (%)	Right (%)
1	71.95	82.93	84.18
2	72.22	82.72	83.54
3	75.93	80.25	78.48
4	74.69	83.33	75.32
5	75.93	88.27	74.68
Average	74.14	83.5	79.24
Weight	0.31	0.35	0.34

Table 2. 5-Fold Cross Validation Results. The table describes the accuracy score of each fold across all camera views (dashboard, rearview, and right side window). The final average accuracy score is listed as well alongside the normalized accuracy scores, which are used as weights.

Based on 5-fold cross validation results from Tab. 2, we observe that the dashboard camera location is the hardest for VideoMAEv2 to learn over while the rearview camera data is the best. The right side window is somewhere between the two and is decently fitted to. We note that in fold 1 and 2, the right side window model outperforms the rearview model yet has a significant drop in performance in folds 3, 4, and 5. We reasonably conclude that the rearview model is the most reliable, as it has consistent accuracy above 80% across all folds. Moreover, the rearview camera even peaks as high as 88.27%, which is 4.13% higher than the next highest accuracy score among all folds and camera models. From the 5-fold cross validation results, we normalize the average accuracy over all folds for each camera model and apply those normalized values as weights for each camera in the multiview ensemble. Furthermore, for certain classes, such as 13 and 14, the rearview camera model alone provides the most clear and accurate predictions. Thus, for those classes, only probabilities generated from the rearview camera model are used. Similarly, for class 8, only results from the right side window model are used since this class, in particular, has a lot of noise which easily causes misclassification.

4.2. Temporal Action Localization Results

4.2.1. Evaluation Score. Our TAL implementation nets an evaluation score of 0.6010, which ranks 6th place on the public leaderboards for Track 3 of the 2023 AI City Challenge. Tab. 3 displays the top public leaderboard rankings.

Rank	Team ID	Score
1	209	0.7416
2	60	0.7041
3	283	0.6734
4	49	0.6723
5	118	0.6245
6	**279 (Ours)**	**0.6010**
7	8	0.5921
8	48	0.5907
9	83	0.5881
10	217	0.5426

Table 3. Public Leaderboard Rankings for A2 Dataset. Our method places 6th overall.

To compute the final score, the predicted action segment start and end times are compared against those of the ground truth annotations. The evaluation server calculates the ratio of overlap between the two. Given a ground truth action interval, it is matched to the closest, most similar predicted action interval. An overlap score is then calculated as the time intersection and union of the two action classes:

$$os(p, g) = \frac{max(min(ge, pe) - max(gs, ps), 0)}{max(ge, pe) - min(gs, ps)} \quad (4)$$

g represents the ground truth activity while p represents the predicted activity. gs and ge are the start and end times for the ground truth activity, and the same applies to ps and pe for the predicted activity. Though, ps and pe are given a tolerance in the range [gs – 10s, gs + 10s] and [ge – 10s, ge + 10s], respectively. Therefore, if either the start or end time of a predicted segment is outside the tolerance, the segment is considered unmatched. Once matching and overlap scores are computed, all remaining unmatched ground truth and unmatched predicted intervals receive an overlap score of 0. The final score is then taken as the average of all matched and unmatched overlap scores.

4.2.2. Ablation Studies. To understand how each component of our post-processing method impacts the final evaluation score, we present an ablation study in Tab. 4.

Threshold	Merge	Filter	Link	Score
✓	✓	✗	✗	0.2958
✓	✓	✓	✗	0.5627
✓	✓	✓	✓	**0.6010**

Table 4. Ablation Study on Post-Processing Components.

From Tab. 4, the threshold and merge steps alone are not sufficient for performing TAL. Without any further filtering of merged segments, the number of predicted segments becomes heavily saturated such that the system is guessing more so than carefully estimating. When adding the filtering step, the final overlap score increases significantly (+0.2669) and becomes more reasonable. The aforementioned score would place 9th overall in the rankings. To increase the score by a small margin (+0.0383), the linking step is crucial, as it refines temporal segments and thus causes less final action bounds to become unmatched. Note that during evaluation, all predicted interval start and end times must be within a 10 second tolerance of the groundtruth start and end times. Even if a predicted interval is generally correct but has a start time which is 12 seconds off from the groundtruth, that predicted interval is considered unmatched. The linking step alleviates such offsets in start and end times.

$r = 0.70$	$r = 0.75$	$r = 0.80$	Score
✓			0.5788
	✓		**0.6010**
		✓	0.5887

Table 5. Ablation Study on Relative Thresholding. r is a fraction of the max probability for a given set of prediction probabilities and determines the relative threshold.

For tuning relative thresholding, we perform an ablation study to test how different values of r affect the performance of the overall TAL system (thresholding, merging, filtering, and linking). We initially find values of r, such as 0.5 and 0.9 to be too extreme and cause over or under filtering. Values ranging from 0.7 to 0.8 tend to generate the best results, with 0.75 providing the highest performance compared to all values of r.

5. Conclusion

Taking advantage of the multiview aspect of Track 3 from the 2023 AI City Challenge, we propose a unique TAL framework that offers temporal boundary refinement via thresholding, merging, filtering, and linking steps. Our method generates temporal segment candidates which may contain the action of interest and then narrows down the candidate space by considering factors such as the duration and mean probability score of a segment. We compute a unique candidate score from these features and take the segment with the highest candidate score as the true-positive action interval. Subsequently, we apply linking which refines the temporal endpoints of the final selected action interval via multiple confidence level boundary analysis. Overlapping bounds from the analysis are processed via NMS with weighted interval fusion to produce the final refined action bounds. In conclusion, our method achieves top results, ranking 6th on the Track 3 public leader for the A2 dataset. We hope this paper provides insight into the development towards temporal action localization solutions for distracted driving behavior, and encourages more research in this area in the future.

References

[1] CDC. Distracted Driving | Transportation Safety | Injury Center | CDC. https://www.cdc.gov/transportationsafety/distracted_driving/index.html, retrieved June 2023.

[2] NVIDIA. AI City Challenge. https://www.aicitychallenge.org, retrieved May 2023.

[3] M. S. Rahman, J. Wang, S. V. Gursoy, D. Anastasiu, S. Wang, and A. Sharma, Synthetic Distracted Driving (SynDD2) dataset for analyzing distracted behaviors and various gaze zones of a driver, 2023.

[4] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev et al. The kinetics human action video dataset. arXiv preprint arXiv:1705.06950, 2017.

[5] A. Ulhaq, N. Akhtar, G. Pogrebna, A. Mian. Vision Transformers for Action Recognition: A Survey. arXiv preprint arXiv:2209.05700, 2022.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[7] L. Wang, B. Huang, Z. Zhao, Z. Tong, Y. He, Y. Wang, Y. Wang, and Y. Qiao. Videomae v2: Scaling video masked autoencoders with dual masking. In *European Conference on Computer Vision*, pages 14549-14560, 2023.

[8] Y. Li, C.-Y. Wu, H. Fan, K. Mangalam, B. Xiong, J., and C. Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4804–4814, 2022.

[9] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, pages 6201–6210, 2019.

[10] C. Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, pages 203–213, 2020.

[11] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1130–1139, 2018.

[12] G. Chen, Y.-D. Zheng, L. Wang, and T. Lu, DCAN: Improving Temporal Action Detection via Dual Context Aggregation. arXiv preprint arXiv:2112.03612, 2021.

[13] T. Lin, X. Liu, X. Li, E. Ding, and S. Wen. Bmn: Boundary-matching network for temporal action proposal generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3889–3898, 2019.

[14] H. D. Le, M. Q. Vu, M. T. Tran, and N. V. Phuc. Triplet temporal-based video recognition with multiview for temporal action localization. In *CVPR Workshop*, 2023.

[15] C. Nguyen, N. Nguyen, S. Huynh, and V. Nguyen. Learning generalized feature for temporal action detection: Application for natural driving action recognition challenge. In *CVPR Workshop*, 2022.

[16] C.-L. Zhang, J. Wu, and Y. Li. Actionformer: Localizing moments of actions with transformers. In Shai Avidan, Gabriel Brostow, Moustapha Cisse, Giovanni Maria ´ Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 492–510, Cham, 2022. Springer Nature Switzerland.

[17] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem. Daps: Deep action proposals for action understanding. In *European Conference on Computer Vision*, pages 768–784. Springer, 2016.

[18] R. Li, C. Wu, L. Li, Z. Shen, T. Xu, X. J. Wu, X. Li, J. Lu, and J. Kittler. Action probability calibration for efficient naturalistic driving action localization. In *CVPR Workshop*, 2023.

[19] Y. Ma, L. Yuan, A. Abdelraouf, K. Han, R. Gupta, Z. Li, and Z. Wang. M2DAR: Multi-view multi-scale driver action recognition with vision transformer. In *CVPR Workshop*, 2023.

[20] W. Zhou, Y. Qian, Z. Jie, and L. Ma. Multi view action recognition for distracted driver behavior localization. In *CVPR Workshop*, 2023.

[21] M. T. Tran, M. Q. Vu, N. D. Hoang, and K.-H. N. Bui. An effective temporal localization method with multi-view 3D action recognition for untrimmed naturalistic driving videos. In *CVPR Workshop*, 2022.

[22] Y. Liu, L. Wang, Y. Wang, X. Ma, and Y. Qiao. Fineaction: A fine-grained video dataset for temporal action localization. In *IEEE Transactions on Image Processing*, vol. 31, pp. 6937-6950, 2022.

[23] K. Li, Y. Wang, Y. He, Y. Li, Y. Wang, L. Wang, and Y. Qiao. Uniformerv2: Spatiotemporal learning by arming image vits with video uniformer. arXiv preprint arXiv:2211.09552, 2022.

[24] J. Liang, H. Zhu, E. Zhang, and J. Zhang. Stargazer: A transformer-based driver action detection system for intelligent transportation. In *CVPR Workshop*, 2022.

[25] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[26] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 6023-6032,* 2019.

[27] Z. Zhong, L. Zheng, G. Kang, S. Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020.

[28] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR Workshops*, 2020.

[29] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR,* 2019.

[30] Z. Tong, Y. Song, J. Wang, and L. Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. arXiv preprint arXiv:2203.12602, 2022.