RALACs: Action Recognition in Autonomous Vehicles Using Interaction Encoding and Optical Flow

Eddy Zhou, Owen Leather*, Alex Zhuang, Alkasim Budhwani*, Rowan Dempster*, Quanquan Li*, Mohammad Al-Sharman*, Derek Rayside, Member, IEEE, and William Melek*, Senior Member, IEEE

Abstract—When applied to autonomous vehicle (AV) settings, action recognition can enhance an environment model’s situational awareness. This is especially prevalent in scenarios where traditional geometric descriptions and heuristics in AVs are insufficient. However, action recognition has traditionally been studied for humans, and its limited adaptability to noisy, unclipped, unpampered, raw RGB data has limited its application in other fields. To push for the advancement and adoption of action recognition into AVs, this work proposes a novel two-stage action recognition system, termed RALACs. RALACs formulates the problem of action recognition for road scenes, and bridges the gap between it and the established field of human action recognition. This work shows how attention layers can be useful for encoding the relations across agents, and stresses how such a scheme can be class-agnostic. Furthermore, to address the dynamic nature of agents on the road, RALACs constructs a novel approach to adapting Region of Interest (ROI) alignment to agent tracks for downstream action classification. Finally, our scheme also considers the problem of active agent detection, and utilizes a novel application of fusing optical flow maps to discern relevant agents in a road scene. We show that our proposed scheme can outperform the baseline on the ICCV2021 Road Challenge dataset (Singh et al., 2023) algorithm and by deploying it on a real vehicle platform, we provide preliminary insight to the usefulness of action recognition in decision making. The code is publicly available at https://github.com/WATonomous/action-classification.

Received 5 April 2024; revised 25 June 2024, 16 September 2024, and 30 October 2024; accepted 7 December 2024. Date of publication 24 December 2024; date of current version 3 February 2025. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) in partnership with General Motors, Canada, under Grant CRD 537104-18. This article was recommended by Associate Editor Z. Cao. (Corresponding author: Mohammad Al-Sharman.)

Eddy Zhou, Owen Leather, Alkasim Budhwani, Mohammad Al-Sharman, and William Melek are with the Mechanical and Mechatronics Engineering Department and the Waterloo Autonomous Vehicle Team in the SAE AutoDrive Challenge (Watonomous.ca) (WATonomous), University of Waterloo, Waterlo, ON N2L 3G1, Canada (e-mail: eddy.zhou@uwaterloo.ca; owen.leather@uwaterloo.ca; a2budhw@uwatleroo.ca; mkaisharman@uwaterlooo.ca; william.melek@uwaterloo.ca).

Alex Zhuang is with the Cheriton School of Computer Science and the Waterloo Autonomous Vehicle Team in the SAE AutoDrive Challenge (Watonomous.ca) (WATonomous), University of Waterloo, Waterlo, ON N2L 3G1, Canada (e-mail: a3zhuang@uwatleroo.ca).

Rowan Dempster, Quanquan Li, and Derek Rayside are with the Department of Electrical and Computer Engineering and the Waterloo Autonomous Vehicle Team in the SAE AutoDrive Challenge (Watonomous.ca) (WATonomous), University of Waterloo, Waterlo, ON N2L 3G1, Canada (e-mail: r2dempster@uwaterlooo.ca; quanquan.li@uwaterloo.ca; drayside@uwaterloo.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2024.3515104.

Digital Object Identifier 10.1109/TCYB.2024.3515104
and with each other. May have different kinds of interaction with their environments which rules out many top-performing action networks which works [13]. The system must infer online and in real-time, breaks static assumptions in prior human action recognition and agents will be moving drastically between frames which makes existing systems unsuitable. On the road, the camera is nonstatic, Vehicle domain pose a number of challenges that make exist-

Active agents from previous works in human action classifica-
tion and extend its capabilities to capture interactions between multiple road-relevant classes; not just humans and the environment.

1) We adapt the idea of learning interactions between agents from previous works in human action classification and agent tracks are appended with action confidence scores (highest score is color of bounding box).

To address these problems, and sway research directions to bridge the gap between human action recognition and action recognition for the road, we propose the Road Active Agent Localization and Action Classification system (RALACs) (see Fig. 2), a novel scheme for road action recognition, following constraints that are closely tied to real autonomous vehicle implementation. Our main contributions include the following.

2) We formulate the problem of detecting only active agents in a road scene to reduce unnecessary computation and enable online real-time inference, and propose a novel action localization architecture that uses information from optical flow maps to detect active agents.

3) We present the problem of dynamic agents in high-
motion scenes which break assumptions from prior works, and propose a novel algorithm for Dynamic Region of Interest (DROI) alignment to encode agent paths in feature tensors which yields better results than existing single key-frame ROI alignment.

To test RALACs’ viability in autonomous vehicles, we share our initial attempt at implementing such a system on our autonomous vehicle research platform [14]. We also provide preliminary insights to the possible advancements action labels can have on decision making. The remainder of this article is structured as follows. Section II presents other related works. Section III illustrates the proposed action recognition methodology. Section IV illustrates the experimental setup design, implementation details and the results of the proposed architecture followed by a discussion. Finally, Section V outlines the proposed and future works.

II. RELATED WORKS

A. Human Action Recognition

Human action recognition is a rather mature field. The task of human action recognition can be split into two subtasks: 1) action localization and 2) action classification [16], [17], [18]. Some models focus on processing each task separately [13], [19], [20] while others chose to both localize and classify actions at once by either transferring features from a localization step to a classifier [21] or by doing both at once [22], [23]. Interestingly, numerous methods ignore the localization task as a whole [24], [25], [26]. This is most likely due to the structure of action classification datasets, with some focusing more on global activity recognition in a clip [27]. These action recognition models are incompatible for autonomous vehicles as they are not robust against untrimmed, unpampered video streams.

As a safety-critical system, an autonomous vehicle should have explainable software modules that can assure the confidence of all stakeholders during widespread adoption. For the sake of developing deep neural models that are compatible and explainable with modular software in safety-critical systems,
we focus our attention on designing a step-by-step approach to action localization and classification. Our related works reflect this.

**B. Human Action Localization**

Action localization, or action proposal generation, consists of finding the temporal boundaries and spatial bounding boxes in which an active object exists.

Most attempts at action localization focused purely on temporal-only localization using convolutional neural networks [28], [29] and LSTMs [30], [31]. These methods often followed an expensive sliding window approach [32], [33], [34].

Recent spatial-temporal localization works consist of supervised [35], [36], semi-supervised [37], and weakly supervised approaches [38]. These solutions use sliding-window techniques, with some increasing their accuracy offline. Some works also focused on building action tubelets incrementally [22], allowing for better use in real-time systems. To make our system usable online, our proposed action localization scheme follows a real-time strategy.

**C. Optical Flow Estimation**

Estimating optical flow is a difficult problem which was traditionally posed as the optimization of a global energy function [39]. This optimization problem was solved using continuous [40] and discrete optimization algorithms [41]. Empowered by CNNs, estimating optical flows has become faster and robust using a simple two stream network, such as the FlowNetv2 architecture [42]. The FlowNetv2 design was improved upon by the RAFT [15] architecture which introduced the use of a recurrent operator that iteratively updates the flow field.

Estimated optical flow, and other semantic abstractions of images, can be fused with raw RGB frames for various detection and segmentation tasks [43], [44], [45], [46], [47]. Rashed et al. [43] studied the use of optical flow with RGB frames for the task of semantic segmentation. The authors compared four different architectures and found that the two-stream architecture with deep fusion of RGB and optical flow resulted in better-overall performance. We take inspiration from their approaches and adapt them for active agent detection.

**D. Action Recognition for Autonomous Vehicles**

In contrast to human action, action recognition for autonomous vehicles is an under-explored field. The ROad event Awareness Dataset (ROAD) [1] was the first to apply the concept of action classification to road agents, and a limited number of methods have been applied to this domain so far. In their description of the winning entries of different subtasks of the challenge, the ROAD authors emphasize that most performance improvements result from improvements in post-processing steps [1]. This seems counter-intuitive for a dataset meant to cater action recognition for autonomous vehicles, which requires an accurate, online solution to action localization and classification.
As an example, Argus++ [48] proposes a tube proposal generation scheme that oversamples fixed length action tubes and subsequently deduplicates them. In contrast to these post-processing-focused approaches, we choose to constrain ourselves to online approaches only.

III. METHODOLOGY

In the following section, we provide a detailed explanation of the proposed RALACs system architecture illustrated in Fig. 2.

A. Overall Description of Framework

RALACs is a two-part system which first performs active agent object detection followed by per-agent action classification. The inputs to the system are sequences of RGB images, which may be streamed live in an AV setting, and the outputs are object bounding box proposals with associated action proposals. Active agent detection is performed using deep fusion of RGB images and optical flow maps at multiple feature scales, which helps the model learn to differentiate between active and inactive agents. Action classification uses a history of bounding boxes, called tubes, to make predictions from both appearance and change over time. It leverages dynamic agent and scene features in interaction encoding to understand context of agents relative to each other and their environment. An object tracker links the two stages together to produce the tubes from the detection module.

B. Action Localization With Optical Flow

Action localization consists of finding the spatial-temporal boundaries in which an active agent occurs. Compared to traditional object recognition, action localization systems must differentiate between agents with and without an action. This reduces unnecessary processing and the chances of misclassification of static road agents (e.g., parked cars).

Our approach focuses on the hypothesis that the relevance, or activeness, of an agent is related to its motion relative to the environment. We also follow an incremental, frame-based methodology to make our system implementable in real time.

For an arbitrary memory bandwidth of \( l \) RGB frames back from current time, our action localization system produces a list of \( n \) active agent tubelets \( T = \{T_1, \ldots, T_n\} \). Each of these tubelets, \( T_i \), contains the class of its active agent \( c_i \) and that agent’s bounding-boxes for each of the \( l \) frames \( \{b_{i,0}, \ldots, b_{i,l}\} \). Since an active agent is subject to occlusion, and can disappear at any time, an active agent tubelet may have no bounding-box for some of the frames.

1) Optical-Flow-Based Active Agent Detection: In this section, we introduce a framewise approach to active agent detection using optical flow. To include motion information, we propose a modified, two-staged object detector which considers both the optical flow and RGB to make predictions.

Optical Flow Estimation: We generate optical flow maps using an online optical flow estimator, such as RAFT [15]. RAFT contains several good attributes that make their network viable for autonomous vehicle applications. These include: 1) strong generalization ability based on the author’s tests with synthetic data [15] and 2) fast inference speed of around \( \sim 20 \) fps on an RTX 3090. Because of these attributes, we chose to use RAFT for generating our optical flow maps.

A persistent challenge with optical flow estimation for autonomous vehicles is that camera data is not coming from a stationary source. Consequently, we end up with noisy flow fields containing not just the motion of active agents but also stationary objects passing by as well.

This issue can be addressed in two ways.

1) Filtering of flow vectors representing the flow magnitude of stationary objects. This would require hard-coded rules.

2) Leaving the flow field as is.

Although filtering stationary flow may seem promising, the process in which to filter it out changes according to the motion of the ego vehicle. For example, when the ego vehicle is moving straight ahead, flow vectors of stationary objects diverge from the center of the frame. In contrast, when the ego vehicle is turning, the stationary flow field may have little to no divergence at all. Instead, most of the stationary flow vectors are facing a direction opposite to which the ego vehicle is turning. There are many other cases that would cause the filtering process to change, so instead of relying on global heuristics, we intend to let our detection neural network implicitly learn to discount stationary flow using the inputted flow map and the RGB frame.

Fusion of Optical Flow With RGB: Our two-stream object detector must encode two data sources: 1) the current RGB frame of size \( h \times w \times 3 \) and 2) the optical flow map, of size \( h \times w \times 2 \), between the current frame and its previous. Because the optical flow map is different in size, we instead use a three-channel, color-wheel representation of optical flow. This can be done by normalizing the magnitudes of the flow vectors and mapping them to an image of a color wheel.

With these two sources, we propose a deep fusion, modified feature pyramid network (FPN) approach to encoding optical flow and RGB for frame-level, active-agent detection.

We start by forwarding the optical flow map and RGB images separately into two different ResNetXt backbone networks. This produces multiple feature maps for each data stream, denoted as \( F_O = \{F_{O_1}, F_{O_2}, F_{O_3}, F_{O_4}, F_{O_5}\} \) for the feature maps of optical flow and \( F_R = \{F_{R_1}, F_{R_2}, F_{R_3}, F_{R_4}, F_{R_5}\} \) for the feature maps of RGB. Note that the dimensions of \( F_{O_i} \) and \( F_{R_i} \), where \( i \) is the feature level, are the same. Following feature mapping, \( F_{O_i} \) and \( F_{R_i} \) are then fused using a summation junction in the FPN to form (1), or the final FPN feature scale for the \( i \)th level. Each \( F_i \) is then fed into a Faster R-CNN prediction head, producing \( n \) post-NMS detections for a single frame \( D = \{[c_1, b_1], \ldots, [c_m, b_m]\} \) where \( c_i \) is the class of the agent with its confidence score and \( b_i \) is its bounding box. We use a low-NMS threshold on the list of detections to increase the number of candidates for downstream tracking

\[
F_i = F_{O_i} + F_{R_i} + \text{UpSample}(F_{O_{i-1}}) + \text{UpSample}(F_{R_{i-1}}).
\]

During training, the use of a summation junction makes backpropagation simple and intuitive. Since we use the three-channel RGB color-wheel representation of optical flow, we
can use the same initial pretrained weights for both backbones. This was found to be better than starting the optical-flow backbone from scratch.

2) Proposal Filtering and Linking: Given a new frame of detections $D_1$ and $l - 1$ frames of past detections $\{D_2, D_3, \ldots, D_l\}$, an online object tracker is used to stitch the detections of $D_1$ onto preexisting tubelets $T$ formed by the past frames. This is done by assigning some detections with a track ID $t_{\text{in}}$ while discarding others. This turns the list of detections $D_1 = [(c_{1,1}, b_{1,1}), \ldots, (c_{1,m}, b_{1,m})]$ into tracked detections of $n$ active agents $D_1 = [(c'_{1,1}, b'_{1,1}, t_{\text{in}}), \ldots, (c'_{1,n}, b'_{1,n}, t_{\text{in}})]$. Detectors without an ID can also form a new tubelet should they overlap with other nontracked detections three frames in a row.

We adopt OC-SORT’s [49] tracking scheme to link agent detections into tubes. OC-SORT contains several good attributes that make their algorithm viable for autonomous vehicles. These include: 1) fast, online approach to object tracking and 2) robust tracking against occlusions and non-linear paths. In order to eliminate the possibility of different object classes in the same track, we separately track agents from different agent classes.

C. Action Classification With Interaction Encoding and Tube Priors

Encoding Tubelets With DROI-Alignment: Given agent tube proposals, the action classifier considers sliding windows of $l$ frames as clips. For every clip, it selects the center frame or key-frame (indexed $l/2$) and uses information from the temporal extent of $l$ frames to produce a confidence vector $\{c_{l/2,j}^1, c_{l/2,j}^2, \ldots, c_{l/2,j}^n\}$ where $n$ is the number of action classes.

Our approach adapts ACAR-Net [13] to accept the position of a keyframe object $j$ throughout the clip $T_j = [b_{1,j}, \ldots, b_{l,j}]$ where $j$ must exist in the key-frame and every $b_{t,j}$ has the same track id. To localize an object’s features in a specific frame, we modify the use of ROI-align in ACAR-Net. Originally, ACAR-Net performs ROI-alignment across $\text{TemporalAvgPool}(F_f)$ and $\text{TemporalAveragePool}(F_s)$ using the bounding box $b_{l/2,j}$ of the object present in the key-frame. This design was justified by the original ACAR-Net authors due to the limited movement of agents in the Kinetics Dataset. Particularly in human activity detection in datasets, it can be observed that agents maintain relatively consistent spatial locality throughout frames.

In contrast to the human activity datasets, ROAD agents include vehicles which move dynamically across spatial regions in frames. As a result, an object’s track throughout a clip often drastically overreaches the exact bounding box of its position at the key-frame. If, within a clip, an object moves out of its bounding box in the key-frame, then ACAR-Net would no longer be encoding information about the object but unrelated artifacts in the background. This could cause unwanted noise and loss of crucial information.

In order to consider Tube Priors, RALACs implements a novel ROI-alignment configuration: DROI alignment. Instead of aligning the key-frame bounding boxes on a temporally pooled feature tensor, we first align the bounding boxes of each frame to its corresponding temporal step in the feature tensor whenever possible and only then pool them. We note that the SlowFast backbone allows us to select feature tensors corresponding to each frame for this purpose.

Suppose for an input of $l$ frames we have a set of temporally indexed fast and slow feature tensors $F_f = \{f_0, f_1, f_2, \ldots, f_{l-1}\}$ and $F_s = \{s_0, s_1, s_2, \ldots, \}$ from SlowFast. Here, a slow feature exists every $\alpha$ frames where $\alpha$ is the ratio of fast features to slow features. A set of an agent’s temporally indexed bounding boxes $bb = [b_0, b_1, \ldots, b_{l-1}]$ also exists from the agent tube priors. We apply ROIAlign on each $f_i$ and $s_i$ with the bounding box $b_i$ where $i$ is the temporal frame to extract ROI feature maps $roi_i$ and $roi_j$. We then finally average all ROI feature maps across the temporal dimension as shown in Algorithm 1. Since the ROI features are extracted from each frame, the resulting $F'_{f}$ and $F'_{s}$ DROI-aligned actor feature maps are therefore encoded with the dynamic bounding box of the agent. This algorithm has an $O(n)$ complexity where $n$ represents the number of bounding boxes for all tracks in a clip.

Encoding Road Agent Interactions: Although the current DROI-Aligned feature maps are sufficient enough to begin classification, our work proposes an extension of ACAR-Net’s [13] higher-order relations reasoning to further improve accuracy. Our approach focuses on the hypothesis that actions of a road agent are largely the result of their interactions with other agents and that these interactions can be learned by a neural network.

The idea of capturing relations between agents and the environment has been brought up in various research fields [13, 50]. In infrared, such mechanisms encoded semantic features between a scene’s low-level and high-level representations to improve detection performance. More specific to human action recognition, capturing higher-order
relations was proposed by the authors of ACAR-Net. In their work, the encoded features of human agents, each tiled and concatenated with the features of the overall clip and then reduced using a convolutional layer, were subjected to a modified transformer-based dot-product attention mechanism they called HR$^2$O. To explain this mechanism, given feature maps of $N$ human agents interacting with their environment, $\{F_i\}_{i=1}^N$, their resultant agent–context–agent feature maps $\{H_i\}_{i=1}^N$ are obtained by first converting each $F_i$ into key $K_i$, value $V_i$, and query $Q_i$ embeddings

$$Q_i, K_i, V_i = \text{Conv2D}(F_i).$$

An $\tilde{H}_i$ is computed through a linear combination of all other agent value, $V_j$, features according to their computed attention weight, $A_{i,j}$

$$A_{i,j}^{(x,y)} = \text{softmax}\left(\frac{(Q_i^{(x,y)}, K_j^{(x,y)})}{\sqrt{d}}\right)$$

$$\tilde{H}_i^{(x,y)} = \sum_j A_{i,j}^{(x,y)} V_j^{(x,y)}.$$

$\tilde{H}_i$ is then subjected to layer normalization and dropout to become $H_i$

$$H_i = \text{Dropout}(\text{Conv2D}((\text{ReLU}(\text{norm}(\tilde{H}_i)))).$$

$H_i$ is then combined with $F_i$ through residual addition to become the final feature map $F'_i$

$$F'_i = F_i + H_i.$$ (6)

It is important to note that the attention mechanism is only applied to feature maps along the same spatial location, reducing computation time and limiting HR$^2$O to first-order interactions only. This means that the features of $F_i$ at the spatial location $(x,y)$, denoted as $F_i^{(x,y)}$, are cross-examined only with the other feature maps at the same spatial location, $F_j^{(x,y)}$.

Once all $\{F'_i\}_{i=1}^N$ are obtained, each agent–context–agent feature map is passed into a fully connected layer with a nonlinearity function to produce the final action confidence scores of each agent.

In ACAR-Net, all feature maps entering their interaction encoding, HR$^2$O, are constrained to human agents only. In RALACs, we extend its functionality in our Interaction Encoding algorithm to instead populate $\{F'_i\}_{i=1}^N$ with the feature maps of multiple different agent classes. Since the feature maps of all agent classes have the same dimension, the HR$^2$O interaction can be used to motivate the model to implicitly learn interactions between all road-relevant classes. During back propagation, the gradient for a specific agent’s feature map will be derived from all other DRAligned agent feature maps, along with the environment feature map. This enables robust action classification of multiclass agents in complex scenes.

Given the produced set of confidence vectors for $N$ agents $X = \{C_0, C_1, \ldots, C_{N-1}\}$, where $C_j$ contains the confidence of $n$ actions for the $j$th actor $C_j = \{c_{j1}, c_{j2}, \ldots, c_{jn}\}$. We apply a sigmoid cross-entropy focal loss [51] to the predicted action confidence vector against the ground truth action vector $Y_j = \{y_{j1}, y_{j2}, \ldots, y_{jn}\}$ for training

$$L_f(p_t) = -\alpha (1 - p_t)^\gamma \log (p_t)$$

$$g(p, y) = \begin{cases} p, & \text{if } y = 1 \\ 1 - p, & \text{otherwise} \end{cases}$$ (8)

$$L_{\text{action}} = \sum_{j=0}^{N-1} \sum_{i=1}^n L_f(g(c_j, y_j)).$$ (9)

The focal loss helps overcome the severe class imbalance of the action class distributions in the ROAD Dataset, in which there is a larger majority of vehicle and pedestrian classes present in the training data as opposed to cyclists, emergency vehicles, etc.

IV. EXPERIMENTATION

A. ICCV2021 ROAD Challenge

We focus our performance evaluations and ablation studies using the ICCV2021 ROAD Dataset [1]. Being the first dataset of its kind, the ROAD Dataset consists of 22 videos of annotated road actions on a moving ego vehicle. These videos are each 8-min long and recorded at an average of 12fps. The result is over 120K frames with 640K annotated actions spread throughout. ROAD no longer accepts test set submissions, so we base our experimental validation solely on its publicly available training and validation data.

The ROAD dataset is organized into 3 splits, each of which represents different sets of validation scenarios. Splits 1, 2, and 3 validate on overcast, nighttime, and sunny scenarios, respectively.

The ROAD dataset provides us with thorough road-relevant action labels which form the basis of all experimental analysis in this article. Because environmental modeling on autonomous vehicles have traditionally relied on object tracks, our suggestion is to maintain the integrity of object tracks and have its framewise action appended as an attribute instead. This approach not only enables the ease of integration on existing autonomy stacks, but also provides room for future attribute expansion. Nevertheless, we note that the use of action tracks for evaluation is limited and we only use this metric to compare with the baseline model. All other experimental metrics are done at the frame level.

1) Action Track Post-processing: To compare our results to the existing baseline and evaluate our end-to-end performance using the ROAD dataset’s official evaluation scheme, our framewise action detections must be processed to form action tracks. We perform tube trimming to process the final actions into tube proposals that cater to the official ROAD dataset evaluation. This can be implemented as an offline or online approach.

Tube Trimming: We solve the problem of action track specificity by trimming agent tubes that already exist. Suppose we have frames in an agent’s tube $T = \left\{ f_{ik}, \ldots \right\}$ with confidence scores $s_{ik}$ for class $k$. Intuitively, we would like to partition $T$ into ordered subsets $T = \{F_a, F_b, F_c\}$ where for some threshold $\varepsilon$, $s_{ik} < \varepsilon$ when $s_{ik} \in F_a \cup F_c$. This operation would isolate the action in $F_b$ and remove noise and other
actions belonging to the same agent in $F_a \cup F_c$. We note, however, that this is not possible in online contexts, so for these cases we simply apply a mask over the frame scores based on the parameter $\varepsilon$, typically set to a value $\leq 0.001$.

2) Implementation Details: Action Localization: We use faster R-CNN with a modified FPN [52] neck and ResNetXt-101 backbone as the object detector. We pretrained the model on COCO with a single RTX 3090 and duplicate its backbone for both optical flow and RGB channels. The backbones of both channels are jointly trained with the detection head. We train the detection model end-to-end for seven epochs ($\sim$140k steps) with a batch size of 16. To stitch these detections into tracks, OCSORT with an IOU of 0.3, and Kalman filter parameters $\delta t = 3$ and $\text{inertia} = 0.2$, were used.

Inactive Agent Pseudo Annotations: The ROAD dataset only annotates active agents (i.e., a parked on the side of the road is not considered to be “active,” but a stopped vehicle at an intersection is), and the lack of explicit annotations for inactive agents generates noisy learning targets for the object detector which could degrade performance. To solve this problem, we propose to generate pseudo-labels for the inactive agents on the ROAD dataset. Since most of the inactive agents are vehicles, we only consider them in our method. First, we train the Faster R-CNN FPN [52] with ResNeXt-101 as backbone on the COCO dataset [53], and we use it to predict vehicles in the ROAD dataset. After obtaining the vehicle detections, we filter for detections which have a confidence score of 0.9 or higher and an intersection over union of less than 0.2 with the ground truth ROAD detections. These detections are then relabeled as a new “inactive agent” class. We use these detections, along with the original annotations, to train our final active agent detector. We found that this approach yields better results than training our detector directly on the data.

Action Classification: We use officially released SlowFast weights pretrained on the Kinetics-400 dataset [54]. We allow these weights to be fine-tuned during training, and do not use any pretrained weights for the interaction encoding block. We observe our best results on average after training for $\sim$140k steps or 7–9 epochs on dual RTX 3090s with a batch size of 8. We use a sigmoid cross-entropy focal loss with $\alpha = 0.25$ and $\gamma = 2$ to overcome the severe class imbalance of the action class distributions in the ROAD Dataset. We use a base learning rate of $8e^{-4}$, which decreases by a factor of 10 by epochs 4, 6, and 7. We perform linear warm-up during the first epoch at a learning rate of $6.4e^{-2}$, and use a weight decay of $1e^{-5}$ with Nesterov momentum of 0.9.

The full end-to-end model has 243M parameters. In terms of computational complexity, the system achieves 15 fps and fully utilizes a RTX 3090 GPU and 2 GB of VRAM. The system was tested on the on-board computer in the Chevrolet Bolt used in the SAE AutoDrive Challenge, which has one RTX 3090 GPU and an AMD Ryzen Threadripper PRO 3975WX. CPU usage was minimal, utilizing only a single core at 5% CPU usage. Experimentally, we observed a fluctuation of $\sim$1% when varying our model’s rate of convergence and batch size.

B. NIST ActEV Challenge

To demonstrate ability to generalize, we evaluate RALACs with the same configuration on the NIST Activities in the Extended Videos (ActEV) challenge. The ActEV challenge uses the multiview extended videos (MEVA) dataset [55], which contains action labels of pedestrians and cars observed stationary security cameras. The dataset format is very similar to ROAD, so we used the same training scheme and hyperparameters.

C. Full Model Evaluation

We evaluate our proposed method against the current baseline model, 3-D-RetinaNet, as it is the only model with comprehensive validation metrics. Table I shows our results on the ROAD dataset. We see that RALACs’ online performance outperforms the current baseline.

Breaking down the evaluation, we see that our overall approach functions considerably better in ideal overcast scenarios. Val 1 is characterized by little weather-related faults, such as rain and glare from vehicles and the sun. In contrast, Val 2 and Val 3 have a considerable number of these faults, and our performance consequently has room for improvement. We also note that Split 2 does not have any training data at
TABLE I

| Frame-Level Eval | Val 1 | Val 2 | Val 3 | Avg  |
|------------------|-------|-------|-------|------|
| 3D-RetinaNet (frame-level) | 26.2  | 11.7  | 21.2  | 19.7 |
| Ours (frame-level) | 31.9  | 12.4  | 22.9  | 22.4 |

| Video-Level Eval | Val 1 | Val 2 | Val 3 | Avg  |
|------------------|-------|-------|-------|------|
| 3D-RetinaNet     | 17.0  | 11.4  | 14.6  | 14.3 |
| Ours (Online)    | 22.7  | 11.5  | 15.4  | 16.5 |
| Ours (Offline)   | 23.6  | 11.7  | 16.1  | 17.1 |

TABLE II

| System         | $P_{miss}@0.1R_{fa}$ | nAUDC@0.2R_{fa} |
|----------------|-----------------------|-----------------|
| Ours           | 0.5311                | 0.5893          |
| BUPT-MCPRL     | 0.5781                | 0.6193          |
| UMD-JHU        | 0.8131                | 0.8300          |
| MLVC-HDU       | 0.8952                | 0.9040          |

From Table III, we see that Input Fusion yields underwhelming results compared to a Faster RCNN trained on pseudo annotations. This could be the result of a multitude of problems, one of which could be the lack of a pretrained backbone.

In contrast, when Optical Flow is encoded and fused at multiple feature scales in the FPN, we get better results than the RGB-only approaches. Fig. 3 provides an example of this. We hypothesize that initially encoding the optical flow and RGB channels implicitly instructs the network to first localize objects and their respective flow field. These localizations are then cross-compared at the summation joint to determine its active/inactive state.

Our results also show that penalizing a pretrained detector for localizing a nonactive agent is not as effective as penalizing it for misclassification. By retraining the detector to detect agents as “inactive” instead of not detecting them at all, we observe faster convergence.

Action Classification Ablation: To examine the impact of our contributions to action classification for road agents, an in-depth, classwise ablation analysis was conducted. Fig. 4 shows various performance metrics of different road action classes. The differences in average precision between classes correlate directly with the frequency of each class in the training data.

Interaction Encoding: Naturally, certain actions depend more on cross-agent interactions than others. In Fig. 4, we see a very similar pattern, where actions like Braking and Waiting to Cross (Wait2X) both benefit significantly from the addition of Interaction Encoding. Both of these actions heavily rely on interactions; as Braking requires some other agent to brake for, and agents Wait2X are waiting for other agents to cross.

Furthermore, we see that Interaction Encoding also provides a general benefit to the precision of all action classes. We attribute this to the model’s increased awareness of action context. That is, the model can now consider the actions of an agent in context with the actions happening around it. This is especially true in interaction-heavy scenes like intersections, as shown in Fig. 5.
Fig. 4. Analysis of action classification improvements by action class. We see that the use of interaction encoding improves the classification of actions that naturally require more interactions with other agents (see braking and crossing (X) actions). Furthermore, DROI alignment is also shown to be an effective method for extracting the information present in agent tracks, especially during actions of large movements.

Most importantly, this overall improvement using Interaction Encoding shows how our network not just successfully learned interactions but interactions that are agent-agnostic. This finding alone shows promise for future interaction encoding schemes, and a possible avenue for future research.

DROI Alignment: Based on our experiments, the addition of DROI Alignment provides an overall benefit to all action classes. We attribute this improvement to the successful encoding of object tracks. Fig. 6 shows a comparison between the data encoded by traditional ROI Alignment as opposed to our DROI strategy. The traditional frame-based input ROI Alignment strategy uses a static bounding box from the middle keyframe, and does not take into account the dynamic spatial movement of the agent between frames. Our proposed tube-based input encodes the dynamic bounding box between frames into the feature map based on tracked agent tubes using Algorithm 1. Our experiments show that the model is not only capable of handling tube representations of an object’s track but also extracting crucial information from them for classifying action.

To better analyze the effects of DROI Alignment as opposed to other improvements, our method specifically improves the ability of our model to classify action classes with higher-dynamic change. In Fig. 4, the addition of DROI Alignment provides significant improvements to classifying actions like turns, crosses, and movement direction. We observe that these actions naturally require more information on trajectory, so it makes sense that the inclusion of tube-based alignment provides our network with an improved ability to detect them.

E. Deployment of RALACs to Research Vehicle

To enhance the scene understanding capabilities of the WAtonomous research platform [14]—and provide preliminary insight into the effects action classification can have in autonomous vehicle decision making—RALACs was deployed
which they are used as input to behavioral planning decisions. For each frame environment model.

Input: Frame timestamps, observed bboxes \(\triangleright\) both sorted from most recent to oldest

Output: For each frame timestamp, finds the temporally closest bbox

1. \(\text{match}_i \leftarrow 0\) \(\triangleright\) Index of bounding box to check match
2. \(\text{matched_bbox} \leftarrow []\)
3. for \(s \in \text{timestamps}\) do
4.     while TRUE do
5.         \(\text{match}_\delta \leftarrow \text{abs}(s - \text{bbox}[\text{match}_i].s)\)
6.         \(\text{next_match}_\delta \leftarrow \text{abs}(s - \text{bbox}[\text{match}_i + 1].s)\)
7.         \(\text{match}_i \leftarrow \text{match}_i + 1\)
8.         if \(\text{match}_\delta < \text{next_match}_\delta\) then \(\triangleright\) Next time delta is greater than current time delta
9.             BREAK
10.         end if
11.     end while
12.     \(\text{matched_bbox}.\text{append}(\text{bbox}[\text{match}_i - 1])\)
13. end for
14. return \(\text{matched_bbox}\)

Algorithm 2 Detection—Frame Buffer Time Sync

Fig. 7. Data pipeline used to deploy RALACs to the perception stack. The resulting action classified tracks are sent to the ADS environment model [14] in which they are used as input to behavioral planning decisions.

![Data pipeline used to deploy RALACs to the perception stack.](image)

In brief, the history of 2-D detections is added to each track. The detections are then time synchronized (see Algorithm 2) with a buffered temporal view from the camera stream. Both the temporal view and the tube of observation are then run through RALACs. The result is a binary classification for each class, and the class with the highest score (if it is over 0.5) is added to the action classification history for that track.

Fig. 8 presents the qualitative results of the deployment, showing the input and output of the deployed action classification model. As seen in the figure, the spatial footprint is tracked across the temporal view to provide a better-visual description of the Region of Interest (ROI), as discussed in Section III-C.

Table IV presents the quantitative results of the deployment, comparing the keyframe-based model against the tube-based model. As seen in the table, DROI Alignment produces a correct classification of stopped the majority of the time, whereas a simple key-frame alignment scheme misclassifies the majority of the time. This can be attributed to the poor ROI input to the action classification model, as analyzed before in Section IV-D.

Table IV: Ratio of Classifications Over the Test Scenario. As Shown in the Table, DROI-Alignment Classifies the Agent Correctly for a Longer Period of Time. Without the Information of the Pedestrian’s Track, the Use of Keyframe Alignment Produces More False Classifications. This Movement Close to the Lane Graph Would Trigger Our Ego Vehicle to Stop Out of Uncertainty

| Model       | Stop | MoveTow | MoveAway |
|-------------|------|---------|----------|
| Keyframe    | 0.15 | 0.00    | 0.85     |
| DROI        | 0.54 | 0.26    | 0.20     |

1) Test Scenario: Real-world testing was carried out at the Waterloo Region Emergency Services Training and Research Centre (WRESTRC). In this work, we present the effects of action classification on an edge-case scenario previously tasked to the WATonomous team by the organizers of the SAE Autodrive Challenge. As shown in Fig. 9’s lane graphs, the pedestrian is close enough to the road such that our perception system is unsure as to whether or not the pedestrian is on the road or not. Our vehicle’s task is to proceed forward after determining that the pedestrian will not interfere with our path despite being so close to the road.

2) Perception Integration: In order to deploy RALACs to the WATonomous autonomous driving stack (ADS), the networks inputs need to be generated in a real-time manner, which is achieved by the data pipeline in Fig. 7.

In brief, the history of 2-D detections is added to each track. The detections are then time synchronized (see Algorithm 2) with a buffered temporal view from the camera stream. Both the temporal view and the tube of observation are then run through RALACs. The result is a binary classification for each...
Fig. 8. Visual and RoI inputs to the ACAR model, as well as the top classification score across the entire tube, and the temporal misalignment for the frame. (a) First frame in the view given to ACAR. (b) Middle (keyframe) in the view given to ACAR. (c) Last frame in the view given to ACAR.

Fig. 9. Comparing different ego behavior in the stopped pedestrian scenario. (a) Overly conservative behavioral decision to stop given no visual indication that the pedestrian will interfere. Occurs when geometric heuristics specified in [14, Algorithm 2] are used. (b) Reasonable behavioral decision to drive past a stopped pedestrian, enabled by accurate action classification as specified in Algorithm 3.

Algorithm 3 Handle Pedestrian With Action

Input: Tracked ped with history of actions, history time \( t \) to consider, set of interference actions \( \text{int}_\text{set} \).

Output: Boolean indicating if the pedestrian should be inserted into the conflict graph.

1: \( n_{\text{interfere}} \leftarrow 0 \) ▶ Counter of interference actions
2: \( n_{\text{total}} \leftarrow 0 \) ▶ Counter of total actions considered
3: for action \( \in \text{ped.action_hist} \) do
4: \( \text{now} \leftarrow \text{time.now()} \) ▶ Get current time
5: if \( \text{now} \! - \! \text{action.stamp} > t \) then
6: BREAK
7: end if
8: \( n_{\text{total}} \leftarrow n_{\text{total}} + 1 \)
9: if action \( \in \text{int}_\text{set} \) ▶ If action is interference
10: \( n_{\text{interfere}} \leftarrow n_{\text{interfere}} + 1 \)
11: end if
12: end for
13: return \( n_{\text{interfere}}/n_{\text{total}} > 0.5 \) ▶ Interfere threshold

The action classification history helps to limit these false positive cases by classifying the intentions of pedestrians based on their and the scene’s appearance descriptions, rather than their geometrical descriptions on the lane graph. In Fig. 9 we see an example of this, comparing the resulting relationship graph under the geometrical implementation, versus the graph under the action classification implementation. The difference in the conflict creation algorithm is presented in Algorithm 3, which determines whether a pedestrian’s action history interferes with the route. This is done via a simple threshold of the ratio of actions which fall within an interference set \( \text{int}_\text{set} \) to total actions within a time history \( t \). In the test scenario \( \text{int}_\text{set} = \{ \text{Stop}, \text{Wait2X} \} \) and \( t = 5 \) s were used.

Overall, this scenario provides important insight into the usefulness of action classification when lane graph heuristics fail. Without information that the pedestrian is perfectly stationary, our vehicle can never make a confident decision to proceed forward. In future works, we hope to extend our experiments to encompass more scenarios and edge cases.

V. Conclusion

In this work, we develop methods for online action classification in road settings that focus on the identification of active agents and the use of agent-action contexts to determine their actions. We demonstrate that our contributions and proposed architecture can surpass the baseline performance of 3D-RetinaNet on the ROAD dataset and we tackle the deployment challenges of using such a system on a real vehicle platform.

Toward the future, we welcome more datasets that set the standard for action recognition, by including more varied and better-defined agent actions. In addition, we believe that future efforts should explore the uses of action classification in decision making, and provide further analysis of its benefits. We believe that it may also be possible to achieve additive performance benefits from existing approaches by combining our contributions with existing post-processing ideas that work...
well, such as the de-duplication approach from the Argus++ authors.

ACKNOWLEDGMENT

The authors would like to thank Chuan Tian (Ben) Zhang for his early ideas and the WA TNomous Server Infrastructure Team whose support made this work possible.

REFERENCES

[1] G. Singh et al., “ROAD: The road event awareness dataset for autonomous driving,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 1, pp. 1036–1054, Jan. 2023.

[2] M. Huang, Z.-P. Jiang, and K. Ozbay, “Learning-based adaptive optimal control for connected vehicles in mixed traffic: Robustness to driver reaction time,” IEEE Trans. Cybern., vol. 52, no. 6, pp. 5267–5277, Jun. 2022.

[3] R. Dempster, M. Al-Sharman, D. Rayside, and W. Melek, “Real-time unified trajectory planning and optimal control for urban autonomous driving under static and dynamic obstacle constraints,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2023, pp. 10139–10145.

[4] A. Petrillo, A. Pescapé, and S. Santini, “A secure adaptive control for cooperative driving of autonomous connected vehicles in the presence of heterogeneous communication delays and cyberattacks,” IEEE Trans. Cybern., vol. 51, no. 3, pp. 1134–1149, Mar. 2021.

[5] M. Al-Sharman et al., “A sensorless state estimation for a safety-oriented cyber-physical system in urban driving: Deep learning approach,” IEEE/CIA J. Automatica Sinica, vol. 8, no. 1, pp. 169–178, Jan. 2021.

[6] C. Xia et al., “Onboard sensors-based self-localization for autonomous vehicle with hierarchical map,” IEEE Trans. Cybern., vol. 53, no. 7, pp. 4218–4231, Jul. 2023.

[7] D. Feng et al., “Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 3, pp. 1341–1360, Mar. 2021.

[8] C. Hu, S. Hudson, M. Ethier, M. Al-Sharman, D. Rayside, and W. Melek, “Sim-to-real domain adaptation for lane detection and classification in autonomous driving,” in Proc. IEEE Intell. Veh. Syst. (IV), 2022, pp. 457–463.

[9] D. Kamale, S. Haesaert, and C.-I. Vasile, “Cautious planning with incremental symbolic perception: Designing verified reactive driving maneuvers,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2023, pp. 1652–1658.

[10] A. Aboah, U. Bagci, A. R. Mussah, N. J. Owor, and Y. Adu-Gyamfi, “DeepSegmenter: Temporal action localization for detecting anomalies in untrimmed naturalistic driving videos,” in Proc. IEEE/CVF Conf. Vis. Pattern Recognit. 2023, pp. 5358–5364.

[11] R. Li et al., “Action probability calibration for efficient naturalistic driving action localization,” in Proc. IEEE/CVF Conf. Vis. Pattern Recognit., 2023, pp. 5269–5276.

[12] L. Zhang, W. Cui, B. Li, Z. Chen, M. Wu, and T. S. Ge, “Privacy-preserving cross-environment human activity recognition,” IEEE Trans. Cybern., vol. 53, no. 3, pp. 1765–1775, Mar. 2023.

[13] J. Pan, S. Chen, M. Z. Shou, Y. Liu, J. Shao, and H. Li, “Actor-context-actor relation network for spatio-temporal action localization,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2020, pp. 464–474.

[14] R. Dempster, M. Al-Sharman, Y. Jain, J. Li, D. Rayside, and W. Melek, “DRDG: A dynamic relation graph for unified prior-online environment modeling in urban autonomous driving,” in Proc. Int. Conf. Robot. Auton. (ICRA), 2022, pp. 8054–8060.

[15] Z. Teed and J. Deng, “RAFT: Recurrent all-pairs field transforms for optical flow,” in Proc. 16th Eur. Conf. Comput. Vis. 2020, pp. 402–419.

[16] B. Sun, S. Wang, D. Kong, L. Wang, and B. Yin, “Real-time human action recognition using locally aggregated kinematic-guided skeleton learner and supervised hashing-by-analysis model,” IEEE Trans. Cybern., vol. 52, no. 6, pp. 4837–4849, Jun. 2022.

[17] Z. Gao, Y. Zhao, H. Zhang, D. Chen, A.-A. Liu, and S. Chen, “A novel multiple-view adversarial learning network for unsupervised domain adaptation action recognition,” IEEE Trans. Cybern., vol. 52, no. 12, pp. 13197–13211, Dec. 2022.

[18] T. Wang et al., “RecAPNet: Action proposal generation mimicking human cognitive process,” IEEE Trans. Cybern., vol. 51, no. 12, pp. 6017–6028, Dec. 2021.
Eddy Zhou is currently pursuing the B.A.Sc. degree in mechatronics engineering with the University of Waterloo, Waterloo, ON, Canada. He leads the Waterloo Autonomous Vehicle Team in the SAE AutoDrive Challenge (Watonomous.ca) (WAtonomous), University of Waterloo. Outside of autonomous vehicle development and research, his research interests lie around achieving artificial general intelligence and its various subproblems like perception, future prediction, and decision making.

Owen Leather is currently pursuing the B.A.Sc. degree in mechatronics engineering with the University of Waterloo, Waterloo, ON, Canada. He is a Researcher with the Waterloo Autonomous Vehicle Team in the SAE AutoDrive Challenge (Watonomous.ca) (WAtonomous), University of Waterloo. He has worked on perception software ranging from spacecraft autonomy to autonomous vehicles. His research interests are computer vision, neural rendering, and 3-D reconstruction.

Alex Zhuang is currently pursuing the B.Sc. degree in computer science with the University of Waterloo, Waterloo, ON, Canada. He has worked on action classification pipelines with the Waterloo Autonomous Vehicle Team in the SAE AutoDrive Challenge (Watonomous.ca) (WAtonomous), University of Waterloo. Outside of self-driving, he has worked on problems ranging from medical NLP to image augmentation and generative AI for several robotics companies. His research interests lie across natural processing and generative models of text and image data.

Alikasim Budhwani is currently pursuing the B.A.Sc. degree in mechatronics engineering studies with the University of Waterloo, Waterloo, ON, Canada. He has been a part of the Waterloo Autonomous Vehicle Team in the SAE AutoDrive Challenge (Watonomous.ca) (WAtonomous), University of Waterloo since 2022 where he has worked on improving autonomy of self-driving cars. His research interests involve reinforcement learning for controls, action classification, and representation learning.

Rowan Dempster received the Bachelor of Computer Science degree from the University of Waterloo, Waterloo, ON, Canada, in 2020, where he is currently pursuing the M.A.Sc. degree in ECE with the Waterloo Autonomous Vehicle Team in the SAE AutoDrive Challenge (Watonomous.ca) (WAtonomous). He has worked on automated driving systems WAtonomous since 2017. His research interests involve graphical environment models for decision making, action classification in video streams, and predictive control schemes. Dr. Dempster is the recipient of the NSERC Alexander Graham Bell Canada Graduate Scholarship.

QuanQuan Li received the Bachelor of Information Engineering degree from The Chinese University of Hong Kong, Hong Kong, in 2015, and the Master of Electrical and Computer Engineering degree from the University of Waterloo, Waterloo, ON, Canada, in 2022. His research interests are deep learning and computer vision, especially object detection, model acceleration, and 3-D.

Mohammad Al-Sharman received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2023. He is a MITACS Postdoctoral Fellow with the RoboHub, University of Waterloo and a Supervisor of the Waterloo Autonomous Vehicle Team in the SAE AutoDrive Challenge (Watonomous.ca) (WAtonomous), University of Waterloo. His research focuses on advancing cutting-edge algorithms in decision making, motion planning, and state estimation to enhance the maneuverability and agility of autonomous systems. His current research interests include autonomous driving, decision-making, perception, reinforcement learning, and computer vision. Dr. Al-Sharman received multiple academic scholarships and excellence awards throughout his undergraduate and graduate studies.
Derek Rayside (Member, IEEE) received the B.A.Sc. degree in systems design engineering and the M.A.Sc. degree in computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 1999 and 2001, respectively, and the Ph.D. degree from the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, in 2010.

He is the Director of Software Engineering and an Associate Professor with the Electrical and Computer Engineering Department, University of Waterloo where he is also a Faculty Advisor (along with William Melek) with the Waterloo Autonomous Vehicle Team in the SAE AutoDrive Challenge (WAtonomous.ca) (WAtonomous).

Dr. Rayside is a member of SAE and ACM, and is a Licensed P.Eng. in Ontario.

William Melek (Senior Member, IEEE) received the Doctoral degree in mechanical engineering from the University of Toronto, Toronto, ON, Canada, in 2002.

He then led the Artificial Intelligence Division, Alpha Laboratories Inc., Toronto. He is a Professor with the Department of Mechanical and Mechatronics Engineering, University of Waterloo, Waterloo, ON, Canada. He also serves as the Director of the Laboratory of Computational Intelligence and Automation, and the Director of RoboHub. He is an expert on robotics, artificial intelligence, sensing, and state estimation. He founded the Laboratory of Computational Intelligence and Automation, University of Waterloo in 2004. He developed Canada’s first industry-ready modular reconfigurable robot (MMR); the state-of-the-art open architecture system is now used in the automotive sector. He has also led the way in designing practical, intelligent and adaptive control architectures for MMRs based on neural networks. Conceptual prototypes have been developed for the nuclear industry in the USA. He holds 12 Canadian and U.S. patents, and his contributions to the manufacturing industry have been featured in the National Post, Globe and Mail, and CBC Television.

Dr. Melek was awarded the Young Engineer Medal of Professional Engineers Ontario in 2006. He is the Past President of the North American Fuzzy Information Processing Society.