WPI

Online Multi-View Naturalistic Driving Temporal Action Localization

A Major Qualifying Project Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the Degree of Bachelor of Science

Submitted by:
Jared Chan

Advised by:
Professor Ziming Zhang

Abstract

Distracted driving behavior is a major concern that claims the lives of many every year. To address the issue, AI City has presented the Track 3 Challenge, which aims to encourage research and development towards solutions that recognize and localize when distracted driving behavior occurs in time. We present a novel online, multi-view architecture that aggregates temporal context and uniquely samples past and present events to label and pinpoint the start and end temporal boundaries of distracted driving actions. Our algorithm is a two-stage Temporal Action Localization (TAL) method, which does not require a boundary detection network or any localization training. It consists of several stages to predict and refine temporal boundaries: aggregation, prediction, consolidation, post-localization processing, and assessment. Furthermore, our method achieves top results in the AI City 2023 Track 3 Challenge and performs highly in run-time efficiency. Our code is available at https://github.com/CarrotPeeler/WPI-Naturalistic-Driving-Action-Recognition-MQP.

Table of Contents

List of Tables

List of Figures

List of Acronyms

CNN: Convolutional Neural Network

DAPs: Deep Action Proposals

DCAN: Dual Context Aggregation Network

LSTM: Long Short-Term Memory Network

M2DAR: Multi-View Multi-Scale Driver Action Recognition

MViTv2: Multiscale Vision Transformer Version 2

MViTv2-B: Multiscale Vision Transformer Version 2 Base Model

RNN: Recurrent Neural Network

SynDD2: Synthetic Distracted Driving 2 Dataset

TAL: Temporal Action Localization

UCF101: University of Central Florida Dataset with 101 Actions

VideoMAE: Video Masked Autoencoders

X3D: Expand 3D

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]. As a means to identify and label such behaviors, the aim of this project is to develop a deep learning, video understanding based system that classifies and temporally localizes different distracted driving behaviors—a computer vision task known as Temporal Action Localization (TAL).

The foundation of this project is based on the *Naturalistic Driving Action Recognition* challenge adapted from the AI City organization [2]. In accordance with the challenge guidelines, the system developed needs to handle data in the form of video which captures a series of distracted driving actions. The data provided, SynDD2 [3], is synthetic and is collected from three different camera angles inside of a vehicle. The data is split into two datasets, A1 for training and A2 for validation, which total to 180 videos (27 hours in length). Additionally, each untrimmed video provided contains 16 unique actions according to their annotations, with a few 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 Kinetics [4] datasets. Some videos have the driver's face blocked with a hat or sunglasses while others do not. 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.

In particular, this project pays specific attention to researching and exploring unorthodox methods for developing TAL strategies for the challenge dataset. A common trend with previous strategies is to heavily focus on post-processing techniques to improve localization and classification. However, we have developed a more practical online solution, where incoming video data is continuously processed to perform ongoing localization. Thus, we minimize post-processing while still achieving top results. To do so, we explore and adapt existing strategies from past challenge submissions to suit our new method and propose additional unique concepts to design an overall architecture to achieve TAL.

1.2 Contributions

The main contributions of this paper are as follows:

- We propose a novel algorithm for Temporal Action Localization that offers online localization capability, yields a fast run-time performance of 160.5 FPS on two GPUs and 158.8 FPS on a single GPU, and performs highly when ranked against other methods in the AI City Challenge 2023 Track 3.

• Our algorithm involves a unique manner of aggregating and sampling temporal context and, in parallel, utilizes single proposal generation to fuse prediction probabilities across three different camera views representing temporal space and action events.

2. Literature Review

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. The subject is an important field of study that improves and contributes to real-world applications related to health monitoring, intelligent surveillance, virtual reality, human-computer interaction, and several others [6].

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 [7]. 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 [7]. 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].

One particular implementation is Multiscale Vision Transformers, which scale the input video resolution to different sizes and fuse feature analysis performed over each stage to capture low and high-level visual details [8]. There currently exist two versions of Multiscale Vision Transformers—with the second version, MViTv2 [9], performing significantly well on action recognition tasks. MViTv2 has shown to outperform other impressive models designed for action recognition tasks, such as SlowFast [10] and X3D-XL [11], on the Kinetics 400, 600, and 700 datasets.

2.3 Visual Prompting

Visual Prompting is a recent topic of study concerned with eliciting improved model understanding by leveraging external information or visual cues applied over the input image or video data. Several studies have proven Visual Prompting to be an efficient alternative to full fine-tuning (where all parameters participate in training) by reducing the number of trainable parameters necessary for a large-scale pre-trained model to learn from small datasets [12, 13, 14]. The aforementioned methods accomplish this by injecting additional learnable parameters directly into the input space (images or videos), which enables the input to conform to a representation that the model would best understand. Therefore, a large-scale model only needs to update a small group of its parameters rather than all of them to learn from small datasets, reducing the computational resources required for training.

2.4 Past Methods

To solve the problem posed by AI City's third track challenge, all past methods and winners have developed Temporal Action Localization (TAL) systems. TAL is another field under action recognition which not only deals with classifying different categories of actions but extracting temporal information as well.

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 [15, 16, 17].

Among past top performers [18, 19] in the AI City challenge, the only effective one-stage method used has been ActionFormer [20], 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 [18] 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 [16] opts to aggregate boundary and proposal level contexts to generate high-quality candidate intervals where actions might occur. Another method is DAPs [21], which aggregates temporal context (video frames) across a sliding window to generate variable length proposals of different temporal scales. However, top performers [22] 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 [22, 23, 24, 25] 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 (fixed) windows, their solutions favored exploring ways to improve classification and post-processing techniques rather than proposal generation. MViTv2 and X3D are commonly employed as the backbone for classification.

Since previous works related to the challenge are concerned with achieving high accuracy, they do not explore online methods that omit the use of future temporal context and extensive post-processing to gain near real-time capability. Works, such as [26], adhere to online TAL execution via applying anchor windows for proposal generation, while other architectures [27] impose frame-by-frame processing to achieve real-time, frame-sensitive localization.

3. Methodology

3.1 Codebase

Our project utilizes PySlowFast, a light-weight codebase providing resources and tools to achieve efficient, high-performance results and fast implementation for video understanding and related novel research ideas. PySlowFast offers implementation for several state-of-the-art classification model backbones, including SlowFast, MViT, X3D, as well as others, and also supports a variety of different tasks, such as classification and detection [28]. Additionally, PySlowFast contains a plethora of pre-trained checkpoints for datasets like Kinetics 400 and 600, which enables training and research carried out on smaller datasets to be quick and efficient. Most notably, the codebase supplies implementation for preprocessing and loading data. In terms of video preprocessing, PySlowFast provides decoding for retrieving video frames and temporal sampling for uniformly selecting subsets of frames. Furthermore, for data augmentation, random spatial cropping and several other methods are offered as well. Overall, PySlowFast is easy to use and ensures code reusability.

3.2 Classification

As part of the AI City challenge, they provide two datasets, A1 and A2, which total to 180 videos (27 hours in length). However, A2 must be used for validation, only, leaving A1 as the only dataset usable for training. Thus, only 150 videos (22.5 hours in length) are available for training, resulting in a rather small dataset which poses a concern for overfitting.

3.2.1 Model Selection and Preprocessing

Based on the success of past submissions for the challenge [18, 23, 29], we use state-of-the-art MViTv2-B as the backbone model for classification. PySlowFast provides a checkpoint for MViTv2-B pre-trained on Kinetics 400, and as mentioned previously, MViTv2

has shown to outperform other state-of-the-art models on Kinetics 400. For training, clips containing action segments are parsed from untrimmed videos according to annotation files with action start and end timestamps. Clips are then decoded and broken down into their individual frames. After applying uniform temporal sampling using a rate of 4, each clip is reduced to 16 frames. Random spatial cropping is then applied to downscale frame resolutions to 224x224 and minimize overfitting. Frames are lastly color normalized to reduce the effects of variable environmental lighting on model understanding. The figure below displays the results of preprocessing after applying the aforementioned techniques.

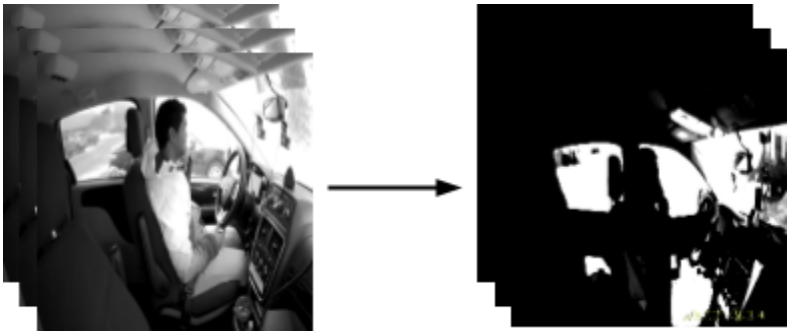

Figure 1. Preprocessing. The left image shows how video clips appear before preprocessing, while the right image displays the after results.

3.2.2 Minimizing Overfitting and Improving Generalization

Originally, we explored research related to Visual Prompting as an attempt to improve model generalization and perhaps reduce overfitting. To do so, we focused on a specific area of Visual Prompting, which involves the use of Adversarial Reprogramming [30] to benefit model understanding by placing pixel perturbations in input images. Based on one study [12] which designs pixel perturbations according to categorical classification loss, they conclude that pixel padding patterns are the most effective for influencing model understanding. Therefore, we designed several similar perturbation padding techniques to test for improvement in model understanding. These designs were created via experimentation to test if different designs had different efficacy on model generalization. Because we train the classification model on all of the data available and do not create separate training splits according to camera view, our designs focus on assisting the model in understanding action categories from different camera perspectives. Below are several examples of prompt designs developed.

Figure 2. Multi-View Padding. We apply randomly initialized perturbations to input frames representing action segments as a means of prompting. When perturbations are updated during training, updates and changes in perturbation design vary according to the camera view the action is recorded from. Thus, there are only three visual prompts applied across the entire training dataset for this prompt design.

Figure 3. Multi-View Variable Padding. Similar to the last design, three prompts are applied across the entire dataset, one prompt for each camera view. This design, however, uses different padding sizes for each side of the image.

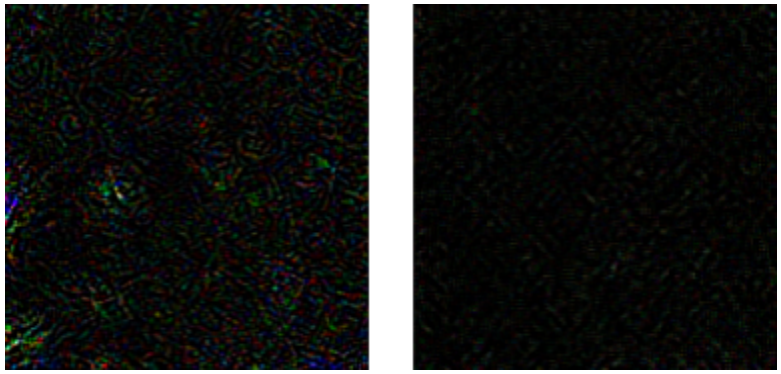

Figure 4. Multi-View Boundary Fixed-Patch. Instead of applying the same prompt over all frames in an action segment, prompts entirely covered in pixel perturbations replace the first and last frames in a 16 frame action sequence. The left image is the prompt replacing the first frame while the right image replaces the last frame.

For training, both the visual prompts and the classification model were kept unfrozen at the same time. By training prompt parameters simultaneously, the goal was to prevent the classification model from memorizing patterns in the prompt perturbations. Conversely, to also test if freezing prompt parameters would assist training, we applied a selective updating technique to only update prompt perturbations when validation accuracy for classification stopped improving. Ultimately, visual prompt designs produced inconclusive results concerning the improvement of model understanding.

Moving on from visual prompting techniques, we shifted focus towards a few state-of-the-art techniques that are well known for reducing overfitting and boosting generalization—Mixup [31] and CutMix [32] which create new synthetic samples from existing data via blending images and labels, or cutting and pasting image patches together. On top of the aforementioned techniques, we also utilize other random data augmentation techniques provided by PySlowFast. Some notable methods include Random Erasing [33], which randomly erases a rectangular portion of an image and its pixels, and RandAugment [34], which performs random image distortions automatically using a multitude of hyperparameters. By using these four data augmentation techniques in conjunction, we generated a checkpoint after 200 epochs for MViTv2-B, and for further testing, a 400 epoch checkpoint.

3.3 Temporal Action Localization

To accomplish TAL, we propose the following architecture which continuously updates and refines temporal boundaries when localizing a single action instance. After performing inference over a single, approximately 2 second clip at 30 FPS (64 frame sliding window), localization can immediately begin without waiting for all action proposals within an untrimmed

video to be classified and processed. Thus, by relying on an incoming stream of 64 frame preprocessed action proposals, our method achieves online localization capability.

3.3.1 Architecture Overview

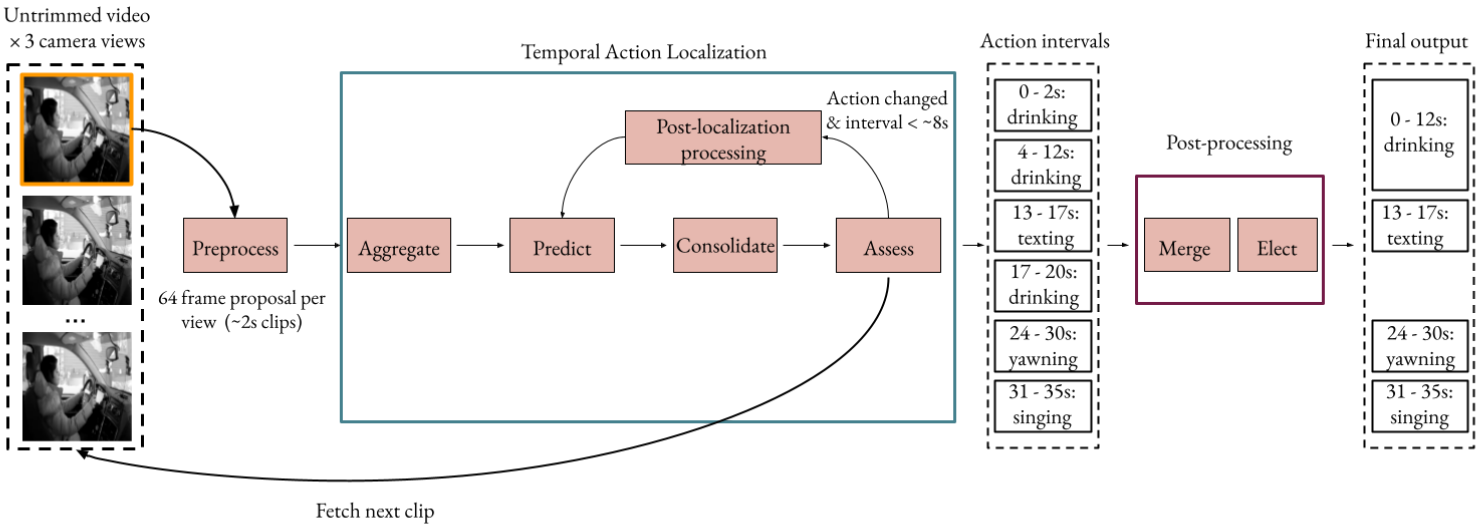

Figure 5. Proposed TAL Architecture. The diagram breaks down the sequence of events that occur to produce action localization results.

First, each untrimmed video is segmented into 64 frame action proposals. Because the AI City track challenge dataset includes three different camera views for each unique set of video footage, the untrimmed videos are then organized such that batches of three are formed, where each batch has three videos representing the same footage from three different angles. Thus, a multi-view ensemble is produced.

The goal of TAL is to detect when changes in actions occur. Because actions can occur within intervals as small as approximately 2 seconds, past and present solutions have used 2 second proposal segments as the base temporal resolution for analysis to help detect short occurring actions. However, the AI City dataset also contains actions as long as 20 to 30 seconds, which presents another challenge. By reducing the temporal length for analysis to roughly 2 seconds, the amount of context and motion provided by the driver is also reduced, limiting the information available to the model for classification. To localize long actions using the aforementioned method, a series of short 2 second action segments must be classified and then further processed to identify the series as a single long action interval. However, as mentioned, when single short action segments are classified without more context, the model is prone to misclassification.

For example, eating and talking are often confused for one another. For both, context is provided when the driver begins to move their mouth. Yet, to reliably distinguish eating from talking, the model needs to see that the driver is reaching for or putting food in their mouth

beforehand. When individually analyzing a series of 2 second clips, events are assumed to be unrelated, preventing the carry over of context through sequential clips.

3.3.2 Aggregation

Our method opts to provide full context for action classification by leveraging information from all past clips involving the same action in addition to the newest clip. Therefore, until a new action differing from the previously classified action is detected, new incoming proposal segments are assumed to contain the same action and, as such, are added to an ongoing aggregation pool. If an action interval is a puzzle to be solved, the aggregation pool acts as the puzzle board, where pieces (temporal clips) are placed in order to create the full picture. By aggregating past and present clips containing the same action, temporal resolution increases, providing the model with more action context to base its prediction probabilities from and improving prediction reliability.

As opposed to popular TAL frameworks which strengthen the generalization of an action instance over a specific temporal interval via the redundancy of overlapping proposals, our method instead exploits information from past temporal clips to provide more context for newer intervals being classified.

Below is a figure illustrating the frame sampling strategy employed for exploiting information from past and present clips.

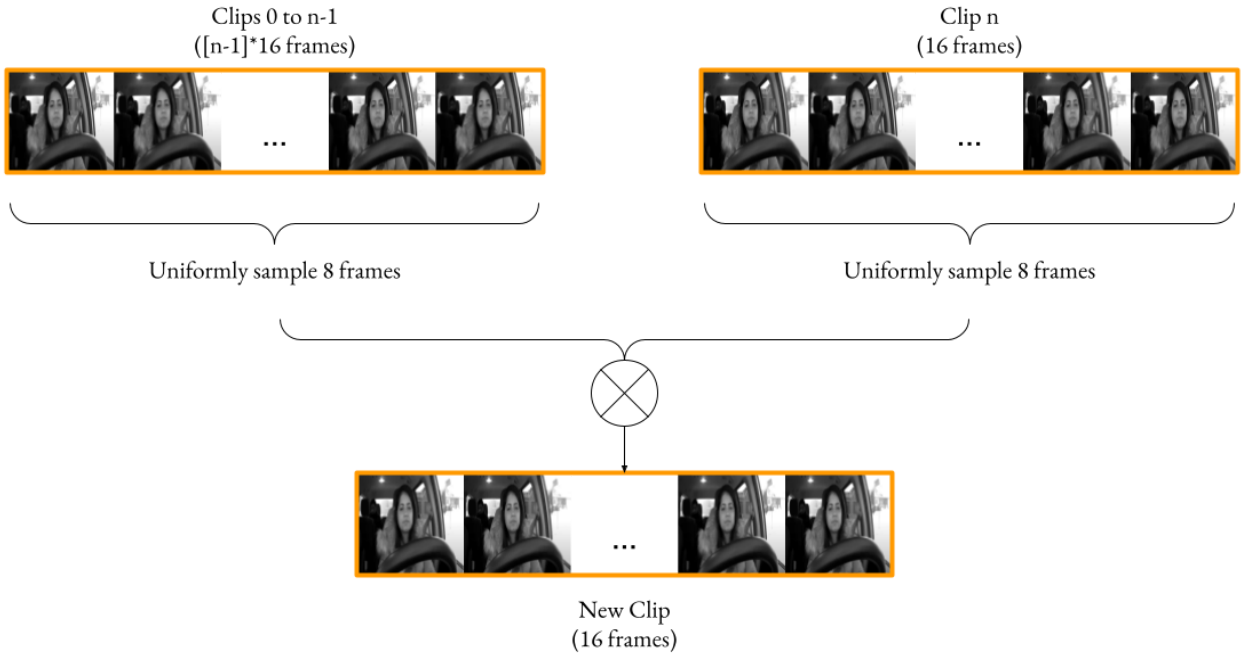

Figure 6. Aggregated Sampling Strategy. The top left image represents all frames from past clips within the action interval, while the top right image displays frames from the newly added clip. The fusion of frames results in the bottom image.

As described by the figure, all frames from past clips, each being 16 frames long, are collected and uniformly sampled to select 8 frames that best represent the action occurring within the ongoing localized interval. Similarly, 8 frames are also uniformly sampled from the newly introduced proposal/clip, and then both sets of 8 frames are concatenated to produce a fused 16 frame clip. By equally sampling the same number of frames from past clips and the new clip, it ensures the model will be confused and will generate low confidence predictions about the previously classified action, indicating the new clip may contain a different action than the previous clips. Otherwise, for highly confident predictions about the previously classified action, it is assumed the new clip contains the same action.

3.3.3 Prediction

Because our method implements aspects of the sliding window technique for proposal generation, it encompasses the same weakness of having localized actions be constrained to a 'window' of size equal to approximately 2 seconds or 64 frames at 30 FPS. Thus, the assumption is that when an action is detected in this window of time, the action occurs throughout the entire time span of the window, which is not always the truth. As mentioned previously, we uniformly sample a combination of frames from both past and present. However, if this sampling strategy results in a new action detected with a probability below a certain threshold, we opt to redesign the aggregate clip such that the 8 frames from the new clip are not uniformly sampled but instead chosen from among the last half of frames. Thus, if this resampled aggregate clip produces better probabilities for the new action detected, the new assumption is that the action starts halfway into the new clip's time interval. The aforementioned resampling step helps to further improve the precision of localized endpoints (start and end times) for detected actions.

While aggregation has merit in strengthening classification confidence over medium to long temporal intervals, the downside is the loss of sensitivity in detecting actions occurring in small intervals (around 8s or less). Thus, in addition to sampling from an aggregated group of past and present clips, a single clip representing the current 2 second temporal interval alone is also input into the classification model to compensate for aggregation's weakness in detecting brief changes in action. Below illustrates the combined process.

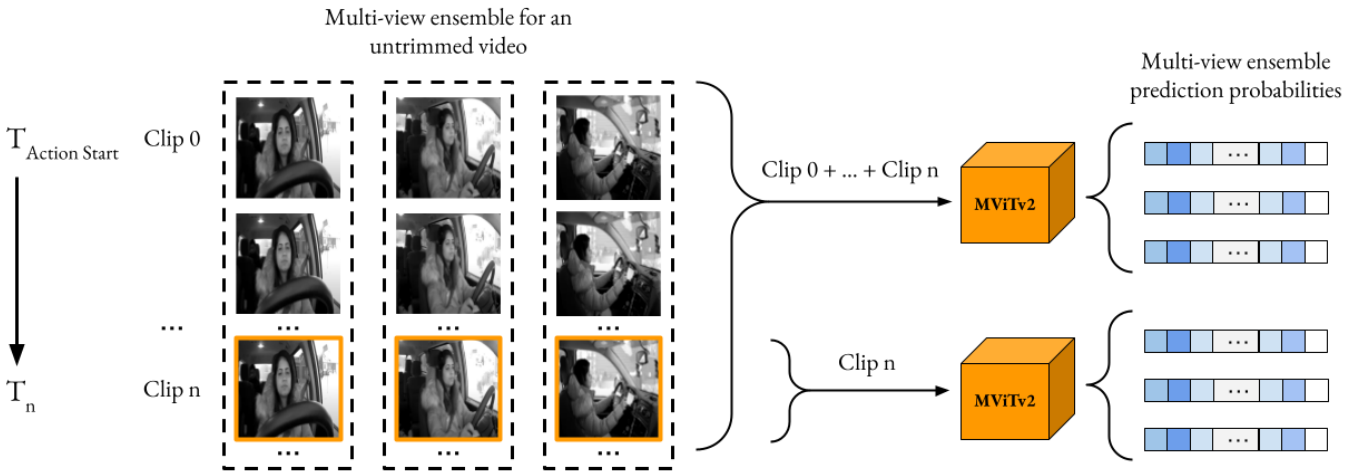

Figure 7. TAL Prediction Step. A multi-view ensemble composed of clips from three different camera views is used to produce two sets of inputs for classification. Clips used in the ensemble are selected starting from the first temporal interval where the current localized action first appears.

Aggregation combined with single proposal analysis smoothes prediction probabilities across several intervals for localizing a single action—reducing noisy, false positive predictions that might otherwise occur with overlapping proposals. Moreover, the increased reliability of predictions resulting from aggregation allows for temporal intervals to be merged continuously throughout localization rather than in post-processing. Therefore, very little post-processing is required to finalize action interval results compared to overlapping proposal methods which require extensive post-processing to eliminate noise and merge intervals.

3.3.4 Consolidation

Once the aggregated clips and single clip for each camera view are input into the classification model to generate prediction probabilities, the aggregated and single clip probabilities are processed separately by the consolidation step of TAL. Based on the 5th place solution, Purdue's M2DAR [23], we consolidate and fuse prediction probabilities across different camera views by applying weights to each matrix and then summing them to compute a weighted average. The highest probability from the final computed matrix is taken as the predicted class. The underlying reason for applying separate weights is due to some camera views being stronger than others in clearly observing an action. For example, when the driver is adjusting the control panel, it is more clearly observed from the right side window view of the car than from the dashboard view.

3.3.5 Assessment

Once final predictions are formed for both the aggregated group of clips and the single clip, the last step is to determine whether both predictions agree or not on the type of action occurring within the current temporal interval. If they agree, it is assumed the agreed upon action is a true positive and the above steps repeat for the next 2 second interval; otherwise, if they disagree, it is assumed the current action being localized has ended and a new action has started. If so, the start time of the localized action is output as the start time of the first aggregated clip while the end time is output as the start time of the current interval where predictions disagree. The final step of localization is to calculate and output the Gaussian weighted mean of all probabilities obtained throughout the localized interval. The recorded probability is used later in the election step to perform the final filtering process before the submission of results to the evaluation server.

3.3.6 Post-Localization Processing

While our method eliminates the *excessive* use of overlapping proposals, it still does utilize them but on a much less frequent scale. Even when using aggregation alongside a single proposal from the current interval, action classification can still produce small amounts of false positives and misclassifications. The aforementioned problem only occurs within short intervals (less than 8 seconds) due to low temporal resolution. To address the issue, post-localization processing is immediately performed after localizing a single action interval less than 8 seconds in length. Re-evaluation ensures predictions within small intervals are reliable, and if not, corrects or rejects them.

By using the aggregated group of clips, past frames are sampled to generate overlapping proposals. Specifically, a 16 frame proposal is produced every 4 frames. After the classification model calculates prediction probabilities for each proposal, a Gaussian weighted average is applied over all probability matrices. The assumption for using Gaussian distribution is that proposals closer to the midway point of the interval are more likely to better represent the action occurring throughout the entire interval than the interval endpoints. Furthermore, based on observation, proposals closer to the start and end of the temporal interval may include actions belonging to a different action sequence. After computing the final probability matrix using Gaussian weights, and therefore deriving the final prediction for the interval re-evaluation, a filtering threshold is applied. Each action class is thresholded differently, specifically due to the bias in confidence of the classification process for each class. To obtain the threshold value for an action class, we find the min prediction probability among true positive intervals and compare it to the max prediction probability among false positives. Then, we determine a threshold based on the max that prevents most false positive probabilities from passing through but is still below the min for true positives. Lastly, compared to the prediction step of TAL, which utilizes an MViTv2 trained for 200 epochs, the re-evaluation process uses a 400 epoch trained MViTv2 in which the model is more confident in its incorrect predictions but overall has higher accuracy.

3.3.7 Post-Processing

The final step of TAL is to perform optional post-processing, which prepares an organized submission file specific to the AI City challenge, containing all action interval timestamps with their respective video ID. The two major reasons for the merge and elect steps are to combine remaining unmerged action intervals and to filter out duplicate action instances.

Although aggregation serves to uniquely merge action segments continuously during localization, cases exist where the driver pauses in the middle of a long action sequence, causing the TAL algorithm to detect a different action. Action intervals broken up by this edge case appear consecutively in the localization results but need another merge to help the evaluation server identify these segments as one interval.

Following the final merge of intervals, the election algorithm, adapted from [23], selects one localized interval per action class to remain for each untrimmed video. Although untrimmed videos may have more than one instance of an action class, the annotators for the test dataset have chosen to only annotate 16 of the most obvious and apparent action intervals that represent each action class the best. We have repurposed the election algorithm to uniquely score intervals not purely based on the Gaussian probability for the entire interval but on the duration of the interval as well. In contrast, if the election algorithm is not used, true positive duplicate actions will not be filtered out, allowing the TAL algorithm to be flexible and usable for different purposes and problems.

4. Results and Discussion

4.1 Classification Results

To achieve the best classification performance, we experimented with different training configurations for MViTv2. Our first approaches involved Visual Prompting in which we added pixel perturbation padding to the input action segments for training. Among our developed designs, only multi-view padding yielded meaningful results. The figure below represents how multi-view padding compares with other prompted and unprompted methods.

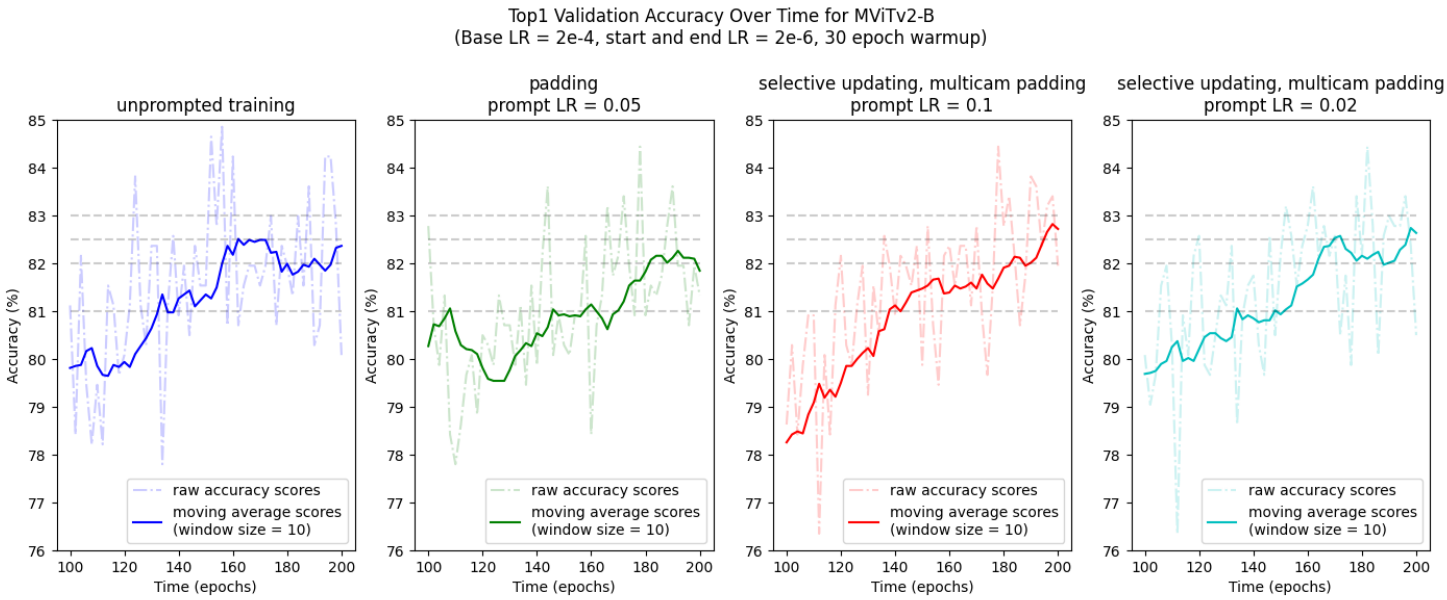

Figure 8. Comparison Between Prompted and Unprompted Methods. The leftmost two graphs show validation accuracy results for unprompted training and normal padding without selective updating. The other two figures display results for multi-view padding with selective updating across different prompt learning rates (0.1 and 0.02). A moving average has been applied to all graphs to better visualize results.

Overall, while the moving average may indicate a slight improvement in accuracy as much as 0.5% with multi-view padding and selective updating techniques, there is not enough consistent evidence nor significant increase in accuracy to conclude such methods will generally improve classification training. However, this is not to say Visual Prompting of the adversarial type will not improve generalization. Due to lack of time, our experiments are simply not extensive enough to determine the effect of such methods. We hope that future works will explore Visual Prompting as a means to improve model understanding of different actions performed from different camera angles.

We shifted focus to well known, state-of-the-art methods for improving generalization, instead, and found Mixup, CutMix, and other data augmentation techniques to improve the fitting of MViTv2 over the challenge dataset. Applying the aforementioned techniques to the training data, we obtained two checkpoints for MViTv2, one after 200 epochs and another after 400 epochs of training. For hyper parameters, we use a base learning rate of 5e-4 and cosine scheduling that starts and ends at 1e-6. Using the AdamW optimizer [35], we perform drastic weight decay of 0.05 and run warm-up for 35 epochs. Training is run in batch sizes of 8 with two RTX A5000 GPUs. To compare the pros and cons of these two checkpoints against the 200 epoch checkpoint obtained without data augmentation, we measure several statistics, represented in Table 1.

	Top-1 Val Accuracy (%)	Mean (Incorrect)	Mean (Correct)	Number of incorrect ≥ 0.9
200 epochs (no augment)	80.91	0.851	0.977	56
200 epochs (with augment)	85.48	0.647	0.859	6
400 epochs (with augment)	87.97	0.690	0.864	10

Table 1. Augment vs. No Augment Checkpoint Comparison. We measure statistics for each checkpoint, such as top-1 validation accuracy, the mean probability among both correct and incorrect predictions, and the number of incorrect prediction probabilities greater than or equal to 0.9.

Based on the above comparison, we evaluated the 200 epoch checkpoint with data augmentation to be the best for classification. Because the classification model, MViTv2, will be analyzing long untrimmed videos that contain only 16 annotated true positive actions throughout, the model's confidence in making incorrect predictions should be relatively low compared to correct predictions, which should be very high. The 200 epoch checkpoint with data augmentation follows this criteria better than the other listed checkpoints, having the largest difference in mean probability of 0.212 between correct and incorrect predictions. Thus, having a larger correct-to-incorrect prediction probability ratio will improve the effectiveness of filtering false positive predictions. Furthermore, we evaluate the checkpoint without augmentation to be rather overconfident in all predictions due to a significant portion of incorrect prediction probabilities being greater or equal to 0.9.

Aside from prediction probability analysis, top-1 validation accuracy also plays an important role. While we elect the 200 epoch checkpoint with augmentation to be used primarily for classification, we still utilize the 400 epoch augmentation checkpoint in post-localization processing, where confidence is not as important as the identification of the correct class.

4.2 Temporal Action Localization Results

4.2.1 Evaluation Score

Our TAL implementation nets an evaluation score of 0.5711, which ranks 9th place on the public leaderboards for the AI City third track challenge. Table 2 displays the top public leaderboard rankings.

Rank	Team ID	Team Name	Score
1	209	Meituan-IoTCV	0.7416
2	60	JNU_boat	0.7041
3	283	UIT-HCM14	0.6734
4	49	ctc-AI	0.6723
5	118	RW	0.6245
6	8	Purdue Digital Twin Lab	0.5921
7	48	BUPT-MCPRL	0.5907
8	83	DiveDeeper	0.5881
9	**279**	**WPI_Envy**	**0.5711**
10	217	INTELLI_LAB	0.5426

Table 2. Top Teams on the Public Leaderboard as of August, 2023. Envy (our team) places 9th overall.

To compute the final score, the submitted action segment annotations are compared against the ground truth annotations to calculate 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:

$$os(p, g) = \frac{max(min(ge, pe) - max(gs, ps), 0)}{max(ge, pe) - min(gs, ps)} \quad (1)$$

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*. Though, *ps* and *pe* are given a tolerance in the range [*gs* – 10s, *gs* + 10s] and [ge – 10s, ge + 10s], respectively. Once matching and overlap scores are computed, all remaining unmatched ground truth and 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 TAL method impacts the final evaluation score, we performed an ablation study. During early development, we experimented with two different systems for consolidating predictions from each camera view into a single prediction. Our first implementation opts to analyze predictions from each camera view at face value,

without any weights applied to different camera views. It simply checks if there is a common predicted action class among the three camera views, and if so, takes that action class to be the best representation of all three predictions. Then, the mean probability among common predictions is computed and is compared against the threshold, 0.790. The threshold is based on the mean correct prediction probability from Table 1 and is slightly lower to reduce harsh filtering. If the probability does not pass the threshold or no common predictions exist, the action interval is not used in final results. We also experiment with fixing the amount of clips that can be aggregated and therefore sampled at once. In general, we find that weighted consolidation outperforms its unweighted counterpart. To add, removing limits on aggregation capacity further improves the evaluation score as well. The post-localization processing and election steps have the largest impact on evaluation score, each contributing an approximately 6% increase when applied to the TAL architecture.

AGG	AGG-T	CONS-U	CONS-W	RES	POS-LOC	ELEC	Score
✘	✔	✔	✘	✘	✘	✘	0.3594
✘	✔	✘	✔	✘	✘	✘	0.4082
✘	✔	✘	✔	✔	✘	✘	0.4418
✔	✘	✘	✔	✔	✘	✘	0.4529
✔	✘	✘	✔	✔	✔	✘	0.5137
✔	✘	✘	✔	✔	✔	✔	0.5711

Table 3. Ablation Study on TAL Components. AGG is aggregation without thresholding, whereas AGG-T is aggregation with thresholding. U refers to unweighted while W is for weighted implementation for consolidation. RES is resampling, POS-LOC refers to post-localization processing, and ELEC is election.

4.2.3 Past Methods

While the final score for our proposed TAL architecture is still well below the top two winners' scores of 0.7416 and 0.7041, our method still achieves 9th place performance and offers unique online temporal action localization as well. Furthermore, our method presents flexibility in terms of how it can be applied. When working with datasets that do not involve duplicate action segments within an untrimmed video, such as the AI City challenge, our method offers a post-processing election stage to remove such duplicates. Otherwise, for datasets where multiple instances of action classes occur per video, post-processing steps like election can be removed to enable full online processing.

Because our TAL method focuses on utilizing multi-view data and is therefore not compatible with common datasets for online and real-time benchmarking, we opt to provide a cross-comparison in runtime-efficiency to draw conclusions against other methods. Since the frame rate of the captured video data largely impacts the inference speed, we choose to only

compare our work against others that perform benchmark tests using datasets with 25-30 FPS videos.

Method	Dataset	Avg. Video Length (mins)	Video FPS	Inference FPS
D. Zhang *et al.* [36]*°	UCF101-24 [38]	-	25	37.8
R. Hou *et al.* [37]*°	THUMOS'14 [39]	3	30	40
KORSAL [27]*°	UCF101-24	-	25	41.8
Y.H. Kim *et al.* [26]°	THUMOS'14	3	30	70.5
DAPs [21] †	THUMOS'14	3	30	134.1
Ours (1 GPU)°	SynDD2 (A2) [3]	8.415	30	**158.8**
Ours (2 GPUs)°	SynDD2 (A2)	8.415	30	**160.5**

° Online * Real-time † Offline - Not available

Table 4. Run-time Comparison.

Compared to other real-time methods which have kindly provided run-time measurements, our TAL implementation outperforms them, achieving a high 160.5 FPS with two GPUs and 158.8 FPS with a single GPU. Thus, for a more efficient, less resource intensive approach, our method offers a single GPU option. Similarly, our method also outperforms other works that impose temporal context aggregation as well but are not real-time, such as DAPs. The underlying reason is mainly due to our method being a training-free algorithm rather than a neural network. As mentioned previously, the low volume of training samples in the AI City dataset results in boundary detector networks being unreliable.

While our method provides an inference speed advantage against other online frameworks, the max latency is still not low enough to impact real-time applications.

4.3 Future Improvements

The largest bottleneck of our method is the classification model rather than the manner of localization. By comparing our final localized submission results against the top two winner's results, we found our classification model frequently failed to detect true positive intervals where class 1 and 13 occurred. Similarly, mistaking class 12 (talking to passenger, backseat) for 11 (talking to passenger, right) was also common. Correcting these mistakes would increase our final score by a significant percentage. As non-matched ground truth actions contribute a score of 0 to the final averaged score, including intervals for classes 1 and 13 where they are missing would prevent such drops in score.

Most notably, the top two methods eliminate these issues by utilizing other models such as VideoMAE [40] and X3D for classification. While X3D and MViTv2 are standard approaches

used for classification on the challenge dataset, VideoMAE is a new solution which is especially interesting due its first place performance [24] in the 2023 AI City challenge. Aside from implementing VideoMAE, another improvement to consider is converting the current TAL algorithm into a boundary detection network. Already mentioned works, such as DAPS, have shown that LSTMs are beneficial for aggregating and compactly storing temporal context (frames) as feature vectors. This opens up the possibility for improving the efficiency of the aggregation step in our proposed method. However, G. Chen *et al.* suggest that 1D convolutions applied over the temporal dimension of videos produce better performance than LSTMs [16]. Furthermore, the aforementioned authors' research on DCAN shows that self-attention mechanisms ignore order and distance and only pay attention to the correlation between two action segment positions [16]. Thus, attention mechanisms are not reliable for boundary detection.

Granted more time, we would also further experiment with Visual Prompting to mitigate misclassifications between similar classes (11 and 12) and strengthen learning over datasets which include actions performed from various camera perspectives. Lastly, enabling the method to process in real-time via removing the sliding window technique would also increase the efficiency of the architecture, as it would eliminate the two second max latency introduced by the window. To conclude, future improvements to our method would include usage of VideoMAE for classification, involving 1D convolutions for aggregating long-range temporal context, and incorporation of Visual Prompting and real-time processing to improve robustness.

5. Conclusion

Taking advantage of the unique multi-view aspect of the Track 3 Challenge from AI City, we propose a new TAL architecture for generating online localization results. Our method aggregates and predicts over past and present temporal context to boost temporal resolution and classification reliability. In parallel, single proposal generation is used to solely analyze present temporal context to strengthen prediction results over the present and compare them with the past. We then fuse predictions across a multi-view ensemble for each temporal interval via a consolidation step, check for changes in action via an assessment step, and finally correct or reject short action segments using post-localization processing. To prepare challenge specific results, localization results are merged and further filtered in optional post-processing. We conclude that our method achieves top results and run-time performance among other contemporary frameworks but still has potential for improvement in certain areas, especially with regards to the classification model. We hope this paper provides insight into the development towards online and real-time solutions for recognizing and localizing distracted driving behavior, and encourages more research into 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] Z. Shuchang, A Survey on Human Action Recognition. 2022.

[7] 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.

[8] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, pages 6824–6835, 2021.

[9] 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.

[10] 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.

[11] 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.

[12] H. Bahng, A. Jahanian, S. Sankaranarayanan, and P. Isola. Exploring visual prompts for adapting large-scale models. arXiv preprint arXiv:2203.17274, 2022.

[13] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. J. Belongie, B. Hariharan, and S.-N. Lim. Visual prompt tuning. In *ECCV*, volume 13693, pages 709–727, 2022.

[14] S. Chen, C. Ge, Z. Tong, J. Wang, Y. Song, J. Wang, and P. Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In *NeurIPS*, 2022.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] Y. H. Kim, H. Kang, and S. J. Kim. A sliding window scheme for online temporal action localization. In *European Conference on Computer Vision*, pages 653–669. Springer, 2022.

[27] K. Abeywardena, S. Sumanthiran, S. Jayasundara, S. Karunasena, R. Rodrigo, and P. Jayasekara, KORSAL: Key-point Detection based Online Real-Time Spatio-Temporal Action Localization. arXiv preprint arXiv:2111.03319, 2021.

[28] H. Fan, Y. Li, B. Xiong, W.-Y. Lo, and C. Feichtenhofer. PySlowFast. https://github.com/facebookresearch/slowfast, 2020.

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

[30] G. F. Elsayed, I. Goodfellow, and J. Sohl-Dickstein. Adversarial reprogramming of neural networks. arXiv preprint arXiv:1806.11146, 2018.

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

[32] 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.

[33] 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.

[34] 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.

[35] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR,* 2019.

[36] D. Zhang, L. He, Z. Tu, S. Zhang, F. Han, and B. Yang. Learning motion representation for real-time spatio-temporal action localization. In *Pattern Recognition*, 103:107312, 2020.

[37] R. Hou, R. Sukthankar, and M. Shah. Real-time temporal action localization in untrimmed videos by subaction discovery. In *BMVC*, 2017.

[38] K. Soomro, A. Roshan Zamir, and M. Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. In *CRCV-TR-12-01*, 2012.

[39] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. 2014. THUMOS Challenge: Action Recognition with a Large Number of Classes. http://crcv.ucf.edu/THUMOS14, 2014.

[40] 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.