Contents

Chapter 1

Action Classification

Acknowledgement: Development of Road Action Detection Network (RAD-Net)V1 was kicked off in Summer 2021 by Rowan Dempster and Chuan Tian (Ben) Zhang. Work continued on RAD-NetV2 in Summer 2022 by Quanquan Li, Eddy Zhou, and Alex Zhuang, supervised by Rowan Dempster.

1.1 Road Action Detection Network V1 (RAD-NetV1)

1.1.1 Introduction

The perception task in automated driving is to understand the scene around the ego vehicle using camera data. What constitutes "understanding" is an on-going conversation, motivated by what types of information are necessary for safe and effective driving decisions to be made autonomously. Historically, the literature has focused on agent detection and classification tasks which operate on single image frames [30]. State-of-the-art algorithms fit to this task, such as Faster R-CNN [24], have seen large increases in performance over the past decade, and have been adopted into driver assistance systems [18]. A major factor in this success is the availability of large scale datasets such as the KITTI vision benchmark [12].

However, as the field of automated driving matures, the perception research community is shifting their attention towards tasks that represent a deeper understanding of the road scene as a whole [25]. A useful task in this regard is action classification. Separating actions requires the classifier to relate spatial and temporal information across frames. Action

classification is also essential for effective autonomous decision making. Especially in urban driving, understanding the intention of other intelligent actors in the scene is crucial to maneuvering safely while not being overly conservative. To this end, the University of Oxford Brookes has recently published the ROad event Awareness Dataset (ROAD) dataset [25], the first dataset to focus on action classification tasks in the road scene domain.

Agent detection is well studied in the road scene domain. Additionally, action classification is well-studied in the human action domain, for example in the AVA ActivityNet Challenge [1]. In this work we combine these algorithms for effective action detection and classification in the road scene domain, proposing a two-stage action detector which we coin the RAD-NetV1. Our contributions are as follows: (1) We found that the action classifiers first developed for the human action domain can also be trained and perform well on road scenes, even though they contain a wider range of intelligent actors (cyclists, vehicles, etc...). (2) Importantly, we have also discovered that transfer learning from large scale human action datasets (such as AVA) significantly improves performance of RAD-NetV1 in the road scene domain. This is evidence that RAD-NetV1 is able to learn a shared neural representation space that is useful to classify actions in various domains, similar to how humans do.

The remainder of this section is organized as follows: Related works including the ROAD tasks and baseline are discussed in Section 1.1.2, RAD-NetV1 and the algorithms used in its two stages are examined in detail in Section 1.1.3, results and comparative findings are presented in Section 1.1.4, and future work is explored in Section 1.1.5.

1.1.2 Related Work

In this section, we review the ROAD dataset, as well as prior work related to the ROAD tasks which motivated our approach discussed in Section 1.1.3.

ROAD Dataset Tasks and Baseline

The ROAD dataset proposes six tasks in the form of a new annotation database for 18 videos selected from the older Oxford RobotCar Dataset [20]. We only discuss the tasks proposed in the ROAD dataset that are relevant to this thesis.

Agent Detection: The agent detection task is to draw a bounding box around each active agent in the scene, and determine the identity class which the agent falls into. Table 1.1 shows the active agent classes, including vehicles, pedestrians, traffic lights, with their

corresponding description. Fig. 1.1 shows the agent class distribution. This task has been well-studied in the literature [38] and there are several useful implementation candidates which can be found on the KITTI vision benchmark leaderboard [29].

Action Classification: Similar to agent detection, except the categories being assigned to are actions, e.g. crossing and overtaking (see Table 1.2 for further details). This task has been studied extensively in the human action domain (see Section 1.1.2). However, ROAD is the first dataset to provide such a wide breadth of action labels for road scenes.

Agent-Action (Duplex) Detection: A combination of the agent detection and action classification tasks, where active agents must be pixel-localized, and have their identity and action categories classified. The agent-action detection task is the main task we work on in this chapter, with the goal of improving on the baseline given in [25].

Alongside the dataset and these tasks, the authors also published a baseline algorithm, coined 3D-RetinaNet. 3D-RetinaNet employs a 3D-CNN architecture by inflating 2D convolutional layers trained on ResNet50 as described in [32]. Focal loss was also used, as introduced in [19] to address the imbalance between foreground and background proposals while training. In order to track detections from 3D-RetinaNet over multiple frames, forming action "tubes", the authors used the incremental approach presented in [3] which relies on solving a linear program to match new detections to existing tubes.

Table 1.1: ROAD active agent classes with description.

Label name	Description
Autonomous-vehicle	The autonomous vehicle itself
Car	A car up to the size of a multi-purpose vehicle
Medium vehicle	Vehicle larger than a car, such as van
Large vehicle	Vehicle larger than a van, such as a lorry
Bus	A single or double-decker bus or coach
Motorbike	Motorbike, dirt bike, scooter with 2/3 wheels
Emergency vehicle	Ambulance, police car, fire engine, etc.
Pedestrian	A person including children
Cyclist	A person is riding a push/electric bicycle
Vehicle traffic light	Traffic light related to the AV lane
Other traffic light	Traffic light not related to the AV lane

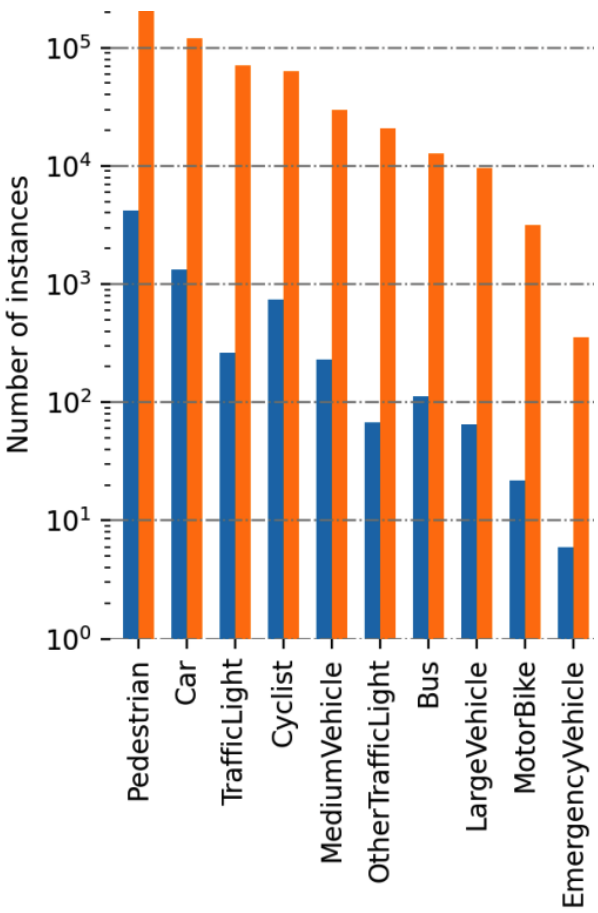


Figure 1.1: Instance counts of various agent classes as specified in [25]. Shows an 1000x class imbalance for some agent classes (Pedestrian vs. Emergency Vehicle). Dealing with this imbalance is discussed in 1.2.1.

Classification of Human Actions

Datasets

Although the ROAD dataset is the first of its kind to focus on road scene action classification, there exist many long-standing datasets covering the human action domain. These include Moments in Time [21] (three second, single-event videos), AVA [14] (untrimmed 15 minute videos, spatio-temporal action labelling, multiple actions per frame), and Kinetics [17] (trimmed 10 second, single-event videos), among others. The AVA dataset and task specification is most closely fit to the ROAD tasks, and is widely used as a benchmark to

Table 1.2: ROAD action labels, with description.

Class name	Description
Moving away	Agent moving in a direction that increases the distance between Agent and AV.
Moving towards	Agent moving in a direction that decreases the distance between Agent and AV.
Moving	Agent moving perpendicular to the traffic flow or vehicle lane
Reversing	Agent is moving backwards.
Braking	Agent is slowing down, vehicle braking lights are lit.
Stopped	Agent stationary but in ready position to move
Indicating left	Agent indicating left by flashing left indicator light, or using a hand signal.
Indicating right	Agent indicating right by flashing right indicator light, or using a hand signal.
Hazard lights on	Hazards lights are flashing on a vehicle.
Turning left	Agent is turning in left direction
Turning right	Agent is turning in right direction
Moving right	Moving lanes from the current one to the right one.
Moving left	Moving lanes from the current one to the left one.
Overtaking	Agent is moving around a slow-moving user, often switching lanes to overtake
Waiting to cross	Agent on a pavement, stationary, facing in the direction of the road.
Crossing road from left	Agent crossing road, starting from the left and moving towards the right of AV.
Crossing road from right	Agent crossing road, starting from the right pavement and moving towards the left pavement.
Crossing	Agent crossing road.
Pushing object	Agent pushing object, such as trolley or pushchair, wheelchair or bicycle.
Traffic light red	Traffic light with red light lit.
Traffic light amber	Traffic light with amber light lit.
Traffic light green	Traffic light with green light lit
Traffic light black	Traffic light with no lights lit or covered with an out-of-order bag.

train and compare action classification algorithms.

View Based Approaches

Most algorithms that process video data do not look at the complete video at once, but rather process short *views* of the complete video scene. The term *view* refers to a temporal (and possibly spatial) crop of a long sequence of frames. Papers of note in this category include the Action Transformer proposed by Rohit et. al in [13]. The Action Transformer ingests short (3 second) views centered at keyframes in the AVA dataset, first passing them through a 3D convolution layer[1], and then a Region Proposal Network (RPN) [24] to extract initial candidate regions in the feature map space. The head of the network classifies and regresses each proposal using a transformer architecture, where the feature map around the proposal make up the memory (keys and values) of the attention mechanism, and the proposals themselves are used as the queries. The transformer is multi-headed and multi-

[1]Inflated 3D convolutions (I3D) are a common building block in video action recognition networks [7, 16, 26, 28], although excluded for brevity in this review.

layered (see [31]) with the motivation that self-attention will provide context from other actors and objects in the clip when the query is updated in each layer, helping to discern complex actions.

The issue with views is that some classifications are not possible given only a few seconds of frames to work with. The SlowFast design proposed by Feichtenhofer et. al [10] as well as Long-Term Feature Bank (LFB)s proposed by Wu et. al [33] provide two methods of at least partially removing this constraint. In SlowFast networks, two streams of temporal information are processed in parallel, one sampled at a high resolution (the "fast" stream), and another at low resolution (the "slow" stream). The fast stream captures detailed motion of actors in the scene, but uses a smaller CNN channel capacity and has a short view length, whereas the slow stream has a large CNN channel capacity and captures large temporal strides over the video. The two processing steams are connected via lateral connections from the fast pathway to the slow pathway, allowing fine motion details to augment the long temporal strides of the slow pathway.

LFBs [33] are similar to the slow stream of the SlowFast network, in that the bank accumulates information from a high temporal resolution stream that views only a few seconds from a 3D CNN. However, whereas SlowFast's slow steam processes its own low temporal resolution steam of information as it is fused with the high resolution stream, the LFB only accumulates information from the short term 3D CNN backbone.

With the efficacy of the SlowFast as a video feature extractors proven by its first place finish AVA ActivityNet Challenge 2019 [1], efforts in the past few years have focused on different network head architectures which compute actor-actor and actor-context relationships to separate harder to classify actions. A successful approach is Actor-Context-Actor Relation Network (ACAR-Net) [23], in which Region of Interest (RoI) (from an off-the-shelf detector e.g. R-CNN) aligned feature maps from SlowFast are first concatenated and passed through a 1x1 convolution to model actor-context relations as done in the Actor-Context-Relation Network [23]. However ACAR-Net goes further by introducing a High-order Relation Reasoning Operator (HR^2O) which computes relationships between pairs of the actor-context relations themselves. The author's argue that an operator such as HR^2O is needed to discern complex actions in the human domain where the action of one actor is dependent on another actor's context (e.g. riding in a car versus driving it). In Section 1.1.4 we examine the question of if such an operator is needed in the road scene domain.

Complete Video Approaches

Recently there have been efforts in natural language processing tasks to apply the transformer architecture to large corpuses, which the quadratic self-attention computation

complexity has previously prohibited. Works such as Longformer [4] and BigBird [34] propose methods to sparsify the self-attention mechanism, which they view as a graph formed by fully connected token nodes. This is achieved by introducing relational inductive biases (see [2]) to the graph structure including (1) limiting self-attention to a constant sized set of local neighboring tokens (Longformer and BigBird) possibly with dilation (Longformer), (2) attending to a constant sized set of random other tokens (BigBird), and (3) attending to a set of externally added global tokens (Longformer and BigBird).

In the video understanding domain, where frames can be viewed as tokens in the "video corpus", these sparsification techniques have also enabled self-attention to be applied over never before possible lengths of videos. The Video Transformer Network (VTN) proposed by Neimark et. al [22] is one such work, directly applying the Longformer sparsification technique to frame tokens produced by any 2D feature extractor (Neimark et. al found the Vision Transformer (ViT) [9] backbone most effective). VTN is able to attend to frames in the video that are most important for the classification task, effectively ignoring "noise" frames.

1.1.3 Methodology

Overall, our methodology behind RAD-NetV1 is to combine state-of-the-art agent detectors with strong classifiers for human actions, striving to improve on the ROAD tasks discussed in Section 1.1.2. In the remainder of the section we will discuss: (1) An overview of the two stages of RAD-NetV1; (2) our approach to object detection and classification using CenterNet; (3) our approach to action classification, comparing the SlowFast + Feed Forward Network (FFW) and SlowFast + Actor-Context-Actor Relation (ACAR) algorithms; (4) integration details of RAD-NetV1's two stages.

RAD-NetV1

Unlike the baseline 3D-RetinaNet, RAD-NetV1 uses a two-stage approach to solve the agent detection, action classification, and duplex detection tasks. This decision was made because agent detection and action classification are individually well-studied tasks, and the strengths of existing solutions can be combined in a two-stage approach.

The first stage of RAD-NetV1 is object detection, for which we use CenterNet [37], the backbone of one of the top performing algorithms (CenterTrack [36]) from the KITTI vision benchmark that only uses RGB images. The CenterNet algorithm produces a list of bounding boxes with agent classifications, effectively solving the agent detection task.

The second stage of RAD-NetV1 deals with action classification, a task that requires special attention and treatment of the temporal dimension to separate action classes. Therefore, a different sort of network architecture is necessary. Instead of the single temporal-resolution 3D-CNN that the baseline employed, we use the more recently developed SlowFast architecture [10] as our video feature extractor. SlowFast employs and fuses multi-resolution temporal streams of frames, combined with non-local [32] blocks (see Section 1.1.2). For the head of the network (which generates the action-class scores), we experimented with two options in order to discern the effect of ACAR-Net: (1) A FFW prediction head as done in the original SlowFast paper [10]; (2) further relational-context reasoning via the ACAR-Net head design [23].

In order for RAD-NetV1 to solve the ROAD tasks, information from both the first and second stage is necessary. Specifically, the stage-two action classifier needs bounding box priors from the stage-one detector. Thus, as shown in Fig. 1.2, when a new *keyframe* is processed it is first fed through the detector to produce bounding box priors. The box priors are then used as the input, along with contextual frames around the keyframe, to the action classifier which provides action-scores on a per-box basis.

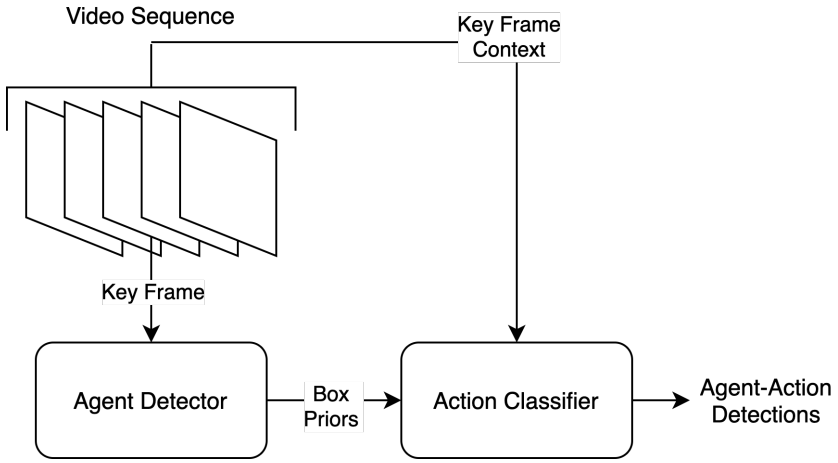


Figure 1.2: Schematic of the RAD-NetV1 agent-action detection pipeline as described in Section 1.1.3

Stage I: Object Detection

Acknowledgement: The object detection work described in this section was done by Chuan Tian (Ben) Zhang.

For the first stage of the RAD-NetV1 pipeline, we applied state-of-the-art object detection algorithms on the ROAD dataset. CenterNet is used and performed well on the MS-COCO 80-class object detection and classification dataset. We performed experiments using CenterNet under two settings: training on the ROAD dataset from scratch and training on the ROAD dataset with initial weights from a pretrained model used for the MS-COCO dataset. Because of the difference in the number of classes (11 in ROAD v.s. 80 in MS-COCO), some head weights are dropped in the transfer learning experiment.

To set up the experiments, we converted the ROAD dataset annotation format into the COCO annotation format, and adapted an existing implementation of the CenterNet algorithm to work with the transformed dataset. To evaluate the results, we used the COCO API, which implements the standard intersection-over-union (IoU) calculations and evaluates the agent labels with the equality condition. This is consistent with the definition of ROAD, which uses 0.5 as the IoU threshold to determine the correctness of bounding boxes.

The learning rate schedule used is identical to the one used for the "ctdet_coco_dla_2x" experiment in the original CenterNet implementation [35].

Stage II: Action Classification Algorithms

Starting out, we had two hypotheses regarding action classification. First, we hypothesized that algorithms which performed well in the human action domain are also well-suited for the road scene domain, even though there is a wider range of intelligent actors, i.e. cyclists, vehicles, in road scenes. Furthermore, we hypothesized that transfer learning can be applied to leverage the existing large scale human action datasets, such as AVA, and improve the action classifier's performance in the road scene domain.

To test these hypotheses we experimented with two action classification algorithms: SlowFast backbone + FFW head and SlowFast backbone + ACAR-Net head. Both of these algorithms performed well on the AVA human action dataset.

We also experimented with different weight initialization settings. Each algorithm was trained in a "scratch" setting where all weights were initialized randomly and in a "pretrained" setting where the Slowfast backbone weights were initialized using a model trained on the AVA dataset. Note that in the "pretrained" setting the head network weights were not transferred as the head network architecture changes based on which set of action classes are being predicted. Both settings were then trained (or fine tuned) on the "train-1" split of the ROAD dataset videos, using the annotated action classes found in Table 1.2, and evaluated on the corresponding "val-1" split. During training, the ground

truth bounding box priors were used as inputs along with 30 (SlowFast + FFW) or 45 (SlowFast + ACAR-Net) frames of view context. An overview of the training process is shown in Fig. 1.3.

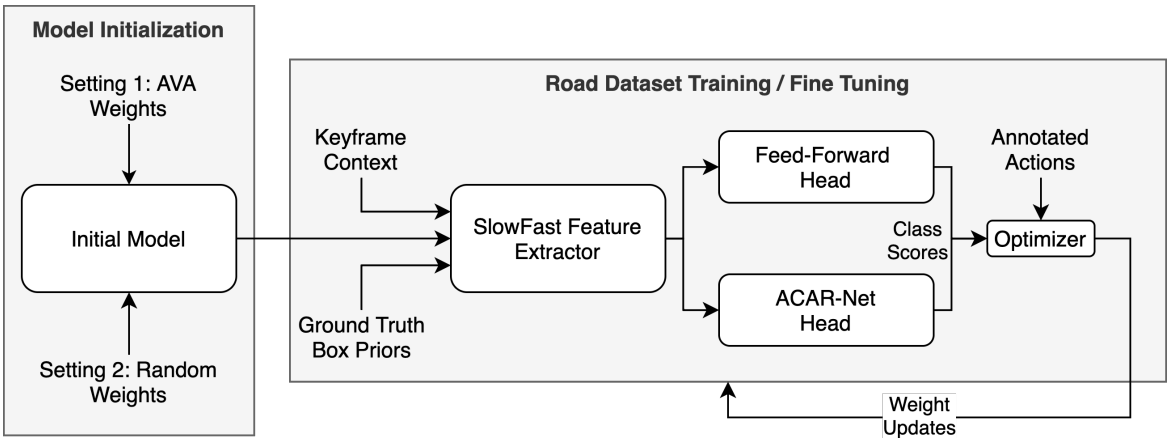


Figure 1.3: Schematic of the action classifier training pipeline as discussed in Section 1.1.3. Note that the two heads are trained separately, and are only shown together here for compactness.

The same optimizer, learning rate schedule, and data augmentation techniques were used as in the publicly available implementations of SlowFast [11] and ACAR-Net [8].

Integration Details

In order to calculate the mean Average Precision (mAP) metrics which are comparable to the 3D-RetinaNet baseline, the two stages discussed previously were integrated by providing the bounding boxes generated by the detector to the action classifier as priors. To this end, the detector first generates a JSON file containing detections on each frame, uniquely keyed by the video and frame IDs. Then, when the action classifier evaluates a keyframe, the unique key for that frame is generated again and box priors and retrieved from the JSON file. The keyframe context is retrieved from the dataset as usual, and passed through the action classifier along with the predicted bounding boxes in the keyframe. This integration process allowed us to evaluate RAD-NetV1 in the same way that the 3D-RetinaNet baseline was evaluated. These results are discussed in the next section.

1.1.4 Results and Discussion

Overall, the agent detector and action classification algorithms performed strongly in their respective stages (achieving 62 mAP and 34 mAP individually). When the two stages were integrated, RAD-NetV1 performance dropped to 25 mAP, one point below the baseline 3D-RetinaNet. These initial results are nevertheless promising, and we believe that RAD-NetV1 will outperform the baseline after further refinement (see Section 1.1.5). The remainder of this section discusses in detail results of the CenterNet, SlowFast, ACAR-Net algorithms.

CenterNet Results

The two CenterNet experiments (training from scratch and transfer learning) were trained for 11 epochs each and they appear to have drastically different performance characteristics. The experiment where the CenterNet model is trained from scratch showed promising decrease in loss initially. However, both the training and validation losses worsened shortly after. The transfer learning experiment, on the other hand, exhibited expected behaviour with the training loss decreasing in each epoch.

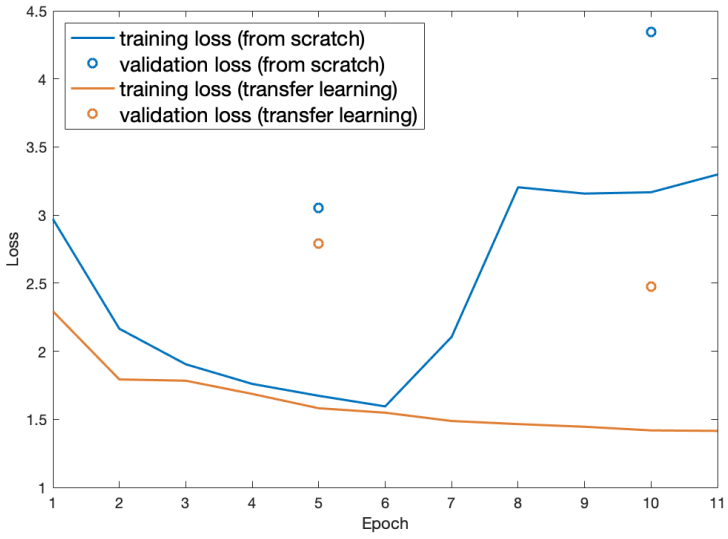


Figure 1.4: Training and validation loss progression the CenterNet experiments. The validation loss is evaluated every 5 epochs.

Fig. 1.4 shows the loss at each epoch of training. The experiment where training started from scratch reached its minimum training loss at epoch six. Then the training loss degraded rapidly. This may be due to the learning rate being set too high. The transfer learning experiment, on the other hand, shows a steadily decreasing loss. Due to time and resource constraints, the training for both experiments was stopped after completing epoch 11.

Type	IoU	TL	FS	Baseline
AP	0.50	0.620	0.549	0.445
AP	0.75	0.321	0.240	N/A
AR	0.50-0.95	0.487	0.430	N/A

Figure 1.5: Object detection and classification: best average precision (AP) and average recall (AR) values evaluated on the val-1 split of the ROAD dataset. Two experiments are shown: training CenterNet from scratch (FS), training CenterNet with transfer learning (TL). The 3D-RetinaNet baseline result as described in [25] is also shown for reference.

Fig. 1.5 shows the best (among all epochs) average precision and average recall values for various experiments. The transfer learning experiment clearly outperforms the experiment where the training is done from scratch. Both transfer learning and training from scratch surpassed the baseline results.

SlowFast and ACAR-Net Results and Comparison

Three different implementations are compared here: (1) SlowFast + FFW (Scratch) has the SlowFast feature extractor trained from scratch (random weight initialization, see Fig. 1.3) and a FFW head; (2) SlowFast + FFW (Pretrain) is the same as SlowFast + FFW (Scratch) but with the SlowFast feature extractor weights initialized from a model trained on the AVA Dataset; (3) SlowFast + ACAR (Pretrain) is the same as SlowFast + FFW (Pretrain) but with the head from ACAR-Net used instead of the FFW head.

Fig. 1.6 shows the mAP performance on the "val-1" split versus number of epochs trained on the ROAD dataset for each implementation. Note that these metrics were generated using the ground truth box priors from the annotation database, in order to evaluate the efficacy of the action classifier in isolation.

These mAP metrics are comparable to those reported in the AVA leaderboard [1]. This confirms our first hypothesis, that action classification algorithms developed for the human action domain also perform well in the road scene domain.

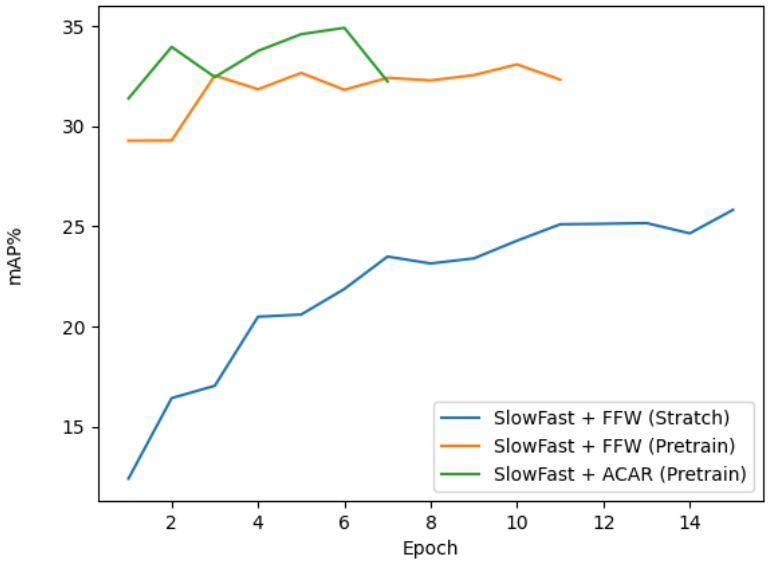


Figure 1.6: mAP versus number of epochs trained on the ROAD dataset for each implementation variant of the action classifier described in Section 1.1.4

Importantly, there is a profound increase in performance when using transfer learning from the AVA human action dataset (see Fig. 1.6 blue vs. orange plots). In fact, the performance is nearly saturated after a single epoch of fine tuning on the ROAD dataset (see Fig. 1.6 orange plot). These findings are evidence that RAD-NetV1 is able to learn shared representations of actions between different domains (human and road scene). This domain transfer phenomena is well-documented in the image classification task, but to our knowledge these findings are first documented evidence in the action classification task.

Lastly, we also note that the ACAR head was able to outperform the FFW head. However the gain was smaller than reported for the AVA Dataset (3 points versus 5 points), leaving effectiveness of ACAR's relational-context design in road scenes up for discussion.

To get a more-fine grained interpretation of these results, Fig. 1.7 shows the performance of each implementation variant split by each action class. Looking at the per-class differences, we see that pretraining the SlowFast feature extractor improved performance across the board. Comparing the FFW and ACAR heads, ACAR noticeably improves performance (by about 10 points) on the "crossing" (directional, waiting, etc...) related

actions. These actions are visually similar, and can only be separated by looking at the context of the scene and relation to other actors, areas that the ACAR-Net was designed for.

We also note that the more difficult (crossing, turning) and most difficult (overtaking) actions have a large temporal scope compared to the easier actions (being stopped, or moving away). Implementation of a LFB (see Section 1.1.2), or variants on the long-term attention mechanism like VTN (see Section 1.1.2) may improve performance in these difficult classes.

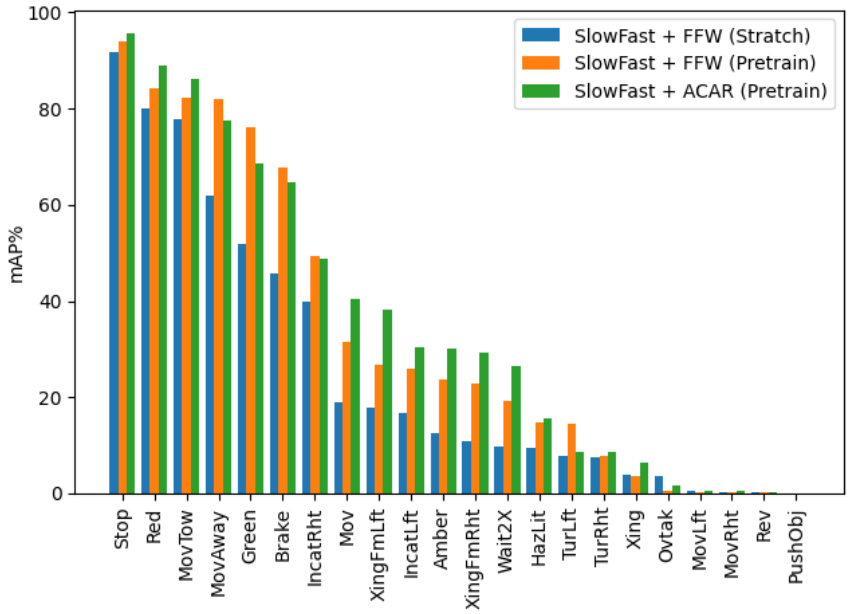


Figure 1.7: mAP versus action class for each action classifier variant discussed in Section 1.1.4

RAD-NetV1 Results and Discussion

Quantitative Results: To assess the quantitative performance of RAD-NetV1, the ground truth box priors used for the action classifier training were replaced with box priors from the object detector. Object detector priors with confidence scores less then 0.3 were filtered out. The introduction of imperfect box priors decreased the mAP of the action classifier by nine points, to 25 mAP for RAD-NetV1.

Qualitative Results: Qualitative results produced by RAD-NetV1 can be viewed here: https://www.youtube.com/watch?v=asylL5mFisw, using one of the videos from the val-1 split. Fig. 1.8 shows example keyframes taken from this video which emphasize why action detection work encouraged by the ROAD dataset is crucial for automated driving systems.

Although our initial implementation of RAD-NetV1 lags behind the baseline's 26 mAP performance on the action task, refinements (discussed below) are available that will close this gap and improve on the baseline.

1.1.5 Conclusions and Future Work

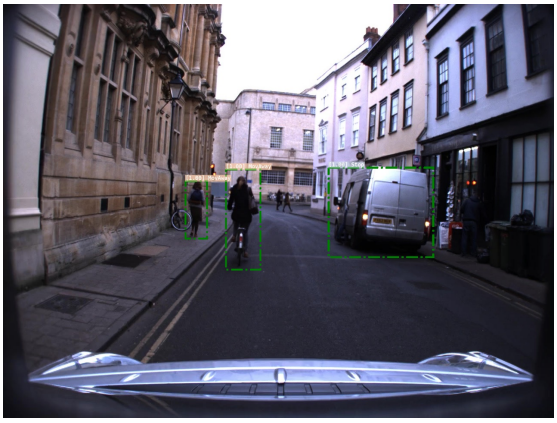
This section described our original two-stage approach, RAD-NetV1, to the action detection and classification tasks posed by the ROAD dataset. The proposed agent detector and action classifier stages both perform well individually. However, the performance degrades when the two algorithms are combined for RAD-NetV1 to solve the duplex task. Refinements are discussed in the next section to address this drop in performance.

Specifically, the concept of "active-agent" detection needs to be addressed. In the ROAD dataset, only active-agents, defined as agents that are actively performing an action, are labeled. However, the CenterNet detector (and other common detectors) are not designed for such a discriminative task, as inactive and active agents may appear identical in a single frame. Thus, the drop in performance when using the detector boxes as action-classification priors instead of the ground truth boxes can be partially explained by a large number of false positive active-agent detections being present in the detector boxes, but not the ground truth boxes. Improvements on active-agent detection is discussed in Section 1.2.1.

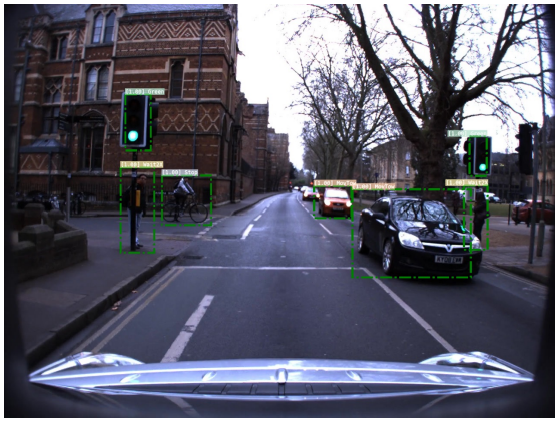
Furthermore, the current single keyframe box prior approach is ill-suited for the difficult actions with large temporal-scope, as the actor of interest may occupy different regions of the contextual frames, relative to the keyframe. Thus, adjustments the RoI Align algorithm from [15] (which both SlowFast and ACAR-Net use to splice the prior boxes out of the CNN feature maps) are necessary to support varying box priors across the contextual frames, as discussed in Section 1.2.2.

Lastly, the imbalance of the action classes in the ROAD dataset also needs to be addressed (see Fig. 1.9 for class instance counts). One simple solution is to over-sample the low-instance classes while training, which is usually done via a denser data augmentation regime for those classes. Another approach is to use a loss function which is less sensitive to classes that the network is already well-fit to. Focal loss [19], as used in the baseline

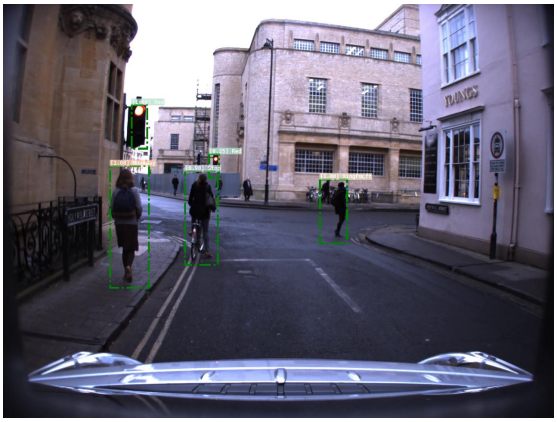
(a) In this frame the ego vehicle is beginning to make a left turn in the presence of a pedestrian. Deciding to yield or proceed is dependent on the actions of the pedestrian, which in this case RAD-NetV1 classified correctly as "Waiting to Cross".



(b) A second scene where the safety of the ego vehicle is dependent on whether the idling van is stopped, or indicating that it will be entering the ego's lane. In this case RAD-NetV1 correctly classifies the van as "Stopped", and the ego proceeds.



(c) This frame presents a more complicated scene where different types of intelligent actors (pedestrians and cyclists) as well as occlusions (of the cross walk for the pedestrian on the right) are present. Deciding to proceed safely through the intersection requires the actions of the two pedestrians, and the bike to be classified, which in this RAD-NetV1 does correctly.



(d) Another challenging scene for deciding how close to the stop line the ego vehicle should be as it approaches the correctly classified red light. The information required here is that the pedestrian on the left is moving away, instead of crossing in front of the ego vehicle. Also, the cyclist is stationary on top of the stop line, forcing the ego vehicle to stop well before the line.

Figure 1.8: Example outputs of RAD-NetV1 on keyframes in a "val-1" split video from the ROAD dataset. These output frames are chosen to emphasize the importance of action classification in road scenes. Best viewed using software with zoom ability to see labels.

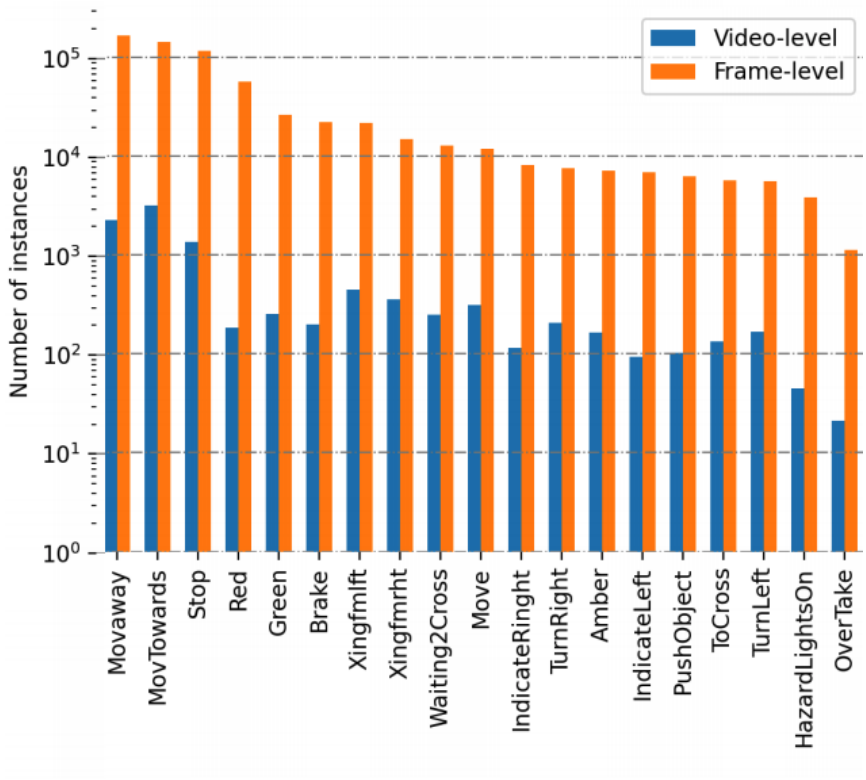


Figure 1.9: Instance counts of various actions as specified in [25]. Shows an 100x class imbalance for some action classes (Move Away vs. Overtake). These imbalances correlate with performance drops as seen in Fig. 1.7.

3D-RetinaNet, is a good candidate in this regard and its implementation is discussed in Section 1.2.3.

Although not discussed in the next section, future work could also include experimenting with convolutional-free video architectures, such as TimeSformer [5], which are built on purely self-attention computations over the space and time dimensions. Such designs, although radically different than the 3D-CNNs discussed in this report, have shown state of the art performance on the Kinetics Dataset, and can be applied to longer views.

1.2 Road Action Detection Network V2 (RAD-NetV2)

In light of the shortcomings of RAD-NetV1, various improvements were investigated, as identified in Sections 1.1.5.

1.2.1 Active Agent Detection

Acknowledgement: The active agent detection work described in this section was worked on collaboratively with Quanquan Li.

A major drawback of RAD-NetV1 was that the first stage detection algorithm had a large number of false positives of active agents, because an inactive vehicle can appear practically identical to an active one, see Fig. 1.10 for an example.

Figure 1.10: Here we see a datapoint from the ROAD dataset which contains both active (which have labels) and inactive (no labels) vehicles. Note how both kinds of vehicles (active and inactive) have a similar appearance, and thus are difficult for an appearance based detector to discriminate between.

Creating an Active-Inactive Agent Dataset

In order to discriminate between active and inactive agents, at the minimum a dataset with labels of inactive agents is required. To generate such a dataset, a common object detector trained on the COCO dataset is used in conjunction with the ROAD dataset. Then, for each frame in the ROAD dataset, let R_i be the set of labeled (active) agents in frame i of the ROAD dataset, and let O_i be the set of all agents detected by the object detector. Then, the set of inactive agents is O_i/R_i, where R_i set membership is established by an IoU of greater than 0.2. The resulting "psuedo-labels" of inactive agents are shown in Fig. 1.11.

Figure 1.11: Example frame from the created active-inactive agent dataset. Agents in green are any active class (as in the original ROAD dataset), whereas agents in red are the newly created pseudo-labels of inactive agents.

Average Precision (AP) results from training on the active-inactive dataset versus training on the original ROAD dataset are presented in Table 1.3

As shown in Table 1.3, using the created inactive agent dataset for training helps the object detector differentiate bewteen active and inactive agents, yielding an improvement of 7.1 points on AP@0.5-0.95 and 6.6 points on AP@0.5, using the original ROAD validation set.

Model	AP@0.5-0.95	AP@0.5
Original ROAD Dataset	30.9	61.3
Inactive Agent Dataset	38.0	67.9

Table 1.3: Comparison of validation accuracy for training on the ROAD dataset versus the created active-inactive dataset.

Super-category Classes

As shown in Fig. 1.1, there is an issue with agent class imbalance in the ROAD dataset, biasing the training of the detector towards the common classes and harming mAP. However, note that specific class identities are not used for the final task of action classification, only the bounding boxes are. Thus, super-classes can be created for classes that are visually similar. For example, merging MedVeh, LarBeh, Bus, and EmVeh into a new class "Vehicle" alleviates the imbalance of EmVeh shown in Fig. 1.1. In total we are left with 6 new super-classes: Ped, Vehicle, Cyc, Mobike, TL/OthTL, Inactive. The new class distribution is well balanced and improves detection mAP substantially as shown in Table 1.4. However, this is not a fair comparison since the task has fundamentally changed. To get a fairer comparison the final task of end-to-end action classification can be assessed using the bounding boxes from the 10-class dataset vs. the new 6-super-class dataset. Results for that are also presented in Table 1.4.

Super-category Class Balance	Detection mAP	Action mAP
	67.9	23.8
✓	80.4	24.9

Table 1.4: Comparison of validation accuracy for the detection model with and without super-category class balance.

As seen in Table 1.4, super-class based detection does in fact improve the end-to-end action mAP by 1.1 points.

(a) The first frame from which optical flow is derived.

(b) The second frame from which optical flow is derived.

(c) The optical flow obtained from the two images.

Figure 1.12: A visualization of the inputs and outputs of the RAFT network. In the output (c) the active agent can be clearly identified, unlike in the inputs (a) and (b).

Optical Flow Signals

As seen in Figs. 1.10 and 1.11, the active and inactive classes generated in Section 1.2.1 are visually similar, and thus do not provide a strong visual discrimination signal for the detector to work with. A better signal of agent activeness is whether or not the agent is moving across time in the scene. Optical flow is a method for computing such a signal for each pixel in sequence of two images. To compute an optical flow image for each frame in the ROAD dataset, we used RAFT: Recurrent All Pairs Field Transforms for Optical Flow [27]. See Fig. 1.12 for an example of the optical flow images generated. In order to apply pretrained convolutional filters to the flow maps, a color wheel representation is used where the direction of the flow vector determines the color and the magnitude of the flow vector determines the intensity.

In order to fuse the appearance information from the RGB images, as well as the motion information from the optical flow image, either a shallow fusion or a deep fusion architecture can be used (see Fig. 1.13). In shallow fusion, the flow map is first concatenated to the RGB image before being passed through the ResNet backbone. Although simple, the main issue with the shallow fusion architecture is that ResNet filters pretrained on ImageNet and COCO cannot be used because the number of channels in the input tensor has changed. Alternatively, a deep fusion architecture can be employed where concatenation is delayed until after the individual application of the ResNet and Feature Pyramid Network (FPN) layers. The benefit of the deep fusion design is that the pretrained ResNet filters can be used individually on each of the 3-channel RGB and colorized flow maps, which increases accuracy as shown in Table 1.5.

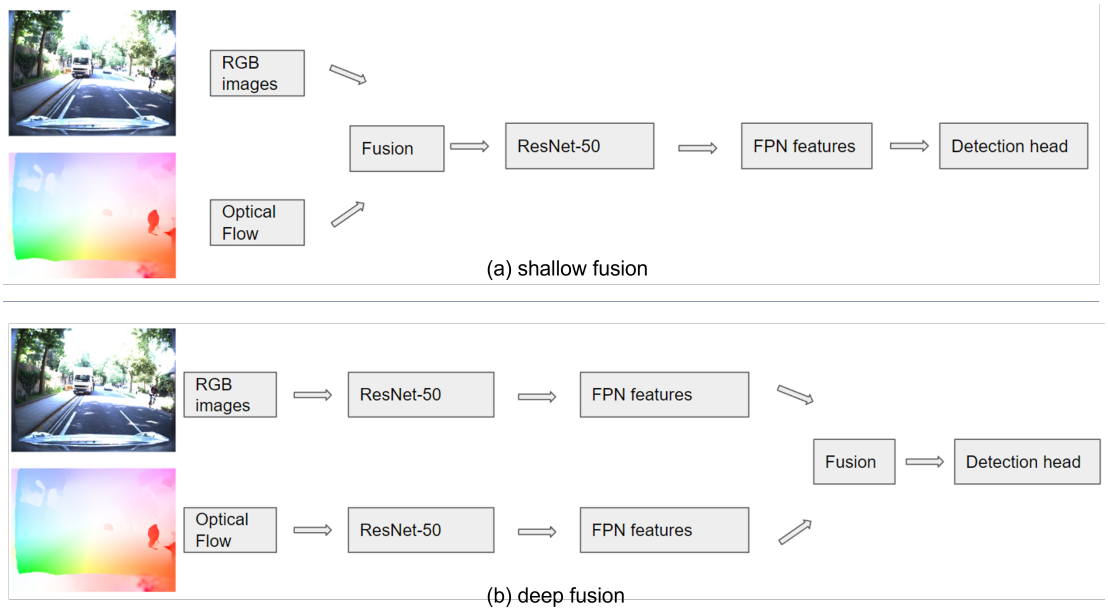


Figure 1.13: Comparing the shallow fusion architecture, which does not allow for usage of pretrained ImageNet and COCO filters, to the deep fusion architecture, which does allow for pretrained filters.

Model	AP@0.5-0.95
Baseline (RGB only)	56.2
Shallow fusion	52.2
Deep fusion	59.9

Table 1.5: Comparison of validation accuracy under different optical flow fusion strategies.

As seen in Table 1.5, introducing the flow map signal does in fact boost performance by 3.7 points when using the deep fusion strategy. However, the shallow fusion actually decreases performance by 4 points. This comparison shows the importance of using the pretrained ImageNet and COCO filters, afforded by only the deep fusion strategy.

Furthermore, as seen in Table 1.6, the deep fusion strategy continues to improve performance by 0.9 points when applied on-top of the inactive agent dataset and super-class improvements discussed above.

Model	AP@0.5-0.95
Baseline (RGB only)	80.4
Deep fusion	81.3

Table 1.6: Comparison of validation accuracy for the best performing optical flow model. The baseline includes using the inactive agent dataset and the super-class balancing strategy discussed above.

1.2.2 Dealing with High Motion Scenes

RAD-NetV1 inherited its keyframe based design from ACAR-Net, which was targeted at low-motion scenes found in the AVA Dataset. In low-motion scenes, spatial information from a single frame suffices to describe the locality of the action across the entire temporal view. However, given the high ego and non-ego motion of agents in the ROAD dataset, spatial information in the keyframe does not accurately localize the action in frames far from the keyframe (see Fig. 1.14 for an example).

Dynamic ROI (Tube) Based ACAR

Acknowledgement: The tube-based RoI-Alignment implementation described in this section was worked on collaboratively with Eddy Zhou.

To mitigate this spatial mismatch across the view, we changed the input definition of the action classification model to take a sequence of detections across the view (i.e. a track, or tube) for each agent in the keyframe. In order to implement this change we needed to introduce a tracker on-top of the detection step, and change the ACAR-Net Head architecture to support multiple RoIs for each agent we are classifying the action of.

For the tracker we chose OC-Sort [6] for its state of the art performance. OC-Sort takes the detections as input, and associates a *tube_uid* with each detection that serves to associate detections over time. We note that OC-SORT is especially capable of handling occluded observations, which happen often in road scenes. In order to eliminate the possibility of different object classes in the same track, we separately track agents from different agent classes.

An illustration of the architecture change to the ACAR-Net Head can be found in Fig. 1.15. As seen in 1.15(a), ACAR-Net originally performed a single RoI-Align operation using the keyframe RoI, after the slow and fast feature tensors are already temporally

(a) First frame in frame-based ACAR-Net input

(b) Middle (keyframe) frame in frame-based ACAR-Net input

(c) Last frame in frame-based ACAR-Net input

(d) First frame in tube-based ACAR-Net input

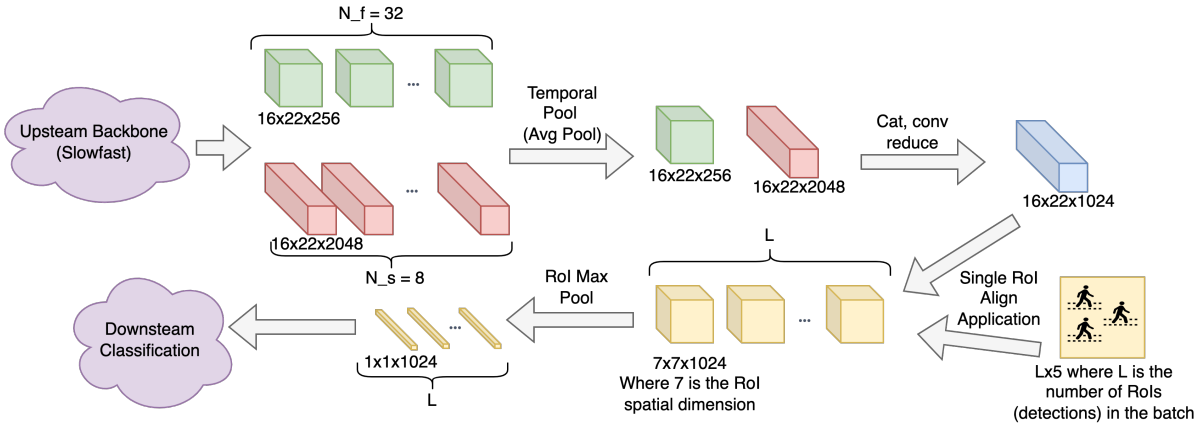
(e) Middle (keyframe) frame in tube-based ACAR-Net input

(f) Last frame in tube-based ACAR-Net input

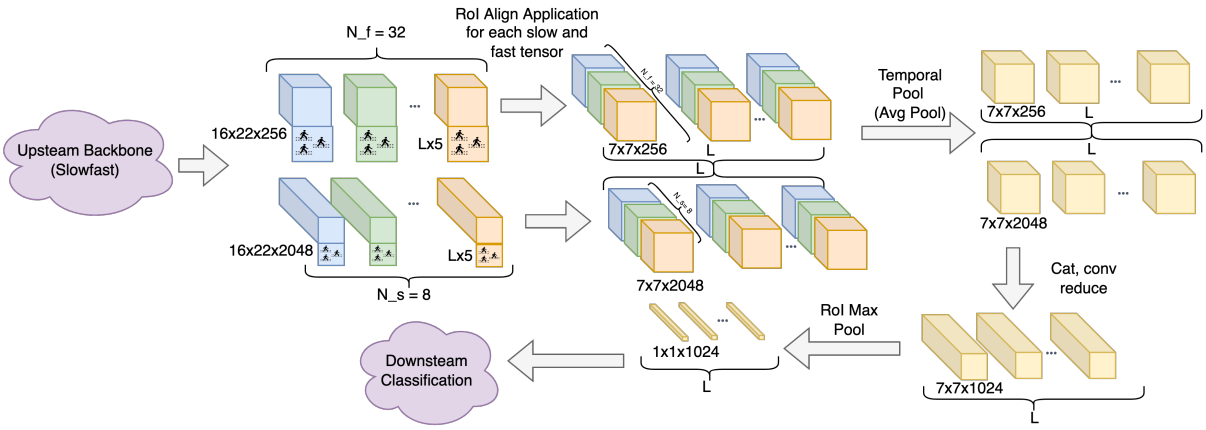
Figure 1.14: Difference between inputs to frame-based vs. tube-based ACAR-Net. The top row (a-c) is the frame-based input, obtained from just the detector. The keyframe detection (b) does not provide good spatial localization of the agent across the entire temporal view (a) and (b). In contrast, the bottom row (d-f) is the tube-based input based on the detector and the OC-Sort tracking algorithm. In combination with the modifications made to to the ACAR-Net model discussed in Section 1.2.2, the tube does provide good spatial tracking across the view, improving action classification accuracy as presented in Table 1.7.

pooled. As motivated by Fig. 1.14, we replaced this keyframe based RoI design with the design found in 1.15(b). In our new design a RoI-Align operation is performed for each fast and slow frame-tensor, using the RoIs from the tube of detections that correspond to each frame-tensor, before temporal pooling is applied. This way, the correct spatial features are extracted from each frame-tensor. This modified RoI-Align approach is outlined in Alg. 1.

The results of the tube based ACAR-Net design are shown in Table 1.7. As seen in the table, the tube-based ACAR-Net design increases performance by 2.5 points on top of the active agent improvements described in Section 1.2.1.



(a) The (original) keyframe RoI based ACAR-Net architecture.



(b) The redesigned tube RoI based ACAR-Net architecture.

Figure 1.15: Comparing keyframe based ACAR-Net with tube based. In (a), only a single RoI-Alignment operation is performed, after temporal pooling. This design works well for low motion datasets such as AVA, but may introduce spatial noise in high motion datasets such as ROAD. In (b) we redesigned the ACAR-Net Head to apply RoI-Alignment for each frame-tensor before temporal pooling. This method provides better spatial alignment of the agent's features across the entire temporal view, but requires a tube of detections as input.

Algorithm 1 Tube RoI-Alignment Adapted to SlowFast

Input: *slowFeats*, *fastFeats*, *bb* {list of bounding boxes per frame}
Output: Feature Tensor with Encoded Tracks through Tube ROI-Alignment

$roi\_fast\_feats = []$
$roi\_slow\_feats = []$
$alpha = \frac{temporal\_len(fastFeats)}{temporal\_len(slowFeats)}$
for *idx, fastFeat* $\in$ *temporal_enumerate(fastFeats)* **do**
  if $(idx + 1)/alpha = 0$ **then**
    $rois = ROIAlign(slowFeats[idx], bb[idx])$
    $roi\_slow\_feats.append(rois)$
  end if
  $rois = ROIAlign(fastFeat, bb[idx])$
  $roi\_fast\_feats.append(rois)$
end for
$fast\_feats = TemporalAvgPool(roi\_fast\_feats)$
$slow\_feats = TemporalAvgPool(roi\_slow\_feats)$
$feats = Concatenate(fast\_feats, slow\_feats)$
return *feats*

Network	Val 1
Original ACAR-Net + Optical Flow Detector	27.8
Tube ACAR-Net + Optical Flow Detector	30.3

Table 1.7: Improvements from tube ACAR-Net design, we see a 2.5 point improvement when using the tube based design discussed in Section 1.2.2. All ablations are reported on validation split 1 since it has the most even class distribution.

1.2.3 Dealing with Class Imbalance

Acknowledgement: The focal loss experiments described in this section were worked on collaboratively with Alexander Zhuang.

As examined in Section 1.1.5, there are magnitudes of difference between the number of action class instances. The baseline 3D-RetinaNet used focal loss to deal with the imbalance, however RAD-NetV1 did not, using vanilla Binary Cross Entropy loss instead. In RAD-NetV2 we adopted the focal loss implementation from 3D-RetinaNet [25] (originally

Algorithm 2 Sigmoid Focal Loss

Input: sigmoid activated *preds*, one hot encoded *labels*, number of positive samples *npos*, focusing parameter γ, weighting factor α

Output: Computed *loss*

$l1 = binary\_cross\_entropy(preds, labels)$
$\alpha\_weight = \alpha * labels + (1 - \alpha) * (1 - labels)$
$pt = preds * labels + (1 - preds) * (1 - labels)$
$\gamma\_focus = \alpha\_weight * ((1 - pt)^{\gamma})$
$loss = (l1 * \gamma\_focus).sum()/num\_pos$
return *loss*

from RetinaNet [19]). The loss implemented is presented in Alg. 2.

The results of switching to focal loss (with $\gamma = 2.0$ and α as the inverse frequency of each class) are presented in Table 1.8. As shown in the figure, focal loss effectively deals with the action class imbalance shown in Fig. 1.9, which is consistent with the results presented in [25], overall boosting performance by 3.0 points. This experiment was run only on the action classification stage, using the ground truth box priors from the ROAD dataset.

Loss Function	Val 1
Sigmoid Loss	34.7
Sigmoid + Focal Loss	37.7

Table 1.8: Improvements from focal loss design, we see a 3.0 point improvement when using the focal loss design discussed in Section 1.2.3. All ablations are reported on "val-1" split since it has the most even class distribution.

1.2.4 Final RAD-NetV2 Results

The final design of RAD-NetV2 with all the improvements from Section 1.2 is shown in Fig. 1.16.

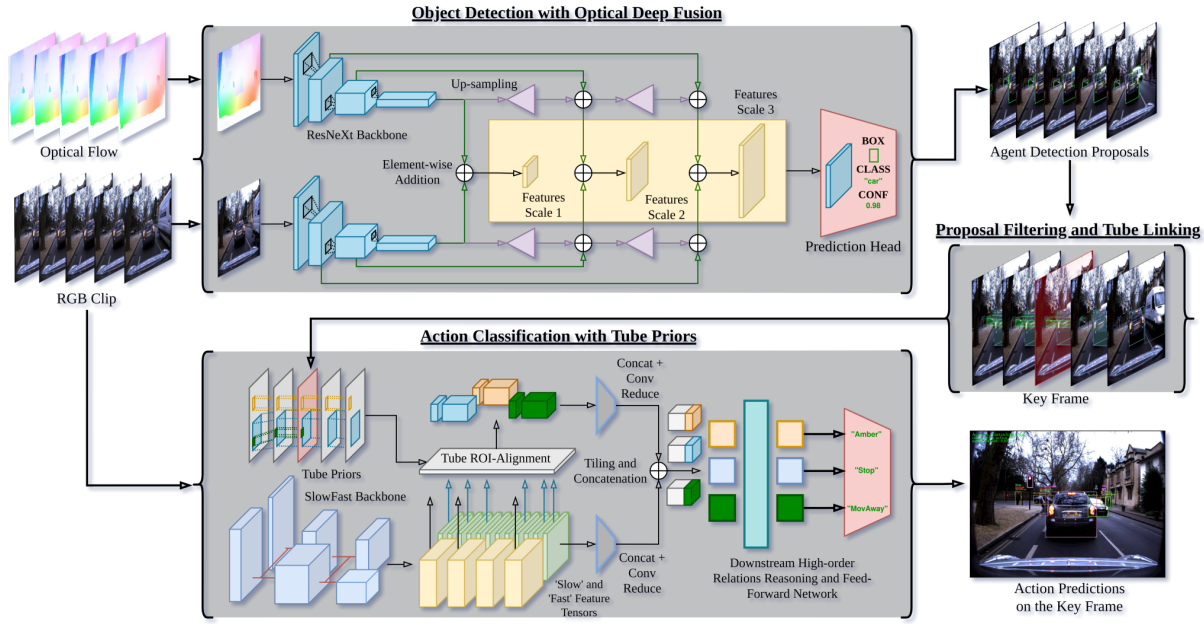


Figure 1.16: RAD-NetV2 system architecture. Given a set of frames in a clip, their optical flow is estimated using RAFT [27]. Both clips (RGB and flow) are sent frame-by-frame into the object detector where each RGB frame and its corresponding optical flow frame are encoded and summed up at multiple feature scales. We found that utilizing a pretrained RGB backbone on 3-channel optical flow improved model predictions and accuracy. Following detection, an online object tracker is used to link detections into tubes. Tubes present in the key frame of the clip are then fed into our action classifier, which encodes agent tubes through a novel RoI-Alignment procedure that takes advantage of the inherent structure of feature encodings outputted by SlowFast. Following encoding, we adopt [23]'s higher-order relations reasoning to compute the attention between each encoded agent tube and the other agents present in the clip. The result is the action predictions of agents present in the key frame.

We focus our performance evaluations on the International Conference of Computer Vision (ICCV) 2021 ROAD dataset. Our results show an improvement over the baseline on average, with a significantly improved frame and video-mAP on the "val-1" split. Full

results are shown in Table 1.9. Qualitative results of the entire RAD-NetV2 pipeline (active agent detection, OC-Sort based tracking, and modified ACAR-Net Head classification) can be found at
https://youtu.be/QQF8zlfEtlI.

Model	Val 1	Val 2	Val 3	Avg
3D-RetinaNet (frame-level)	26.2	**11.7**	21.2	19.7
Ours (frame-level)	**30.3**	10.0	**22.9**	**21.1**
3D-RetinaNet (Online)	17.0	11.4	**14.6**	14.3
Ours (Online)	21.1	11.4	13.4	15.3
Ours (Offline)	**24.0**	**11.73**	13.87	**16.52**

Table 1.9: Frame and video-mAP scores on the ROAD dataset. We compare frame-mAP@0.5IOU and video-level mAP@0.2IOU following the 3D-RetinaNet baseline. Experimentally, it is observed that these results may fluctuate by ∼1%. For our online results, Tube-ACAR considers a temporal extent of 32 frames, so we are able to perform box interpolation within gaps of 32 frames or less.

1.3 Deployment of RAD-NetV2 to Bolty

To enhance the scene understanding capabilities of Bolty, and drive in a less conservative manner, RAD-NetV2 was deployed to the perception stack, and integrated with the environment model.

1.3.1 Perception Integration

In order to deploy the RAD-NetV2 to the Automated Driving Stack (ADS), the network's inputs need to be generated in a real-time manner. Refer to Section **??** for a refresher on the data pipeline design, the relevant pieces are copied to Fig. 1.17 in more detail.

In brief, the history of 2D observations is added to each track. The observations are then time synchronized (see Alg. 3) with a buffered temporal view from the camera stream, and both the temporal view and the tube of observations are run through the action classification model. The result is a binary classification for each class, and the class with

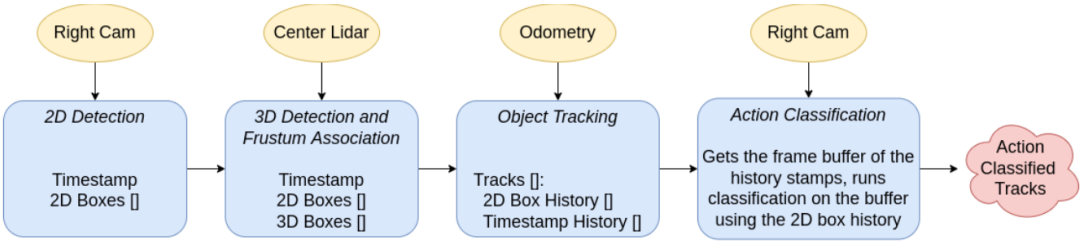


Figure 1.17: Data pipeline used to deploy RAD-NetV2 to Bolty. The resulting action classified tracks are sent to downstream modules (environment modeling), where they are used as input to behavioral planning decisions.

Algorithm 3 Detection - Frame Buffer Time Sync

Input: Frame *timestamps*, observed *bboxes* (both sorted from most recent to oldest)
Output: For each frame *timestamp*, finds the temporally closest *bbox* and *time_delta*

$match\_i = 0$
$matched\_bboxes = []$
$time\_deltas = []$
for $stamp \in timestamps$ **do**
  while TRUE **do**
    $match\_delta = abs(stamp - bboxes[match\_i].stamp)$
    $next\_match\_delta = abs(stamp - bboxes[match\_i + 1].stamp)$
    if $match\_delta < next\_match\_delta$ OR $match\_i + 1 >= len(bboxes)$ **then**
      BREAK
    end if
  end while
  $matched\_bboxes.append(bboxes[match\_i])$
  $time\_deltas.append(abs(stamp - bboxes[match\_i].stamp))$
end for
return *matched_bboxes*, *time_deltas*

the highest score (if it is over 0.5) is added to the action classification history for that track.

Integration Results

Real world testing was carried out at the Waterloo Region Emergency Services Training and Research Centre (WRESTRC). The entire test scenario can be viewed at `https:`

(a) First frame in the temporal view given to ACAR-Net

(b) Middle (keyframe) in the temporal view given to ACAR-Net

(c) Last frame in the temporal view given to ACAR-Net

Figure 1.18: Figure showing the visual and RoI inputs to the ACAR-Net model, as well as the top classification score across the entire tube, and the temporal misalignment for the frame.

//youtu.be/KdrLhPqi4Rk, which shows a pedestrian stopped on the side of the road. Fig. 1.18 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, as discussed in Section 1.2.2.

Table 1.10 presents the quantitative results of the deployment, comparing the keyframe based model against the tube based model. As seen in the table, the Tube ACAR-Net model produces the correct classification *stopped* the majority of the time, whereas the Keyframe ACAR-Net model misclassifies the majority of the time. This can be attributed to the poor region of interest input to the Keyframe ACAR-Net model, as analyzed before in Section 1.2.2.

Model	Stop	MovTow	MovAway
Keyframe	0.15	0.00	0.85
Tube	0.54	0.26	0.20

Table 1.10: Ratio of classifications over the test scenario.

1.3.2 Integration with Environment Model

The only change to the environment model is that incoming tracks of agents now have a history of action classifications attached to them. The environment model uses this additional information to better discern whether or not to create relations between the tracks and other entities in the Dynamic Relation Graph (DRG).

For example, refer to the original modeling of pedestrian tracks and their potential conflict relations with lanelets as laid out in Alg. **??**. In that original implementation of the algorithm, the decision of whether or not a pedestrian conflicts with lanelet was based on (1) whether the linear trajectory prediction of the pedestrian intersects with the lanelet and (2) if the pedestrian is stationary, whether it is within some radius of the lanelet. It is clear that these geometric heuristics will lead to a large number of false positive conflict relationships being instantiated. Not all pedestrians whose predicted linear trajectory enters a lanelet will in-fact enter that lanelet - perhaps there is simply a t-intersection in the sidewalk - and certainly not all pedestrians standing close to the road will jump out into traffic (see Fig. 1.19(a) for an illustration of this failure case).

The action classification history helps to limit these false positive cases by classifying the intentions of pedestrians based on their appearance descriptions, rather than geometrical descriptions. Alg. 4 implements a less conservative version of Alg. **??** by taking advantage of these appearance based intention classifications. Fig. 1.19 compares the resulting DRG state under the conservative Alg. **??** to the resulting DRG state under the improved Alg. 4 (using $safe\_set = \{Stop, Wait2X\}$). In the figure we see that the new implementation does improve decision making, allowing the Autonomous Vehicle (AV) to proceed past the stopped pedestrian.

(a) Shows the overly conservative behavioral decision for the ego vehicle to stop when there is no visual indication that the pedestrian will interfere with the ego vehicle. This flawed behavior can occur when only using the geometric heuristics specified in Alg. **??**, or when using an inaccurate action classifier as discussed in Section 1.3.1.

(b) Shows the reasonable ego behavior of driving past a stopped pedestrian, enabled by accurate action classification as specified in Alg. 4.

Figure 1.19: Comparing different ego behavior in the stopped pedestrian scenario under different DRG implementations.

Algorithm 4 Handle Pedestrian With Action

Input: Tracked *ped* with history of action classifications, history time t to consider, set of safe actions $safe\_set$.

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

$n\_interfere = 0$
$n\_total = 0$
for $action \in ped.action\_hist$ **do**
  if $time :: now() - action.stamp > t$ **then**
    BREAK
  end if
  $n\_total += 1$
  if $action.label \notin safe\_set$ **then**
    $n\_interfere += 1$
  end if
end for
return $\frac{n\_interfere}{n\_total} > 0.5$

References

[1] Google AI. Ava challenge. Available at http://research.google.com/ava/challenge.html.

[2] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[3] Harkirat Singh Behl, Michael Sapienza, Gurkirt Singh, Suman Saha, Fabio Cuzzolin, and Philip HS Torr. Incremental tube construction for human action detection. *arXiv preprint arXiv:1704.01358*, 2017.

[4] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[5] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021.

[6] Jinkun Cao, Xinshuo Weng, Rawal Khirodkar, Jiangmiao Pang, and Kris Kitani. Observation-centric sort: Rethinking sort for robust multi-object tracking. *arXiv preprint arXiv:2203.14360*, 2022.

[7] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[8] Siyu Chen. Acar github. Available at https://github.com/Siyu-C/ACAR-Net.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold,

Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[10] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019.

[11] Haoqi Fan Yanghao Li Bo Xiong Wan-Yen Lo Christoph Feichtenhofer. Slowfast github. Available at https://github.com/facebookresearch/SlowFast.

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[13] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2019.

[14] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6047–6056, 2018.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[16] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[17] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[18] Xinran Li, Kuo-Yi Lin, Min Meng, Xiuxian Li, Li Li, and Yiguang Hong. Composition and application of current advanced driving assistance system: A review. *arXiv preprint arXiv:2105.12348*, 2021.

[19] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[20] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.

[21] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Tom Yan, Lisa Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(2):502–508, 2019.

[22] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. *arXiv preprint arXiv:2102.00719*, 2021.

[23] Junting Pan, Siyu Chen, Mike Zheng Shou, Yu Liu, Jing Shao, and Hongsheng Li. Actor-context-actor relation network for spatio-temporal action localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 464–474, 2021.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[25] Gurkirt Singh, Stephen Akrigg, Manuele Di Maio, Valentina Fontana, Reza Javanmard Alitappeh, Suman Saha, Kossar Jeddisaravi, Farzad Yousefi, Jacob Culley, Tom Nicholson, et al. Road: The road event awareness dataset for autonomous driving. *arXiv preprint arXiv:2102.11585*, 2021.

[26] Graham W Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *European conference on computer vision*, pages 140–153. Springer, 2010.

[27] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[28] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[29] A. Geiger P. Lenz C. Stiller R. Urtasun. Object tracking evaluation (2d bounding-boxes). Available at http://www.cvlibs.net/datasets/kitti/eval_tracking.php.

[30] Jessica Van Brummelen, Marie O'Brien, Dominique Gruyer, and Homayoun Najjaran. Autonomous vehicle perception: The technology of today and tomorrow. *Transportation research part C: emerging technologies*, 89:384–406, 2018.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[32] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[33] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 284–293, 2019.

[34] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*, 2020.

[35] Xingyi Zhou. Centernet github. Available at https://github.com/xingyizhou/CenterNet.

[36] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020.

[37] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019.

[38] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.