

Temporal Driver Action Localization using Action Classification Methods

Munirah Alyahya, Shahad Alghannam*, Taghreed Alhussan*

Saudi Technology and Security Comprehensive Control Company, Riyadh, Saudi Arabia
{malyahya, salghanam, talhussan}@tahakom.com

Abstract

Driver distraction recognition is an essential computer vision task that can play a key role in increasing traffic safety and reducing traffic accidents. In this paper, we propose a temporal driver action localization (TDAL) framework for classifying driver distraction actions, as well as identifying the start and end time of a given driver action. The TDAL framework consists of three stages: preprocessing, which takes untrimmed video as input and generates multiple clips; action classification, which classifies the clips; and finally, the classifier output is sent to the temporal action localization to generate the start and end times of the distracted actions. The proposed framework achieves an F1 score of 27.06% on Track 3 A2 dataset of NVIDIA AI City 2022 Challenge. The findings show that the TDAL framework contributes to fine-grained driver distraction recognition and paves the way for the development of smart and safe transportation. Code will be available soon.

1. Introduction

Statistics from the World Health Organization [1] indicate that numerous accidents arise from driver distraction. The increase in driver distractions requires an understanding of driver actions, which can ensure traffic safety and reduce the likelihood of road traffic accidents. The problem of recognizing driver distraction has been intensively studied in recent years with the advent of Autonomous Vehicles (AV) and Advanced Driver Assistant Systems (ADAS). An important reason for this is the lack of total dependability on AVs. As cases have indicated, human attention is required in certain cases, such as in Tesla accident cases [2]. In addition, when a driver becomes distracted, the ADAS must have the capability to control the vehicle.

Many researchers have offered definitions of the phrase “driver distraction.” Lee et al. [3] defined driver distraction as “the diversion of attention away from activities for safe

driving toward a competing activity.” Driver distraction can be categorized into manual and mental types. In the case of manual distraction, the hand is occupied with something else rather than the wheel. By contrast, for mental distraction, the driver's mind is occupied with something other than driving [4]. Manual distractions are detectable using advances in the field of computer vision. The common challenges of driver distraction recognition from the visual perspective are lighting conditions (lighting of out-of-road components, sunlight, and vehicle light), face occlusions, hand movement and shadow illusion, insufficient training data, and changes in the drivers' head pose and eye movements [5].

Most existing driver action recognition methods classify driver actions based on images [2], [5]–[10]. These methods fail to capture long-term actions due to the absence of temporal information. Certain methods use pose estimation to classify driver actions [11]–[15]. However, only using pose information can lead to the loss of important spatial data. Finally, to the best of our knowledge, no prior research on driver action recognition has studied the problem of temporal driver action localization.

With the aim of accounting for the above-mentioned research gaps and challenges in the domain, the NVIDIA AI City 2022 Challenge established a new challenge track (Track 3) to address the naturalistic driver data analytics problem. The objective of Track 3 is to find the start time and end time for 18 driver distraction actions in an untrimmed video. This challenge motivates us to study naturalistic driver data analytics problems from the action recognition perspective.

In this paper, we reformulate the problem from temporal driver action localization to the action classification problem, through the development TDAL framework. The proposed framework consists of three stages: preprocessing, action classification, and temporal localization. The preprocessing stage takes untrimmed video and generates equal-length clips. These clips are fed into the action classification model. Then, the outputs of the classification model are sent to the last stage, which performs temporal action localization. This generates the start and end times of all the driver distracted actions in the

* These authors contributed equally to this work.

untrimmed video. The proposed method achieved an F1 score of 27.06% without using external data.

The rest of the paper is organized as follows. Section 2 provides a review of the related work. The proposed method is present in Section 3. Section 4 Experiments. Finally, Section 5 presents the conclusion of the paper.

2. Related Work

Driver Action Recognition. Driver action recognition has been extensively studied in recent years, but it is still undergoing active and continuous exploration. Different approaches have been used in several studies. The most recent approaches are based on supervised learning techniques. Following the success of 2D convolutional neural networks (CNN) in extracting spatial information, many researchers have used 2D CNNs to classify driver actions [2], [5]–[10]. Recently, a hybrid deep learning-based approach was proposed [5], which is a stack Bidirectional Long Short-Term Memory (BiLSTM) network with the pre-trained CNN Inception-V3, to capture both spectral and spatial features. The weighted ensemble method in [7] was based on evaluating a weighted sum of pre-trained networks, Inception-V3 and AlexNet, on five image sources: raw images, skin-segmented images, face images, hands images, and “face+hands” images. In [8] and [9], the authors used a foreground segmentation algorithm to differentiate between the driver and the background, and then the result was fed into CNNs. By contrast, the authors in [2] modified VGG-16 by replacing the fully-connected (FC) layers with convolutional layers. This contributed to a reduction in the number of parameters. The RCNN object detection algorithm was used in [10]. The authors adopted Gaussian Mixture Model (GMM) for skin-like region extractor as a region proposal, rather than using selective search. The main limitation of these approaches is that they are designed to learn how to classify driver actions from the image level, in the absence of temporal information.

Diverse approaches have been applied using pose estimations [11]–[14]. Existing techniques utilize 2D pose features and incorporate them with other features to recognize driver distractions. In [11], driver pose is fused with other features (flow of pose (HoDF) and the interactions of the driver with objects) to construct handcrafted features, after which the following three classifiers were used: Naive Bayes, Random Forest, and Support Vector Machine. On the other hand, [12]–[14] integrated pose features with CNN features to generate informative and discriminative features. Other techniques have used 3D pose with Graph Convolutional Networks (GCNs) [15] or Recurrent Neural Networks (RNN)[16]. The shared limitation of each of these methods is that they only use a limited list of actions.

Action Recognition. Action recognition focuses on studying human behaviors. The emergence of large-scale

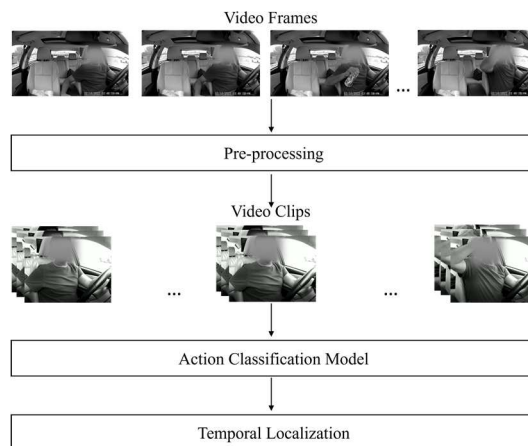


Figure 1: TDAL framework.

datasets, such as kinetics [17] and something to something [18], has significantly improved research works in the action classification area. These works often use a trimmed video to classify a sampled clip from a video into specific action classes. The network architectures that have been extensively used in the literature, and that have achieved considerable performance, can be classified into two types: the first type takes RGB and optical flow frames as input to capture spatial (i.e., appearance) and motion information, respectively, such as I3D [19] and TSN [20]. The main disadvantage of this type is that in some works the two streams are learned separately; also, using optical flow will increase the system overhead. The second type uses a 3D network that captures spatial-temporal information using a stack of RGB frames as input, such as C3D [21] and SlowFast [22].

3. Proposed Method

The TDAL framework consists of three stages, as illustrated in Figure 1. The first stage is preprocessing, which takes video as input and generates N clips. Each clip is passed as input to the action classifier model. The classifier output is then sent to the final stage to generate the actions’ start and end times.

3.1. Problem Formulation

Suppose that $D = \{D_{train}, D_{test}\}$ is a raw dataset, where the training dataset $D_{train} = \{uv_n, \Psi_n\}_{n=1}^{l_{train}}$, and l_{train} is the total number of untrimmed videos. Each data instance of D_{train} consists of an untrimmed video uv_n and corresponding ground truth Ψ_n (multi-label video). The untrimmed videos are infrared with 30 frames per second (fps). Ψ_n can be represented as tuples $\{(t_s, t_e, y_m)\}_{m=1}^C$, where t_s, t_e are the start time and end time, respectively, y_m is the class number and C is the number of distracted driver actions in the untrimmed video uv_n . The challenge

is to develop a framework that can predict the distracted driver actions class in each instance in D_{test} , including the start and end times, where the test set is $D_{test} = \{uv_n\}_{n=1}^{l_{test}}$ and l_{test} is the total number of untrimmed videos.

This paper aims to leverage the large-scale trimmed videos dataset in the action classification field. Therefore, we propose a method that learns from trimmed videos to identify the distracting actions classes, start and end times in the untrimmed videos, D_{test} . As such, we crop each untrimmed video in D_{train} using the associated t_s, t_e for each action class y_m in C . The new training set is $X = \{v_n, y\}_{n=1}^{l_x}$, where each instance in X consists of trimmed video v_n with a class label y , and l_x is total number of distracted actions trimmed videos.

3.2. Pre-processing

Driver Tracking. Since cars may contain drivers and passengers, we adopted the You Only Look Once (Yolov5) model [23] pre-trained on the COCO dataset [24] to detect and track driver spatial location in the video. YOLO [25] is a single-stage detector that has high performance in processing frames in real-time compared to other state-of-the-art methods (e.g., Faster-RCNN and SSD). YOLOv5 is the last version from the YOLO family, which achieved good results in terms of inference speed and accuracy [23].

The driver tracking algorithm designed in this research consists of two steps. In the first step, we use the Yolov5 model to identify all the individuals' bounding boxes in the uv_n frames. We set the confidence threshold to λ . In the second step, we compute the area of all predicted individuals' bounding boxes, after which the video is cropped using the individual bounding box with the largest area. The purpose of considering the bounding box with the largest area is to ensure that the whole of the driver is included in the video. Cropping uv_n frames reduces the noise, which enables the next stage to focus only on the information relevant to the driver.

Video Segmentation. We split each instance in D_{test} into N clips. Each clip should be sufficiently large to capture temporal information relating to the relevant action. We selected 64 frames as the clip size $clipSize$, which is the typical temporal depth [22]. In this paper, we investigated two different splitting settings, as shown in Algorithm 1 and Algorithm 2. These two settings aid the framework in accurately classifying the actions and identifying its boundaries (i.e., start and end times). The first setting (type 1) examines driver actions that occur in two consecutive seconds. In this case, uv_n is divided into N clips and the value of N is equal to the video length in seconds divided by two seconds.

By contrast, in the second setting (type 2), driver actions are examined that occur only in one second. In the latter case, uv_n is divided into N clips, where N is equal to video length in seconds. As mentioned above, the clip size should

Algorithm 1 Video Segmentation – type 1

Input: $frameList, clipSize, videoLength, videoFullRate$
1: $startIndex \leftarrow 0$
2: $endIndex \leftarrow clipSize$
3: $numberOfClips \leftarrow videoLength / (2 \times videoFullRate)$
4: $clip \leftarrow []$
5: **for** $i = 1$ **to** $numberOfClips$ **do**
6 **for** $j = startIndex$ **to** $endIndex$ **do**
7 $clip.insert(frameList[j])$
8 **end for**
9 **save clip as video**
10 $clip \leftarrow []$
11: $startIndex \leftarrow endIndex - 4$
12: $endIndex \leftarrow startIndex + clipSize$
13: **end for**

Algorithm 2 Video Segmentation – type 2

Input: $frameList, clipSize, videoLength, videoFullRate$
1: $startIndex \leftarrow 0$
2: $endIndex \leftarrow clipSize$
3: $numberClips \leftarrow videoLength / videoFullRate$
4: $centerIndex \leftarrow videoFullRate + (videoFullRate / 2)$
5: **for** $i = 2$ **to** $numberClips$ **do**
6 $startIndex \leftarrow centerIndex - (clipSize / 2)$
7 $endIndex \leftarrow centerIndex + (clipSize / 2)$
8 **save frameList[startIndex] to**
 $frameList[endIndex]$ **as video**
9 $centerIndex \leftarrow centerIndex + clipSize$
10: **end for**

contain 64 frames, and so each clip in the type 2 setting takes 17 frames from the previous and post seconds.

3.3. Action Classification Model

For the TDAL framework to succeed, the action classifier plays a central role. Training the action classification model requires huge data and computational resources to reach a satisfactory result. Thus, this work adopted SlowFast [22] as an action classifier with Resnet50 [26] as the backbone. This decision was made according to a trade-off between training computational resources, accuracy, and inference speed.

Pre-processing. For each video in X , we performed driver tracking, but in a different way from Section 3.2. We use Yolov5 model to identify all the individuals' bounding boxes in the v_n frames with the confidence score equal to or greater than λ . We link the bounding boxes through frames based on the intersection of union (IoU) threshold α to construct a tube for each individual. Then, each tube's frames will be cropped based on the first bounding box and stored as a video. After that, we clean the resulted videos manually and remove any videos that may negatively affect the training phase.

The training set of this challenge is relatively small with respect to the number of classes and public action classification dataset. Additionally, some classes had a high similarity, such as a driver singing and driver talking to a passenger. If we had trained our model using only the training set, either overfitting or underfitting problems would have emerged. Therefore, to avoid these problems in our research, we increased the number of samples in each class using synthetic data in addition to the X set. Since all the videos are greyscale in terms of color, the image colorization technique of Su et al. [27] was applied to perform colorization on the entire training set X .

Training Procedure. We used several techniques in the training phase to address the problem of underfitting. We initialized the model weights using pre-trained weights for similar datasets, such as kinetics [17]. In turn, we trained the model using only X set and without using the synthetic data until the model was overfitted. Following this, we reduced overfitting by resuming training after adding the synthetic data (colored data) to the training set. Also, we increase the number of random augmentation samples in each epoch during training. The last two steps were undertaken to improve the model’s generalization capability.

3.4. Temporal Localization

As indicated in Section 2, the performance of action classification is promising. We sought to exploit this fact by using the classifier output in this stage and storing the action probability p for each clip in uv_n . Following this, the probabilities P are passed through the temporal localization algorithm presented in Algorithm 3.

The temporal localization procedure is undertaken in three steps. In the first step, we take the most promising temporal information for a certain action. As such, the top k probabilities are returned that are associated with its clips for a specific action n . The second step aims to construct the proposed intervals for a certain action; the interval is the period of time where the action is happening, which can be specified by the start time and end time of the period. However, the two consecutive clips will form an *interval* if the time that separates them is less than *acceptSec*. Otherwise, it will be considered a new proposed interval. After that, the proposed intervals are filtered; if the interval must be greater than *rejectSec*, then it is sorted based on the max-average probability. In the third step, we ensure that no two actions’ intervals overlap; otherwise, the action with lower priority updates its interval by the next max-average probability. The action priority is defined as follows: the action with the higher priority is that for which the action classification model has performed more effectively. The above-mentioned steps are repeated for each action class.

Algorithm 3 Temporal Localization

Input: P /* probabilities for each clip in video */
1: $seqInterval \leftarrow []$ /* contains sequence of clip for a certain action */
2: $interval \leftarrow []$ /* contains all proposed intervals (start and end sec) for certain action */
3: $intervals \leftarrow []$ /* contains intervals for all actions */
4: **for** n **in** $actionsClasses$
5: **Step 1:** getting the top k probabilities associated with clips
6: $clip, prob \leftarrow GetTopK(P, n)$
7: **Step 2:** generating temporal proposals
8: **for** s **in** $clips$
9: **if** $next\ s \leq s + acceptSec$ **do**
10: $seqInterval.insert(s)$
11: **else**
12: $startTime \leftarrow seqInterval[0].toSec()$
13: $endTime \leftarrow seqInterval[-1].toSec()$
14: $interval.insert(startTime, endTime)$
15: $seqInterval \leftarrow []$
16: **end if**
17: **end for**
18: Filter intervals ($interval, rejectSec$)
19: Sort max avrage ($interval$)
20: $intervals[n] \leftarrow interval[0]$
21: **end for**
22: **Step 3:** updating the overlapped action
23: **while** $interval$ **overlapped** $intervals$
24: $intervals[n] \leftarrow Get\ next\ proposed\ interval$
25: **end while**
Output $intervals$

4. Experiments

4.1. Data Analysis

Experiments were performed using the NVIDIA AI City Challenge 2022 Track 3 dataset. This is a third-party dataset [28] consisting of data captured from 10 drivers. The drivers performed 18 different tasks in random order under two conditions: once in the absence of an appearance blocker (e.g., hat or sunglasses) and another in presence of an appearance blocker.

The Track 3 dataset provides three different angles recorded simultaneously by infrared cameras, resulting in 60 videos in total. The average length of a video is 9.3 minutes with the frame rate of 30 fps. Additionally, the dataset is partitioned into two subsets A1, and A2, each containing 5 different drivers. The aim of the A1 and A2 datasets is to develop the algorithm. In which A1 for training and A2 for evaluation. The main objective of this challenge is to classify driver actions into predefined classes and find the start and end times without using any

Table 1: A1 training and testing dataset splits.

Class Number	Action Name	A1 Train			A1 Test		
		videos	Total Seconds	Average (secs/video)	videos	Total Seconds	Average (secs/video)
0	Normal Forward Driving	6	109	18	2	34	17
1	Drinking	6	110	18	2	31	16
2	Phone Call (right)	8	131	16	2	30	15
3	Phone Call (left)	6	110	18	2	38	19
4	Eating	6	110	18	2	41	20
5	Text (Right)	7	108	15	2	32	16
6	Text (Left)	6	118	20	2	36	18
7	Hair / makeup	6	145	24	2	29	14
8	Reaching behind	6	115	19	2	48	24
9	Adjust control panel	6	125	21	2	45	22
10	Pick up from Driver floor	6	126	21	2	31	16
11	Pick up from Passenger floor	7	97	14	2	32	16
12	Talk to passenger at the right	6	123	20	2	44	22
13	Talk to passenger at backseat	6	116	19	2	29	14
14	yawning	7	109	16	3	23	8
15	Hand on head	7	121	17	2	36	18
16	Singing with music	6	117	20	2	48	24
17	shaking or dancing with music	6	140	23	2	29	14

external data or A2 set in the training phase.

A1 dataset. A1 is a set of labeled untrimmed videos. We trimmed only the rear angle view based on the actions’ ground-truth. After that, we analyzed the trimmed videos and cleaned them by omitting any driver who engaged in a complex action (e.g., singing with music while texting or talking to the passenger in the backseat while texting). The remaining A1 dataset was divided into a training set 75% and a test set 25%. Table 1 shows the A1 dataset. For more information about the dataset split, it will be available on the GitHub repository soon.

4.2. Implementation Details

Driver tracking. We used Yolov5, the small model (Yolov5s) version [23] pre-trained on COCO dataset [24] to detect and track each person in a video. The input frames size is fixed as the benchmark video size (1920 \times 1080). Experimentally, we conducted the finest confidence threshold $\lambda = 0.5$.

Action classification model. For the preprocessing, we set α to 0.30 and λ to 0.5. We used SlowFast-R50 as action classifier model, which is pretrained on the Kinetics-400 dataset [17] and fine-tuned on X sets. First, we train it using the default hyperparameters with some modification on each. We set max epoch to 440, batch size to 12, number

of samples for data augmentation is set to 1, and the base learning rate is set to 1e-5. After that, we increased our dataset through applying image colorization [27] using their default hyperparameters. Then, we resume the training from the last checkpoint in first step till epoch 730, with batch size set to 4, number of samples for data augmentation is set to 3, base learning rate is set to 1e-4, and train jitter scales [256, 256].

Temporal localization. By experimentation, we set $k = 12$, $acceptSec = 10$ and $rejectSec = 4$.

4.3. Evaluation Metrics

In order to evaluate stage 2, we consider top1- accuracy and top-5 accuracy measured by the equation (1). Top-1 accuracy is prevalent accuracy; it checks if the highest probability matches the class label. Whereas top-5 accuracy, checks if the one of the highest top-5 probabilities matches the class label to consider it as a correctly classified action.

$$Accuracy = \frac{T}{a} \quad (1)$$

where T is the number of correctly classified actions, and a is the total number of actions.

The evaluation of NVIDIA AI CITY CHALLENGE 2022 for track 3 will be based on the action identification performance model, measured by the F1 score presented in equation (2). F1 score is a harmonic mean of recall and precision. Specifically, a true-positive (TP) activity identification will be considered when the action was correctly identified as starting within one second of the ground-truth start time and ending within one second of the ground-truth end time of the activity. Whereas a false-positive (FP) is the activity in the proposed approach that fails to be correctly identified within the slot. Lastly, a false-negative (FN) activity is a ground-truth activity that was not correctly identified.

$$F1\ score = \frac{2TP}{2TP + FN + FP} \quad (2)$$

4.4. Experiment Results and Discussion

Action Classification Model Evaluation. The first experiment investigated the following question: “Is the model capable of identifying actions correctly?” The proposed action classification model achieved 64% for the top-1 accuracy and 97.3% for the top-5 accuracy on the A1 test set. However, the model failed entirely to identify certain actions, as shown in Figure 2. These actions include normal driving, texting (left), talking to a passenger on the right, talking to a backseat passenger, and singing to music. One possible reason is the similarity between the actions, such as talking to a passenger on the right and talking to a passenger in the backseat, where the two actions require gesturing to the right. In addition, the camera angle (rear view) influenced the result in the text (left) action, which is due to the fact that the phone and hand movements were hidden by the vehicle wheel. This result indicates that the model needs more information to distinguish between the top-5 actions. This can be achieved either by increasing the size of the training dataset or using more than one camera angle to uniquely represent each action.

This paper also sought to investigate the learned features. Therefore, features were extracted from the last global pooling layer, and we used UMAP [29] to project it to lower dimensions. Figure 3 presents the feature visualization for the A1 test set. As the Figure 3 shows, the action classification model was capable of finding a level of similarity between the features of each action.

Evaluation on TDAL framework. The proposed TDAL framework was tested on the A2 set provided by the NVIDIA AI City 2022 Challenge team. We tested the two video segmentation settings in Section 3.2. Table 2 shows the F1 score for the two algorithms. From the results in Table 2, Algorithm 1 outperformed Algorithm 2 by a margin of 14.1%.

Finally, the proposed framework achieved an F1 score of 27.06%. It is reasonable to conclude from Figures 2 and 3 that the action classification model demonstrates

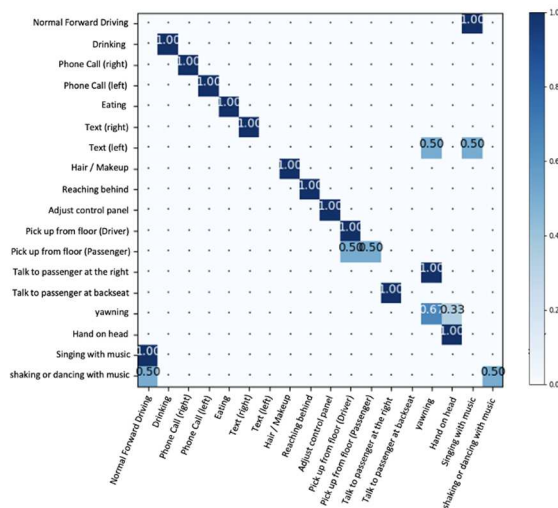


Figure 2: Confusion matrix for top-1 accuracy using A1 test set.

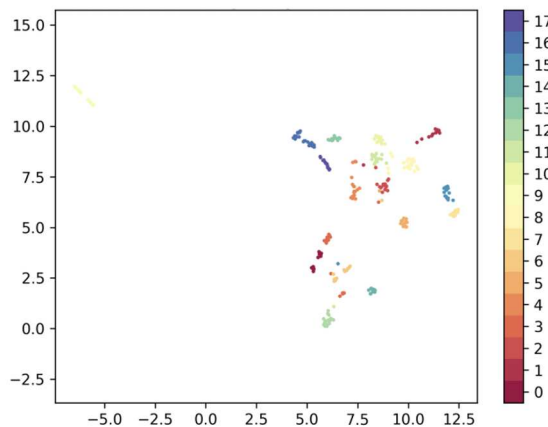


Figure 3: Features visualization for the A1 test set.

superiority and robustness. The reduction in the F1 score is a consequence of the poor performance of the temporal localization algorithm.

Table 2: The performance of our method using different video segmentation settings on A2 set. **A** is the results of using Algorithm 1 and **B** is the results of using Algorithm 2.

	A	B
F1-score	0.2706	0.1316

5. Conclusion

This paper reformulated the temporal driver action localization problem to the action classification problem. The proposed framework consists of three stages: preprocessing, action classification and temporal localization. We showed that using an action classification


algorithm can enable the identification of a given action's start and end time. In Track 3 of the NVIDIA AI City Challenge 2022, the proposed framework achieved an F1 score of 27.06%. In the future, we intend to enhance the temporal localization algorithm.

Acknowledgments

The authors would like to thank Saudi Technology and Security Comprehensive Control Company (Tahakom) for funding and supporting this research. Also, the authors would like to thank the General Manager of the Systems Engineering department at Tahakom Professor Abdulrahman Alarifi for his valuable discussions, insightful comments, and ongoing support throughout this research.

References

- [1] "Home." <https://www.who.int> (accessed Apr. 14, 2022).
- [2] B. Baheti, S. Gajre, and S. Talbar, "Detection of Distracted Driver Using Convolutional Neural Network," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Salt Lake City, UT, USA, Jun. 2018, pp. 1145–11456. doi: 10.1109/CVPRW.2018.00150.
- [3] Michael A. Regan, John D. Lee, Kristie Young, Ed., *Driver Distraction Theory, Effects, and Mitigation*, 1st ed., vol. 13. 2008. [Online]. Available: <https://www.taylorfrancis.com/books/mono/10.1201/9781420007497/driver-distraction-michael-regan-john-lee-kristie-young>
- [4] N. Moslemi, M. Soryani, and R. Azmi, "Computer vision-based recognition of driver distraction: A review," *Concurrency Computat Pract Exper*, vol. 33, no. 24, Dec. 2021, doi: 10.1002/cpe.6475.
- [5] J. Mafeni Mase, P. Chapman, G. P. Figueredo, and M. Torres Torres, "A Hybrid Deep Learning Approach for Driver Distraction Detection," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, Korea (South), Oct. 2020, pp. 1–6. doi: 10.1109/ICTC49870.2020.9289588.
- [6] A. Jamsheed V., B. Janet, and U. S. Reddy, "Real Time Detection of driver distraction using CNN," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, Aug. 2020, pp. 185–191. doi: 10.1109/ICSSIT48917.2020.9214233.
- [7] H. M. Eraqi, Y. Abouelnaga, M. H. Saad, and M. N. Moustafa, "Driver Distraction Identification with an Ensemble of Convolutional Neural Networks," *Journal of Advanced Transportation*, vol. 2019, pp. 1–12, Feb. 2019, doi: 10.1155/2019/4125865.
- [8] M. Leekha, M. Goswami, R. R. Shah, Y. Yin, and R. Zimmermann, "Are You Paying Attention? Detecting Distracted Driving in Real-Time," in *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, Singapore, Singapore, Sep. 2019, pp. 171–180. doi: 10.1109/BigMM.2019.00-28.
- [9] Y. Xing *et al.*, "End-to-End Driving Activities and Secondary Tasks Recognition Using Deep Convolutional Neural Network and Transfer Learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, Jun. 2018, pp. 1626–1631. doi: 10.1109/IVS.2018.8500548.
- [10] S. Yan, Y. Teng, J. S. Smith, and B. Zhang, "Driver behavior recognition based on deep convolutional neural networks," in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, Changsha, China, Aug. 2016, pp. 636–641. doi: 10.1109/FSKD.2016.7603248.
- [11] H. Naveed, F. Jafri, K. Javed, and H. A. Babri, "Driver activity recognition by learning spatiotemporal features of pose and human object interaction," *Journal of Visual Communication and Image Representation*, vol. 77, p. 103135, May 2021, doi: 10.1016/j.jvcir.2021.103135.
- [12] M. Wu, X. Zhang, L. Shen, and H. Yu, "Pose-aware Multi-feature Fusion Network for Driver Distraction Recognition," in *2020 25th International Conference on Pattern Recognition (ICPR)*, Milan, Italy, Jan. 2021, pp. 1228–1235. doi: 10.1109/ICPR48806.2021.9413337.
- [13] A. Behera, Z. Wharton, A. Keidel, and B. Debnath, "Deep CNN, Body Pose, and Body-Object Interaction Features for Drivers' Activity Monitoring," *IEEE Trans. Intell. Transport. Syst.*, vol. 23, no. 3, pp. 2874–2881, Mar. 2022, doi: 10.1109/TITS.2020.3027240.
- [14] A. Behera and A. H. Keidel, "Latent Body-Pose guided DenseNet for Recognizing Driver's Fine-grained Secondary Activities," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Auckland, New Zealand, Nov. 2018, pp. 1–6. doi: 10.1109/AVSS.2018.8639158.
- [15] P. Li, M. Lu, Z. Zhang, D. Shan, and Y. Yang, "A Novel Spatial-Temporal Graph for Skeleton-based Driver Action Recognition," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, New Zealand, Oct. 2019, pp. 3243–3248. doi: 10.1109/ITSC.2019.8916929.
- [16] M. Martin, J. Popp, M. Anneken, M. Voit, and R. Stiefelhagen, "Body Pose and Context Information for Driver Secondary Task Detection," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China, Jun. 2018, pp. 2015–2021. doi: 10.1109/IVS.2018.8500523.
- [17] W. Kay *et al.*, "The Kinetics Human Action Video Dataset," *arXiv:1705.06950 [cs]*, May 2017, Accessed: Apr. 14, 2022. [Online]. Available: <http://arxiv.org/abs/1705.06950>
- [18] R. Goyal *et al.*, "The 'something something' video database for learning and evaluating visual common sense," *arXiv:1706.04261 [cs]*, Jun. 2017, Accessed: Apr. 14, 2022. [Online]. Available: <http://arxiv.org/abs/1706.04261>
- [19] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," *arXiv:1705.07750 [cs]*, Feb. 2018, Accessed: Apr. 14, 2022. [Online]. Available: <http://arxiv.org/abs/1705.07750>
- [20] L. Wang *et al.*, "Temporal Segment Networks for Action Recognition in Videos," *arXiv:1705.02953 [cs]*, May 2017, Accessed: Apr. 14, 2022. [Online]. Available: <http://arxiv.org/abs/1705.02953>

- [21] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," *arXiv:1412.0767 [cs]*, Oct. 2015, Accessed: Apr. 14, 2022. [Online]. Available: <http://arxiv.org/abs/1412.0767>
- [22] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast Networks for Video Recognition," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), Oct. 2019, pp. 6201–6210. doi: 10.1109/ICCV.2019.00630.
- [23] "GitHub - ultralytics/yolov5: YOLOv5  in PyTorch > ONNX > CoreML > TFLite." <https://github.com/ultralytics/yolov5> (accessed Apr. 14, 2022).
- [24] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," *arXiv:1405.0312 [cs]*, Feb. 2015, Accessed: Apr. 14, 2022. [Online]. Available: <http://arxiv.org/abs/1405.0312>
- [25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv:1506.02640 [cs]*, May 2016, Accessed: Apr. 21, 2022. [Online]. Available: <http://arxiv.org/abs/1506.02640>
- [26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, Accessed: Apr. 14, 2022. [Online]. Available: <http://arxiv.org/abs/1512.03385>
- [27] J.-W. Su, H.-K. Chu, and J.-B. Huang, "Instance-Aware Image Colorization," p. 10.
- [28] Mohammed Shaiqur Rahman, "Synthetic Distracted Driving (SynDD1) Dataset." MURI/AUSMURI Project: Rationalization of Interphase Instabilities during Thermo-Mechanical Gyration Typical, Apr. 19, 2022. doi: 10.17632/PTCP7RP3WB.2.
- [29] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *arXiv:1802.03426 [cs, stat]*, Sep. 2020, Accessed: Apr. 14, 2022. [Online]. Available: <http://arxiv.org/abs/1802.03426>