Efficient Naturalistic Driving Action Localization

Rongchang Li¹, Cong Wu¹,⁵, Linze Li¹, Zhongwei Shen², Tianyang Xu¹, Xiao-jun Wu¹, Xi Li³, Jiwen Lu¹, Josef Kittler⁵
¹Jiangnan University. ²Suzhou University of Science and Technology. ³Zhejiang University. ⁴Tsinghua University. ⁵University of Surrey.
{li_rongchang, congwu, linze.li}@stu.jiangnan.edu.cn, shenzw@usts.edu.cn
{tianyang.xu, wu_xiaojun}@jiangnan.edu.cn
xilizju@zju.edu.cn, lujiwen@tsinghua.edu.cn, j.kittler@surrey.ac.uk

Abstract

The task of naturalistic driving action localization carries significant safety implications, as it involves detecting and identifying possible distracting driving behaviors in untrimmed videos. Previous studies have demonstrated that action localization using a local snippet followed by probability-based post-processing, without any training cost or redundant structure, can outperform existing learning-based paradigms. However, the action probability is computed at the snippet-level, the input information near the boundaries is attenuated, and the snippet size is limited, which does not support the generation of more precise action boundaries. To tackle these challenges, we introduce an action probability calibration module that expands snippet-level action probability to the frame-level, based on a preset snippet position reliability, without incurring additional costs for probability prediction. The frame-level action probability and reliability enable the use of various snippet sizes and equal treatment for information of different temporal points. Additionally, based on the calibrated probability, we further design a category-customized filtering mechanism to eliminate the redundant action candidates. Our method ranks 2nd on the public leaderboard, and the code is available at https://github.com/RongchangLi/AICity2023_DrivingAction.

1. Introduction

Abnormal driving conditions pose serious safety hazards, involving facial movements such as yawning, or full-body movements such as texting. The naturalistic driving action localization requires the detection of distracted driving actions of the driver, which has the potential to make driving safer by alerting drivers or autonomous vehicles to potential hazards and reducing the number of accidents caused by human error. The AI City Challenge dataset collects driving data using three cameras from different perspectives inside the vehicle, with the objective of accurately identifying distracted actions and localizing their start and end times. This task has significant applications in enhancing traffic safety and developing smart cities.

The previous top-performing solutions [15, 28] for naturalistic driving action localization involved a multi-stage process. First, the video was segmented into smaller snippets, and then an action recognition model was used to predict the action class probability. Finally, a training-free post-processing strategy was employed to convert the snippet-level class probability sequence into the localization results. However, the method of generating localization results based on snippet-level class probabilities has the following issues: (1) The finest temporal resolution is limited by the snippet size, which may decrease the reliability of the predicted class probabilities; (2) Assigning the same class probability score to every temporal position within a snippet is not reasonable for positions near the snippet boundaries.

To address these issues, we propose an action probability calibration method. Our calibration method generates frame-level action probability scores from various snippet-level results with preset reliability, without introducing additional computations. Furthermore, we utilize prior knowledge from the multiple camera views to calibrate the action scores. With the proposed action probability calibration, overlapping snippets of different sizes can be used to parse videos without losing information at each temporal point. Based on calibrated action probabilities, we then propose the class-customized filter mechanism that gradually extends and filters local action recognition results to long-term action regions.
2. Related Work

2.1. Action Recognition

Action recognition is the fundamental research of video analysis, which involves the classification of short-trimmed videos using end-to-end deep learning methods. There are two typical architectures of deep learning based action recognition models: CNN-based and Transformer-based.

The CNN-based model adopts convolution blocks as the basic network units. 2D-based methods first extract spatial features and then fuse the spatial features through temporal modeling [13, 16, 30, 31]. Independent temporal modeling augments temporal features with flexible motion information. Video inherent motion information, e.g., spatial difference [14, 20], short-term difference [29] and learnable adaptive motion information [12, 22], have an impact on the 2D-based methods. 3D-based methods process the spatial-temporal information of videos as 3D objects directly [3, 27, 34]. SlowFast [10] and X3D [9] explore the influence on performance of different model dimensions, and propose high-performance network structures according to different task scales. Recently proposed transformer-based methods split videos into 3D tokens and utilize transformer layer to extract the spatial-temporal features [1, 2, 8, 21].

DCAN [5] improved the quality of the generated action proposals by aggregating contextual semantics at the boundary level and proposal level to improve the localization performance. Despite the more complex nature of the two-stage approach, it has higher localization accuracy.

One-stage action localization does not generate proposals and intends to localize the action directly without generating a proposal in a single shot [17, 19, 33, 37, 38]. Actionformer [37] utilizes the FPN of the Transformer to perform one-stage action localization. However, previous approach [28] proved that the above method is not applicable to the AI City Challenge Track3 dataset [24, 25] due to the limited number of training samples. The previous methods adopt post-processing techniques to obtain action localization results, without requiring additional training. But in their solutions, the temporal resolution is related to the size of the snippet, and the classification results at the snippet level are directly assigned to all frames, neglecting the difference between the boundary frames and the central frames in the snippet. Our proposed method also adopts the training-free post-processing solution. But we design a novel probability calibration method to overcome the mentioned problems without introducing extra inference costs.

3. Method

3.1. Overall pipeline

As shown in Fig. 1, the main components of our proposed process are as follows:

- **Snippet-level Action Recognition**: The input video is divided into snippets of varying sizes, then recognition models are used to predict the action probability of each snippet.

- **Action Probability Calibration**: We propose the reliability score for each frame to guide aggregating

![Figure 1. Overall pipeline. The input video is divided into overlapping snippets of various sizes, which are then processed by action recognition (AR) models to predict snippet-level action probabilities. These probabilities are then mapped to frame-level and calibrated using the proposed calibration method. The calibrated probabilities are then fed into the action localization component to generate action regions.](image-url)
the action probabilities of different snippets. This allows us to calibrate snippet-level action probabilities to frame-level probabilities while maintaining inference cost. Subsequently, the frame-level probabilities are integrated and calibrated from multiple viewpoints using prior knowledge to obtain the final probability score.

- **Action Localization**: Based on the frame-level action probabilities, we gradually expand the action boundaries to obtain the final localization results. Additionally, we propose a category-customized filtering mechanism to filter the redundant candidates.

### 3.2. Snippet-level Action Recognition

Given a video, as shown in Fig. 2, we divide it into $n$ overlapped snippets $\{S_1, S_2, \ldots, S_n\}$. Then $F$ frames are sampled from each snippet with an interval of $R$. These sampled frames are then fed into an action recognition (AR) model to obtain the action probability scores $P \in \mathbb{R}^C$ for each snippet. We split the training videos into meaningful segments according to the annotations and use these segments to train the action recognition network. Though we can choose arbitrary action recognition models to predict action probabilities, we chose the lightweight X3D model [9] due to the characteristics and scale of the naturalistic driving action dataset.

### 3.3. Action Probability Calibration

The above-mentioned procedure yields action probabilities for each video snippet. [15, 28] determine the probability of each frame position as the average score of various snippets containing the frame, thus making the action recognition results owe a snippet-level temporal resolution.

Accordingly, using the larger snippet size will cause coarser recognition results, which will further deteriorate the subsequent localization performance. Moreover, it seems unreasonable to assign the snippet-level action category probability to all frames within the snippet.

To address these problems, we propose a training-free probability calibration method to generate frame-level action probability scores from snippet-level results, as shown in Fig. 3. The method includes allocating the snippet-level action probability scores to each frame within the snippet and pre-defining the reliability of the probability allocation for different frames, which determines whether to trust the allocated action category score. Specifically, we assume that the reliability of the allocated probability follows a Gaussian distribution at different positions within the snippet. The distribution makes sure that the frames closer to the middle have higher reliability than the frames closer to the edges, which is in line with common sense. The reliability weight for frame $f$ that is $l_f$ distance away from the center of snippet $s$ is formulated as:

$$w_{f,s} = \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{l_f^2}{2\sigma^2}} \quad (1)$$

where $\sigma$ is the standard deviation. According to this formula, the reliability weights at the edges decrease as the size of the video snippet increases. Based on the reliability weight of probability, we define the probability score for
After calibrating the action probability, we choose the class with the highest score as the classification result, and its corresponding probability score is assigned as the confidence score. As shown in Fig. 4, for each category, we determine a threshold to filter out frames with low confidence scores. After the filtering process, we obtained some rough candidate regions of interest for the foreground, i.e. distracted driving actions. As these candidate regions represent local results, we next merge regions belonging to the same action category and with intervals not exceeding $\lambda_1$ to obtain candidate regions of longer duration. Then, we again filter the candidates that are too short (i.e. duration is smaller than $\lambda_2$). We further define the confidence of candidate regions as the average confidence score of the frames within the region. Then, for each category, we filter out candidate regions with low confidence scores to obtain the final localization results. As candidate regions can be redundant, we set the filter threshold for class $c$ as $\text{Max}(P(c)) \times \text{ratio}$. 

\section{4. Experiment and Results}

In this section, we present the experiments results of our proposed method on the AI City Challenge 2023 Track3 Dataset. Additionally, we provide a detailed description of the dataset, evaluation metric, implementation details, and extensive ablation experiments.

\textbf{Datasets.} The AI City Challenge 2023 Track3 dataset consists of 210 videos performed by 35 drivers, totaling approximately 34 hours. The dataset is divided into three subsets: training set A1, validation set A2, and testing set B, which contains videos performed by 25, 5, and 5 drivers, respectively. The dataset was collected by having each of the 35 drivers complete two data collection tasks under different appearance blocks. Each task consisted of 16 different activities, such as talking on the phone, eating, and reaching back. The activities were completed in approximately eight minutes each. Videos were recorded synchronously at 30 fps from three perspectives: Dashboard, Rear View, and Right Side. Hence, A1, A2 and B contain 150, 30, 30 videos respectively.

\textbf{Evaluation metric.} Evaluation for AI City Challenge 2023 track 3 is based on activity identification performance, measured by the average activity overlap score, which is defined as follows:

\begin{equation}
\text{os}(p, g) = \frac{\text{max}(\text{min}(ge, pe) - \text{max}(gs, ps), 0)}{\text{max}(ge, pe) - \text{min}(gs, ps)},
\end{equation}

where $gs$ and $ge$ are the start time and end time of ground-truth activity $g$, respectively. $p$ is the best predicted activity match of the same category to $g$, os means highest overlap. With the additional condition that the start time $ps$ and end time $pe$ of the predicted activity must fall within a temporal range of $[gs - 10s, gs + 10s]$ and $[ge - 10s, ge + 10s]$, respectively. The overlap between $g$ and $p$ is defined as the ratio between the time intersection and the time union of the two activities. After matching each ground truth activity in order of their start times, any unmatched ground truth activities or unmatched predicted activities will receive an over-
lap score of 0. The final score is calculated as the average overlap score among all matched and unmatched activities.

4.1. Implementation Details.

Training action recognition models. We utilize the Kinetics-400 [3] pre-trained X3D-L [9] to predict action probabilities. To train the action recognition model, we trim the long videos in the training set (A1) to meaningful action segments according to the annotations. To better utilize the information from the background class, we also trim the unlabeled video (belonging to the background class) regions to form the enlarged training set. We call the enlarged training set A1\_expand. For each input video, two options are available for frames sampling: \( F = 8 \) or \( F = 16 \) frames, with sample rate \( R \) setting as 4, 8, 12 or 2, 4, 6. So, the snippet lengths are 32, 64, or 96. For each fixed-length snippet, we then vary the overlap ratio to 0%, 25%, 50%, and 75%, respectively. The initial learning rate is 5e-4. We utilize the Adam [11] optimizer with cosine annealing [23] as the learning rate schedule. The batch size is set as 48 and the number of total train epochs is 35. The input is first resized to 512 \times 512 and then cropped to 448 \times 448. We adopt scale jitter, rand augment [6] and mixup [39] for data augmentation. All the models are trained on 2 NVIDIA GeForce RTX 3090 GPUs.

Action probability inference. For Action probability inference, we maintain the same video pre-processing settings. Specifically, we resize each frame of the input video to 512 \times 512, and use a snippet of size 32, 64, and 96 (corresponding to sampled frames \times \text{sample rate}). The standard deviation \( \sigma \) in Eq. (1) used for calibrating the action probability is set to 30, which we find can give reasonable reliability weights to frames near the snippet boundary. Finally, we ensemble the results on A1 and A1\_expand by different sampling methods, different snippet overlap rates, and different views.

Temporal action localization. When filtering out frames of low confidence, the specific threshold of class \( c \) is first defined as the average confidence scores of frames recognized as class \( c \) minus a margin (0.1). Then, we compress the thresholds greater than 0.5 to 0.5, and raise the thresholds less than 0.1 to 0.1, in order to adjust the thresholds and make them more reasonable. For firstly filtering candidate regions, \( \lambda_1 \) is set as = 8s, and \( \lambda_2 \) is set as 1s. For second filtering candidate regions, the \( \text{ratio} \) is set as 0.95 for filtering as many redundant regions as possible.

4.2. Main results

Tab. 1 displays the Top-10 methods on the Track3 public leaderboard, where our proposed method achieved the 2nd place with an average overlap score of 0.7041.

<table>
<thead>
<tr>
<th>Rank</th>
<th>TeamID</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>209</td>
<td>0.7416</td>
</tr>
<tr>
<td>2</td>
<td>60(Ours)</td>
<td>0.7041</td>
</tr>
<tr>
<td>3</td>
<td>49</td>
<td>0.6723</td>
</tr>
<tr>
<td>4</td>
<td>118</td>
<td>0.6245</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>0.5921</td>
</tr>
<tr>
<td>6</td>
<td>48</td>
<td>0.5907</td>
</tr>
<tr>
<td>7</td>
<td>83</td>
<td>0.5881</td>
</tr>
<tr>
<td>8</td>
<td>217</td>
<td>0.5426</td>
</tr>
<tr>
<td>9</td>
<td>152</td>
<td>0.5424</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>0.5409</td>
</tr>
</tbody>
</table>

Table 1. Comparison to other submissions methods on AI City Challenge 2023 Track3 A2 validation dataset.

<table>
<thead>
<tr>
<th>Frames Rate</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F = 8 )</td>
<td>0.5697</td>
</tr>
<tr>
<td>( (4,8) )</td>
<td>0.5791</td>
</tr>
<tr>
<td>( (8,16) )</td>
<td>0.5805</td>
</tr>
<tr>
<td>( (4,8,12) )</td>
<td>0.5955</td>
</tr>
</tbody>
</table>

(a) Study on different snippet size.

<table>
<thead>
<tr>
<th>Snippet overlap ratio</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5783</td>
</tr>
<tr>
<td>25%</td>
<td>0.5761</td>
</tr>
<tr>
<td>50%</td>
<td>0.5855</td>
</tr>
<tr>
<td>75%</td>
<td>0.5955</td>
</tr>
</tbody>
</table>

(b) Study on different overlap ratio.

4.3. Ablation studies

All ablation experiments are conducted using our self-defined training and test sets. Specifically, we adopt 25% samples of the given training set as the validation set (user id: 85870, 86356, 86952, 96269, 99882), and the remaining 75% samples for action recognition model training. The ablation experiments are based on the same training and validation sets as described in Sec. 4.1.

Study on different sample strategy. We experimented with different sampling strategies to evaluate the effectiveness of various snippet sizes in probability correction, as provided in Tab. 2 (a). The baseline achieved a score of 0.5697 when the sampling frame was \( F = 8 \) and the sampling rate was \( R = 4 \) (with a snippet size of 32). By integrating model with a sampling rate of \( R = 8 \) (with a snippet size of 64), the score improved to 0.5791. The performance was further improved by continuing to integrate the model with \( R = 12 \). Finally, we integrated all the models with snippet sizes of 32, 64, and 96 for sampling
frames of 8 and 16, achieving the best result with a score of 0.5955. Meanwhile, we conducted detailed experiments on different snippet overlap ratios, as shown in Tab. 2 (b). When the snippets are not overlapped, the final localization score achieved is 0.5783. Adapting 25% overlap results in a degradation of performance. However, when the overlap rate is increased to 75%, the problem of weakened edge information is greatly alleviated by combining our sampling and fusion strategies to further refine the temporal resolution, leading to the best performance.

**Study on action probability calibration settings.**

Tab. 3 (a) showcases our investigations of the probabilistic fusion of different views. It shows that single view probabilities are not sufficient for effective localization post-processing, hence multi-view fusion is necessary. Tab. 3 (b) presents the view fusion settings and category adjustment strategy. As mentioned in Sec. 3.3, it is not reasonable to assign the probability of the snippet level directly to all frames within the snippet, as the reliability of different frames within a snippet can vary. Similarly, the reliability of certain categories may differ across views. Experimental results support these ideas, showing that using mean fusion without adjusting view categories results in average performance. However, using Gaussian fusion and category adjustment operations significantly improves performance.

**Study on different Temporal Action Localization settings.**

Similarly, we conducted detailed experiments on the filtering parameters for post-processing of localization, as shown in Tab. 4. We compare different thresholds for region merging and region filtering. The results indicate that $\lambda_1 = 8$ and $\lambda_2 = 1$ are the most appropriate thresholds.

**References**


