Master Thesis

Assessing aggressive driving behavior using attention based models

Jonathan Aechtner

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science of Data Science for Decision Making
at the Department of Advanced Computing Sciences
of the Maastricht University

Thesis Committee:

Mirela Popa
Anna Wilbik

Maastricht University
Faculty of Science and Engineering
Department of Advanced Computing Sciences

July 6, 2023

Abstract

Aggressive driving behavior poses significant risks to road safety. This thesis aims to enhance the understanding of aggressive driving behavior through the use of attention-based models, feature extraction techniques, and the METEOR driving dataset. The dataset, known for its extensive annotations, was refined to tailor to the specific needs of this research. Feature extraction was a cornerstone of this study, and the research employed pre-trained backbones to extract frame-level and clip-level features. The experiments conducted revealed the relationship between the dimensionality of feature representation and model performance, and the importance of frame rates in capturing subtle details. Additionally, the models OadTR and Colar, both attention based were implemented, combined and evaluated, indicating unique strengths and weaknesses for each. Across the board there exists a clear correlation between the frequency of an action in the training's data, and each models ability to classify it. The thesis introduced two main contributions. First, the integration of the OadTR model with the Colar model, creating a novel hybrid that can exploit historical exemplars while predicting future frames. Second, the generation of salient cues for model interpretability using the agent paths through spatial and temporal dimensions. The findings of this study are particularly relevant for applications in autonomous vehicles, where real-time interpretability and minimal computational resources are crucial.

Contents

Chapter 1

Introduction

1.1 Problem Statement

Aggressive driving behavior (ADB) poses a significant threat to road safety. Understanding the psychological indicators of ADB is crucial for assessing the likelihood of a driver engaging in aggressive actions [1]. However, with the advent of autonomous vehicles, there is a pressing need to address the challenges associated with predicting and mitigating aggressive driving behavior in the context of human-autonomous vehicle interactions. Autonomous vehicles must be capable of accurately predicting the movements of human drivers, especially those exhibiting aggressive driving behavior, to ensure safe navigation and reduce the risk of accidents. This research aims to enhance the understanding of ADB and develop attention-based models that can effectively identify and predict aggressive driving behavior, particularly in the context of autonomous vehicles.

1.2 Background

Aggressive driving has been a longstanding issue, with roots tracing back to the early days of modern road infrastructure. J.J. Leeming highlighted the competitive and aggressive attitudes of drivers as early as 1969 [2]. Since then, research in this area has evolved, leading to a distinction between road rage and ADB. While road rage involves the intent to harm, ADB is characterized by a disregard for the safety and well-being of other road users, often motivated by impatience, annoyance, or time-saving [3].

With the emergence of autonomous vehicles, understanding and predicting ADB becomes even more critical. Autonomous vehicles interact with human drivers, who may exhibit unpredictable and aggressive behavior. The ability of autonomous vehicles to recognize and respond to such behavior is essential for ensuring road safety.

1.3 Significance

This research holds significance in several aspects. Firstly, it contributes to the enhancement of road safety by developing models capable of identifying and predicting aggressive driving behavior. This is particularly important in the context of autonomous vehicles, which need to navigate safely among human drivers.

Secondly, by focusing on attention-based models, this research brings in cutting-edge technology to address a real-world problem. The application of attention-based models can lead to more accurate and efficient predictions compared to traditional methods.

Thirdly, the research has implications for policy-making and regulation. The insights gained can be used to inform policies and regulations regarding autonomous vehicles and road safety. This is particularly relevant as autonomous vehicles become more prevalent on the roads.

Lastly, the research emphasizes the importance of high-quality data and representation in developing effective models. It highlights the need for datasets that are representative of real-world driving behaviors, which can be a valuable resource for future research and development in this area.

1.4 Research Questions

This thesis aims to answer the following research questions:

1. Which sampling approach of video data yields the best results in the context of transformer models?
2. Which attention-based model architecture is best suited for the task?
3. How can the model's explainability be improved by identifying salient cues for driving behavior assessment?

Chapter 2

Related Work

The task of detecting aggressive driving behavior in video data intersects several distinct research areas. To facilitate clarity and comprehension, this section is structured according to the architecture of a typical deep learning-based model for online action detection. Consequently, the discussion begins with an examination of feature extraction methods, followed by a comprehensive overview of various model designs and approaches. Additionally, the psychological and behavioral aspects of aggressive driving behavior are explored, providing a broader context for understanding the problem at hand. A subsection dedicated to explainable artificial intelligence (XAI) methods for online action detection models is also included, highlighting their significance in ensuring the interpretability and accountability of the developed models. This organization allows for a coherent presentation of the diverse aspects involved in addressing the challenge of aggressive driving behavior detection.

2.1 Feature extraction

The extraction of features from image and video data is a fundamental task in computer vision. Early methods for feature extraction relied on hand-crafted descriptors such as SIFT [4], which required domain expertise to design and were limited in their ability to generalize across different datasets and applications. In recent years, differentiable architectures that can be trained on large-scale datasets have emerged as a powerful approach for feature extraction in computer vision [5]. These architectures, can automatically learn hierarchical feature representations of image and video data, leading to improved performance on a wide range of tasks, including the problem of action detection. The goal of feature extraction is to transform raw image or video data into a compact and meaningful representation that captures the salient information about the underlying objects or scenes [4, 5, 6]. The term "backbone" has been widely adopted in the deep learning literature to refer to the feature extraction module of a larger neural network. In this thesis, we also adopt this convention and use the term "backbone" to describe the feature extraction component of our proposed action detection model.

2.1.1 Backbone pre-training & fine-tuning

Various computer vision tasks employ somewhat similar architectures to reduce high-dimensional features into a more compact and low-dimensional representation. This process is prevalent regardless of the specific mechanisms, such as convolutional approaches used in deep convolutional neural networks (CNNs) [7, 8] or attention-based approaches applied in transformer models [9, 10]. The reduced feature representation is typically flattened and then fed as input to a task-specific network, such as a multilayer perceptron for classification [7, 8] or a recurrent neural network for sequence prediction [11].

This similarity in architecture allows for a wide range of models to be used as feature extractors, whether they rely on convolutional neural networks (CNNs), attention mechanisms, or other approaches. In particular, pre-trained models have become a popular choice for feature extraction due to their ability to reduce computational costs and accelerate the setup of experiments. By leveraging pre-trained models, researchers and practitioners can benefit from transfer learning and take advantage of the feature extraction capabilities already learned by the models on large-scale datasets.

Several pre-trained models have demonstrated potent feature extraction capabilities. For instance, VGG-16 [12] and ResNet-50 [13] are two widely used CNNs for feature extraction in computer vision tasks. VGG-16 is a 16-layer CNN that is known for its excellent accuracy on the ImageNet dataset, while ResNet-50 is a 50-layer residual network that achieves state-of-the-art performance on a range of tasks. In addition, other convolutional models like EfficientNet [14] and MobileNet [15] have been shown to provide efficient and effective feature extraction capabilities, making them popular choices for tasks with limited computational resources.

Alternatives from the domain of attention based models include the swin-transformer [9], or the Vision Transformer model (ViT) [10]. Google has announced a modified version of the Vision Transformer model that utilizes significantly more parameters than the original ViT model [16]. Specifically, the new model employs 22 billion parameters, which is several orders of magnitude larger than the 86 million parameters of the original ViT model. Following the scaling hypothesis [17, 18], as well as the results reported in the release-paper, this bigger model outperforms many previous models on various computer vision related tasks. Despite its impressive performance, this model will not be used in this thesis, due to its novelty and code-unavailability.

However, choosing the right backbone for a given problem is not only it's architectural design, but mainly the dataset it was pre-trained on. A popular choice for this is the aforementioned ImageNet dataset which consists of 14 million images of more than 20,000 categories [19]. In the field of action detection, numerous datasets have been developed to address various research interests and challenges. A significant portion of these datasets, such as UCF101 [20], HMDB51 [21], and Kinetics [22], are specifically designed for human action recognition. While these datasets have driven progress in their respective domains, their focus on human actions may not necessarily cater to other action detection problems, such as those involving driving data. Finding a suitable backbone pre-trained on driving data can be quite challenging, as the availability of such datasets is limited compared to those targeting human action recognition [23].

2.1.2 Convolutional Backbones

Convolutional Backbones rely on convolutional filters to perform feature extraction. Yann Le Cun et al. introduced convolutional filters to the domain of computer vision in 1989 [24]. While their potential for learning hierarchical features in images was recognized early on, the practical use of convolutional filters was limited by their dependence on large image datasets for training and high-performance computational infrastructure, such as graphics processing units (GPUs). As the availability of such resources improved in the 2010s, convolutional filters became an essential component of deep neural networks for computer vision tasks [12].

With their effectiveness in computer vision tasks involving image data established, convolutional filters were adapted for use with video data. Videos are a natural extension of images, introducing a temporal dimension to the spatial dimensions inherent in images. The combination of two spatial dimensions and one temporal dimension yields three-dimensional objects, which require three-dimensional convolutional filters. However, the addition of the third dimension results in a significant increase in computational complexity, making it exponentially more challenging to learn these filters compared to the two-dimensional case [25]. For instance, a 2D convolutional filter with a kernel size of 3 consists of nine parameters, while a 3D filter with the same kernel size consists of eighteen parameters. This increased complexity poses a significant challenge in developing effective methods for video analysis. The process of extracting video features by means of convolutional operations presents two further limitations. Firstly, each convolutional filter is restricted by its kernel size, which defines its receptive field. While smaller kernel sizes yield good results with regards to generalization, they restrict the receptive field to the mere notion of motion along the two spatial dimensions [12]. Secondly, and to address the first issue the information is gradually accumulated through pooling operations and stacked convolutional layers. While this delivers good results in practice it may still be ineffective and inefficient [26].

Despite these shortcomings, models based on the convolutional operation still achieve state of the art (SOTA) performance on many benchmark tasks, including image classification [27] and human action recognition [28]. Additionally, they are used as feature extraction modules for many attention based networks [29, 30, 31]

The fundamental architecture for convolutional neural networks (CNNs) remains relatively consistent across various computer vision tasks. In all the studies reviewed, stacked convolutional and

pooling layers are utilized to reduce the high dimensionality of the input, resulting in a feature representation with smaller dimensions. Typically, this representation is flattened and utilized as an input to a subsequent network that is specific to the task at hand (e.g., a Multilayer Perceptron (MLP) for classification).

This similarity allows for the easy adoption of different CNNs as feature extraction modules. By simply chopping of the task specific part, a feature extraction module can be obtained. Popular choices for such feature extraction modules in the domain of action detection are the Temporal Segment Network [32], which has been pretrained on a variety of tasks including the Activity-Net [33] challenge and Kinetics [34]. Alternatively the ResNet-I3D architecture [35] from the family of 3D CNN models, is also available, pretrained on a huge variety of models.

2.1.3 Attention Based Backbones

Attention mechanisms have gained increasing popularity in the field of computer vision, providing a powerful alternative to convolutional operations for feature extraction. These mechanisms are designed to learn and capture long-range dependencies and global context more effectively than local convolutional filters [36]. The attention mechanism operates by learning a distribution over the input data, rather than focusing on the raw input values themselves, allowing the model to selectively focus on specific regions of the input data that are relevant for a particular task. This ability to selectively attend to different regions of the input makes attention mechanisms particularly well-suited for handling sequences and structured data, such as video data, which inherently has a temporal dimension [29, 26].

Breaking image data into a sequence is a non-trivial challenge, as it requires the model to efficiently represent spatial information in a sequential format. Several approaches have been proposed to tackle this issue, such as flattening the input image into a sequence of non-overlapping patches [10] or utilizing hierarchical structures with shifted windows [9]. Despite these challenges, attention mechanisms have demonstrated impressive performance on various computer vision tasks, often rivaling or surpassing their convolutional counterparts [10, 9, 37, 26].

2.2 Action Detection

The task of action detection has a long-standing history in the field of computer vision, dating back to the early works on human activity recognition [38]. Generally speaking, the goal of action detection is to determine the start and end points of a given action in an untrimmed video. It must therefore be distinguished from the video classification task, which typically assumes that the entirety of the video is relevant for the classification. Action detection has important applications in various fields, including surveillance, sports analysis, human-computer interaction, and autonomous systems, among others [39, 40].

In contrast to video classification, where a single label is assigned to the entire video, action detection requires localizing the actions in time and potentially also in space. This localization may involve detecting actions within the video frame, segmenting the region of interest, and determining the temporal boundaries of the action [41, 42]. The challenges associated with action detection stem from various factors, including the diversity of actions, the variability in viewpoints, the presence of occlusions, and the complexity of background scenes [40].

Over the years, a variety of approaches have been developed to tackle the action detection problem. Early methods relied on handcrafted features, such as optical flow [43], Histogram of Oriented Gradients (HOG) [44], and Histogram of Optical Flow (HOF) [45]. These features were then used as input to machine learning algorithms, such as Support Vector Machines (SVMs) [45] or Hidden Markov Models (HMMs) [46], to classify and localize actions.

With the advent of deep learning, more recent methods have focused on leveraging convolutional neural networks (CNNs) [47], 3D CNNs [48], and recurrent neural networks (RNNs) [49] for action detection. These deep learning-based approaches can automatically learn hierarchical feature representations from large-scale video datasets, which has led to significant improvements in action detection performance. More recently, attention mechanisms and transformer-based models have been introduced to the domain of action detection, further pushing the state of the art in this challenging problem [30, 26].

2.2.1 Online Action Detection

The task of online action detection can be regarded as a specialized variant of action detection, where the model is restricted to using only past and present frames to make predictions. In more formal terms, given a sequence of video frames $F = f_1, f_2, \ldots, f_t, f_{t+1}$, the objective is to predict the action label y_t at time step t using only the information available in the frames up to time step t, i.e., $F_t = f_1, f_2, \ldots, f_t$. The model is not permitted to use any information from future frames, meaning that y_t can only be inferred from F_t and not from any f_i where $i > t$.

This constraint limits the model's ability to leverage the full temporal context of the action, which often contain crucial information about the action's progression and outcome [50]. This makes the task of online action detection more demanding, as the model must be able to accurately predict actions based on potentially incomplete information [51].

For the problem of detecting ADB, this constitutes a major challenge, since the appearance of aggression may be at the very end of the action. For example an overtaking maneuver could be entirely non-aggressive up until the final lane change, when the overtaking vehicle cuts in too closely, prompting the overtaken vehicle to break. To provide valuable predictions it would be desirable to potentially label instances as aggressive, before the real aggression takes place.

In the early stages of online action detection, the focus was on extracting handcrafted features, such as dense trajectories [52] and improved dense trajectories [53], to represent and analyze video data. With the emergence of deep learning, researchers explored more advanced models, including 3D CNNs [48] and two-stream CNNs [47], which effectively leveraged spatial and temporal information from video sequences. The adoption of recurrent neural networks (RNNs), particularly long short-term memory (LSTM) networks [49], further facilitated the modeling of temporal dependencies in video data. However, LSTMs faced limitations such as difficulty in capturing long-range dependencies and lack of parallelizability.

A method to address the limitation of inaccessible future frames is to *anticipate future frame representations*, employing these predictions to enhance the classification accuracy at the current frame f_t. This idea is inspired by the fact that humans consistently think about present actions by anticipating the future [54]. Typically, this approach employs an encoder-decoder architecture [30, 51]. In this architecture, the encoder is set up to learn frame representations that are most informative to the task. The decoder, on the other hand, is learned to provide predictions about future frame representations. Earlier works rely on LSTM and/or TRN cells [51, 55] to model temporal dependencies. The introduction of the online action detection transformer (OadTR) constitutes a shift towards the transformer architecture, where encoder and decoder both employ the multi-headed self-attention mechanism [30]. While this yields good results, computing MSA across all input frames presents a computational challenge, that can be elevated by using a different strategy called *consulting exemplars*. The idea of this approach is to utilise the fact, that examples of a category share characteristics and can therefore be described by representatives. This presents the problem of finding these representatives, which the authors of the colar paper (derived from consulting exemplars) solve clustering all examples of a single category and subsequently defining the example closest to the cluster center as exemplars. This yields representative exemplars per category [56]. The differences between the implementations however, exceed aforementioned conceptual differences. A more detailed overview of differences between these two models will be provided in section 3.

Additionally, online action detection necessitates real-time processing, adding an extra layer of complexity to the task. As a result, models designed for online action detection must balance computational efficiency with prediction accuracy to ensure practical applicability in real-world scenarios.

2.3 Explainability and Interpretability in Computer Vision

Deep learning models have gained widespread adoption in the field of computer vision due to their remarkable performance and ability to automatically learn complex features from large-scale data. However, these models are often referred to as "black-box" models, as their internal workings and decision-making processes can be challenging to interpret and understand [57]. This lack of explainability and interpretability poses significant concerns for stakeholders, particularly in high-stakes applications where transparency, trust, and accountability are crucial [58]. Moreover, the opaqueness of deep learning models can hinder the identification and resolution of biases or errors in predictions,

which may have far-reaching consequences [59]. Consequently, there is a growing interest in developing Explainable Artificial Intelligence (XAI) techniques that can enhance the interpretability of deep learning models in computer vision and facilitate human understanding of their underlying mechanisms [60]. In order to optimize the efficacy of the explanations generated, it is imperative to consider not only the specific problem at hand but also the characteristics and needs of the intended user [61].

A prevalent technique for gaining insights into algorithms within the computer vision domain involves the utilization of heatmaps, which highlight the most influential regions of an image for a particular outcome [62]. The bulk of heatmap generation methodologies can be categorized into two primary groups: gradient-based methods and attribution methods. Gradient-based methods leverage the backpropagation process, essential for training neural networks, to obtain gradients concerning each layer's input. The Gradient*Input method was the first to effectively implement this concept [63]. Subsequent methods, such as Integrated Gradients [64] and SmoothGrad [65], employ distinct averaging techniques to refine the resulting visualizations. These methods, however, are class-agnostic, indicating that it is challenging to discern which input region is accountable for individual classes when dealing with multi-label problems [62]. GradCAM [66] offers class-specific explanations by utilizing the gradients from the deepest layers and upscaling from these low-spatial dimensions. Although this can yield coarse outcomes, it enables the method's application to any convolutional neural network (CNN) type, including 3D convolutions [67].

In contrast, attribution methods generate explanations by recursively decomposing a network's decisions into the contributions from preceding layers, with the theoretical foundation laid by Deep Taylor Decomposition (DTD) [68]. Layer-wise Relevance Propagation (LRP) [69] is a notable implementation of this strategy, extended by Contrastive-LRP [70], Softmax-Gradient-LRP [71], and TIBAV [62], all employing different techniques to produce class-specific explanations.

Other methods, such as perturbation approaches [72], focus on the data itself. By introducing minor alterations to the data and monitoring the model's varying behavior, inferences about the decision-making process can be drawn. However, these methods are computationally demanding due to the requirement for separate model inference with each data perturbation. Shapley value methods [73] encounter similar issues.

The majority of explainable AI (XAI) approaches in computer vision are tailored to image input [62, 64, 65, 66, 72, 73]. Adapting these methods to video data is not a trivial task. Partially, because computational overhead challenges are exacerbated when incorporating a temporal dimension.

2.4 Understanding Driving Behaviors

Driving behavior is influenced by a multitude of factors such as personality traits, age, experience, gender, distraction, weather conditions, and the influence of alcohol and drugs. Studies have shown that personality traits such as aggression, sensation-seeking, and anxiety can affect driving behavior [74]. Furthermore, age and driving experience play a significant role in risk-taking, decision-making, and reaction times [75]. Gender differences have also been identified in driving behavior, with variations in risk-taking, aggression, and accident rates between men and women [76, 77]. Driver distraction, particularly due to mobile phone usage and in-vehicle technologies, has been recognized as a major factor contributing to traffic accidents [78]. Weather conditions, including rain, fog, and snow, have been found to significantly impact driving behavior and traffic safety [79]. Lastly, the detrimental effects of alcohol and drugs (i.e. cannabis) on driving performance and safety have been extensively documented [80]. Understanding these various factors is crucial for developing effective interventions and policies aimed at improving road safety. As Interactions with other road users are a fundamental aspect of driving, accurately estimating the behaviors of other traffic participants is crucial for any autonomous vehicle (AV) system. Despite the human ability to quickly and intuitively predict behavior, accurately modeling human behavior remains a key challenge for the successful implementation of AVs [81, 82].

As AV technology continues to develop, there is a pressing need for sophisticated algorithms that can accurately model human behavior in real-time and respond appropriately in unpredictable situations.

2.4.1 Aggressive Driving Behavior

Various methodologies have been proposed to assess the aggressiveness of driving behavior, many of which have emerged from the fields of sociology and traffic modeling [83, 84]. The vast majority of these metrics rely on survey data, such as the Driving Anger Scale, which employs a 5-point Likert scale to measure the extent of anger experienced in a given situation [85]. This scale has been expanded by the Driving Anger Expression Inventory, which identifies four distinct modes of driver anger that can be aggregated to yield the Total Aggressive Expression Index [86]. While these approaches offer valuable insights into the sociological and psychological underpinnings of aggressive driving, their applicability to autonomous driving is limited, as they primarily focus on the individual driver. In the case of AVs, the challenge becomes identifying aggressive driving through the vehicle's driving style, rather than attempting to understand, avoid or mitigate driver aggressiveness.

One approach to evaluating the aggressiveness of a given driver through the perception of their driving style was proposed in 2020 under the name **CMetric** [86]. This approach follows the intuition that aggressive driving is constituted by the distance to other road participants, as well as the number of such participants immediately surrounding the vehicle in question. In other words, aggressive driving does not occur on an empty road. To facilitate this, the authors first developed a Dynamic Geometric Graph, which was subsequently used to compute closeness centrality and degree centrality. These two centrality measures were calculated at each time step for each vehicle. Subsequently, these measures were used to compute the style likelihood estimate and the style intensity estimate, which were intended to answer the question of how likely a given driver is to exhibit aggressive driving and how pronounced this behavior is. With a classification accuracy of more than 90%, these findings are a meaningful contribution to the problem of understanding aggressive driving behavior. To measure the usefulness of the classifications provided, the authors conducted user studies, where they compared the time at which humans and CMetric determined the presence of aggressive driving behavior. These findings revealed that CMetric lagged 0.23 - 1.28 seconds behind the human control. At 50 km/h, a lag of 1.28 seconds translates to approximately 17.7 meters, which can have catastrophic effects in the fast-paced environment of urban traffic. Despite the excellent accuracy, there are some hard limitations that are not easy to overcome. For one, the proposed metric only works on four different types of aggressive driving behaviors, which are *Overspeeding*, *Overtaking*, *Sudden Lane-Change*, and *Weaving*. The authors indicated that the algorithm could be easily extended to *Tailgating*. However, since distance measures are so foundational to the approach, it is difficult to envision how this metric could be applied to aggressive driving styles such as *Rule Breaking* or *Cutting*, which have no indication in closeness whatsoever. Additionally, the computations to determine the centrality values must be run for every agent in the frame. It is therefore questionable if the premise of real-time execution can be maintained for large numbers of actors. Moreover, the authors conceded that CMetric only works if at least one other actor (other than the EgoVehicle) is present [86]. CMetric has been successfully applied to generate traffic simulations featuring aggressive drivers [87].

Another important aspect of understanding aggressive driving behavior is to establish a comprehensive list of behaviors, which are considered aggressive. Drawing from the literature, a variety of studies and sources have provided definitions of aggressive driving behaviors that can serve as a foundation for further research. In particular, [3] and [88] propose the following behaviors as indicative of aggressive driving:

- Tailgating
- Weaving in and out of traffic
- Improper passing (e.g., cutting in too close in front of a vehicle being overtaken)
- Passing on the road shoulder
- Improper lane changes (failure to signal)
- Failure to yield the right of way to other road users
- Preventing other drivers from passing
- Unwillingness to extend cooperation to motorists unable to merge or change lanes due to traffic conditions

- Driving at speeds far in excess of the norm which results in frequent tailgating, frequent and abrupt lane changes
- Running stop signs
- Running red lights
- Not respecting traffic regulations

By incorporating these behaviors into the analysis, researchers aim to develop a more robust understanding of aggressive driving that can be applied to the detection and assessment of such behaviors in autonomous vehicles.

2.5 Datasets

A considerable number of recent datasets have been created for applications in the domain of autonomous vehicles [89, 90, 91, 92]. Nevertheless, since most of these datasets are tailored towards tasks such as scene segmentation, object recognition, or vehicle trajectory analysis, they rarely include labels for behavioral characteristics. To address this limitation, some studies have employed crowdsourcing strategies for the annotation process, effectively expanding the available labels for a given dataset to encompass driving behaviors [86].

2.5.1 METEOR Dataset

The METEOR Dataset is a publicly accessible video dataset containing driving scenes from the driver's perspective under various traffic conditions. Since its introduction, the dataset has been cited by 7 subsequent publications, according to Google Scholar. The majority of these publications introduce different datasets [93, 94, 95] or concentrate on behavior prediction using graph models [87, 96]. The limited number of published works utilizing this dataset could be partially attributed to its relatively recent publication in 2021.

Comprising approximately 100GB of traffic video data, the METEOR dataset shares similarities with the HDD dataset in terms of capturing the driver's viewpoint [92]. However, the METEOR dataset provides more comprehensive annotations that align with traffic and scene understanding. Each frame is categorized based on various attributes, such as behavior, traffic density, area, and lighting conditions. A complete overview of all labels featured in the dataset can be found in Figure A.1. Unfortunately, a significant discrepancy exists in the frequency of different annotations within the dataset, with the "behavior" category displaying a highly imbalanced class distribution. This imbalance presents a challenge for detecting less frequent actions but is also well-suited for identifying aggressive driving behavior, as aggressive actions like rule-breaking are rare in ordinary traffic. Further details on label distributions and their influence on the experiments are discussed in Section 3.

Moreover, the video frames can be densely populated, with up to 40 agents from as many as 9 different categories per frame. Research conducted by the dataset authors demonstrates that scene heterogeneity, including agent diversity, weather, and road conditions, reduces object detection performance by a factor of 6-8 compared to benchmarks [91]. These findings emphasize the challenges associated with accurately detecting behaviors in complex, real-world environments, underscoring the necessity for enhanced methods and models that can accommodate such heterogeneity.

Chapter 3

Methodology

3.1 Research Design

The primary goal of this research is to determine if and how aggressive driving can be assessed using attention-based deep learning models. To do so, this study seeks to answer three main primary questions:

1. Which sampling approach of video data yields the best results in the context of transformer models?
2. Which attention-based model architecture is best suited for the task?
3. How can the model's explainability be improved by identifying salient cues for driving behavior assessment?

The findings of this study will serve as a stepping stone for the automotive industry, particularly for autonomous vehicle development, as such vehicles could use information about aggressive driving agents to alter their future interactions with these agents.

The dataset used in this research is approximately 100GB of video data from the METEOR Dataset. It contains specific types of aggressive driving behavior, which were determined by refining the initial seven labels, present in the data, to four. The final labels used in this research are *Background, Over-Taking, LaneChange, WrongLane, Cutting*, with Background denoting the absence of labels encoding aggressive driving behavior.

Deep learning models were chosen for this research due to their widespread adoption in contemporary related work and their ability to automatically extract features, handle large datasets, and learn complex patterns. As a data science major, employing deep learning frameworks is a natural choice, aligning with the techniques and methodologies prevalent in the field (see Related Work section for more details).

To answer the first research question, four different backbones, or base models for feature extraction, were tested. Two of which work on a frame based level, returning one feature vector per input frame. The other two process video in clips, where a clip is made up of 6 or 8 frames respectively. Additionally, these two different approaches represent the utilization of different frame rates, as the frame level features are based on 15FPS, while the clip level features are generated on 30FPS.

Research question two is investigated by testing two attention based model architectures and evaluating their performance. Additionally, these two distinct architectures were aggregated, resulting in a third model that was evaluated on the online action detection task.

To answer the third research question, an occlusion based approach was employed. This approach excludes one agent at a time from the video and compares the resulting classification output to the classification of the original video. This strategy results in local explanations, providing an understanding of the influence of individual agents on the overall classification.

Beyond the experiments aimed directly at answering the research questions, a series of ablation studies were conducted to investigate the influence of model size, different numbers of encoder/decoder layers, dropout values and other factors on the overall performance of the models.

To alleviate the computational burden of the training process, this research uses feature extraction models, pre-trained on different datasets, including the ActivityNet challenge [33] and the Kinetics dataset [34]. Even though these fine-tunings are not driving-specific, they were chosen for their wide adoption in literature, including the original implementation of the two online action detection models [56, 30], evaluated in this study.

Apart from the RGB-videos provided in the METEOR Dataset, no additional features or data sources were incorporated into the research. The models first use a feature extraction module to extract features from videos, and different models are subsequently trained based on these extracted features. The final classification is made solely based on the information extracted from video frames.

By carefully designing the research methodology and employing various deep learning model architectures best suited for analyzing video data of driving behavior, this study aims to provide valuable insights into assessing aggressive driving behavior using attention-based models. The findings of this research have the potential to contribute significantly to the development of safer and more efficient autonomous vehicles.

3.2 Dataset and Label Refinement

As outlined in section 2, the METEOR driving dataset offers annotations for a large selection of rare and unusual behaviors, that are rarely featured in other driving datasets. Figure A.1 provides a comprehensive overview, which behaviors are present in the data. While some of these behaviors can hardly be considered aggressive others are explicitly described as aggressive in the accompanying publication [91]. On average, all behaviors annotated in the dataset are shorter than 3 seconds, implying, that at a framerate of 15 frames per second (FPS) $3 * 15 = 45$ Frames are enough to capture the entirety of most action.

The final decision to use the METEOR driving dataset for this thesis was primarily driven, by its representation of rare and aggressive behaviors, as well as it being underrepresented in scientific publications.

The publication introducing the dataset also provides a benchmark for the task of **offline** action detection. While online action detection is a more challenging problem, the model used for this could potentially be used to extract features from the video given that it is fine-tuned on the data. However, the link provided by the authors of the dataset, pointing to a google drive, does not work as intended, meaning the pre-trained model is not accessible.

3.2.1 Deriving Labels Encoding ADB from Meteor Dataset

To train models capable of detecting aggressive driving behaviors, we mapped the behaviors established as aggressive (based on the definitions from related work) to the labels native to the Meteor dataset [91].

Establishing equivalents between these two lists is not as easy as it would appear, since the METEOR paper defines some behaviors as explicitly aggressive. A prime example of this is Lane Changing. Generally, this would not be considered aggressive behavior. However, the METEOR paper states that the label *"Lane change w. lane markings"* is given, *"when agents **aggressively** change lanes on roads with clear lane markings."* [91].

To best utilize the given dataset, it was decided that labels explicitly affirming the existence of aggression should be considered alongside labels matching the definition from [3]. Lastly, the METEOR label *cutting* was approved for the final list of labels. The METEOR paper states that cutting represents an agent of interest interrupting the road crossing of a slower entity (e.g., bicycle, pedestrian, etc.)

To convert agent-level labels to frame-level, each frame is assigned a binary label based on the actions of all agents within it. If at least one agent is performing a specific action, the frame is labeled as 1 for that action; otherwise, it's labeled as 0. This is necessary, because action detection models typically require frame-level annotations.

It is noted that these actions are independent of one another, implying that an agent can show multiple aggressive actions at the same time. This is notable, because it creates a multilabel case, where different labels can co-exist at the same time. This has far reaching consequences for model implementation and training.

Behavior	Reason for inclusion
Overtaking	encodes aggression as mentioned in METEOR paper
Overspeeding	part of a common definition for ADB
Lane change w. or w/o. lane markings	encodes aggression as mentioned in METEOR paper
Running a traffic light	part of a common definition for ADB
Wrong Lane	part of common definition for ADB
Wrong Turn	part of common definition for ADB
Cutting	part of a common definition for ADB

Table 3.1: labels encoding ADB with the reason for inclusion

3.2.2 Label Refinement

Initial experiments were conducted with all available agents and actions in the dataset. However, due to poor results, the scope of the experiments was iteratively narrowed. At first the number of actions was reduced based on their occurrence in the dataset. As can be seen in table A.1, the number of frames featuring interesting behaviors as well as agents varies greatly. It was assessed that the poor model performance could be (partially) attributed to the wide range of features needed to encode all different agents/actions. This is supported by the argument that an aggressive action is typically performed in relation to another agent/objet (Overspeeding being the exception to the rule). As each agent can perform each aggressive action against any other agent, this leads to $N^2 * A$ combinations of aggressive behavior where N is the number of agents and A is the number of aggressive actions. While this is an upper bound for possible combinations and the dataset provides no real way to determine who these actions are committed against, it can be seen in table A.1 that some agents are more likely to act aggressively than others. Based on these findings it was decided to only include actions performed by the actors *Car, MotorBike* and *Scooter* into the final labels, since they were the most present among the aggressive actions. A more detailed overview of the actions per agent class can be found in table A.1 in appendix A.

Additionally, the labels *LaneChanging(m)* and *LaneChanging* were merged into a single label. The rationale behind this decision is that the primary distinction between these two labels lies in the characteristics of the road (presence or absence of lane markings), rather than any difference in the actual lane-changing behavior [91]. Given that the feature extraction backbones employed in this research were not fine-tuned to specifically identify such subtle differences in road conditions, it was hypothesized that the models may struggle to accurately differentiate between these two labels.

Lastly, a background label was added, as is standard practice in almost all recent works on online action detection [97]. Background is defined as the absence of all considered actions.

3.3 Feature Extraction

3.3.1 Choosing Pre-trained Backbones Over Custom Pre-training

In this research, backbones pre-trained on benchmark datasets are exclusively employed, as opposed to fine-tuning them for the specific task. Practical challenges, such as the slow input/output infrastructure in relation to the data size, hindered the fine-tuning process, rendering it both time-consuming and inefficient. Since these obstacles could not be overcome, fine-tuning was not a viable option. However, several factors indicate that using pre-trained models is an acceptable alternative for this study. Firstly, adopting pre-trained backbones ensures compatibility with the original publications by using the same models and pre-training checkpoints, thus providing a consistent foundation for comparison. Secondly, the absence of specialized backbones designed for driving data further supports the choice to rely on existing pre-trained models. Consequently, the decision to use pre-trained backbones was deemed the most appropriate approach for this research, despite the potential benefits of fine-tuning.

3.3.2 Feature Extraction Characteristics

This research aims to compare different feature extraction pipelines. In total, four feature extraction approaches were evaluated, which can be categorized based on various criteria.

A key distinction between feature extractors is whether they provide frame-level or clip-level features. Frame-level features generate a feature vector for each input frame, while clip-level features yield a single feature vector for a sequence of consecutive frames. Consequently, clip-level features encode more temporal context, while frame-level features primarily capture spatial information, delegating the temporal encoding to the model itself. Since the feature extractors were not fine-tuned for the METEOR dataset, and clip-level feature extractors were optimized for human action recognition, it was hypothesized that more general feature extractors, fine-tuned on image classifications from diverse domains, would deliver improved performance. This hypothesis is grounded in the belief that broader pre-training on varied data results in more robust and versatile features than those derived from specialized, domain-specific data.

Frame-level feature extractors can be further classified based on their use of convolutional filters or attention mechanisms. Attention-based models have recently demonstrated competitive results compared to their convolutional counterparts, often outperforming them. Therefore, attention-based models were included in this study. However, attention-based models encoding clip-level features were challenging to obtain and were not incorporated in this research.

Feature extractors can also be distinguished based on whether they incorporate optical flow estimation. For frame-level feature extractors, this involves extracting optical flow features between two consecutive frames. To ensure the model does not have access to information about upcoming frames, optical flow at time T_0 is calculated between frames T_{0-1} and T_0. This necessitates disregarding the first frame of each video. The clip-level feature extractor outputting optical flow computes features based on the displacement between six consecutive frames.

Frame-level and clip-level features were extracted with different framerates. While the clip-level features were extracted at 30FPS, the frame-level features used 15FPS. The rationale behind utilizing different frame rates for extracting frame-level and clip-level features primarily revolves around ensuring operational efficiency and maintaining the integrity of the action representation. We opted for 15FPS for frame-level features in order to prevent significantly increasing the time disparity between the two feature levels. This is essential as it facilitates coherence in temporal information across different feature extraction levels. The chosen frame rate also allows for a sufficient time frame to accurately depict the majority of actions, which is crucial for reliable feature extraction. Implicitly, this arrangement offers a unique opportunity to evaluate the impact of different frame rates on the performance of the overall model.

Indeed, clip-level features intrinsically demand a larger quantity of frames to construct valid examples. As elaborated in section 3.4, a valid example is composed of 64 consecutive time-steps. Instituting a 15FPS extraction rate for clip-level features would inevitably result in a considerable diminution in the number of available instances for training and testing. In order to ensure the durability of the learning process and to maintain an adequate sample size for both training and testing phases, clip-level features were deliberately extracted at a more accelerated frame rate of 30FPS. This not only optimized the quantity of valid examples, but also facilitated a more comprehensive data representation, thereby promoting robustness and accuracy in the model's performance.

3.4 Models for online action detection

3.4.1 OadTR

As outlined in section 2, the Online Action Detection Transformer (OadTR) represents a significant advancement over previous designs, incorporating the transformer architecture into the online action detection task [30]. This architecture is primarily composed of two key components: the encoder and the decoder.

It is important to note at the outset that the implementation of the OadTR model is complex, characterized by a multitude of modifiable hyperparameters. While the model's GitHub repository can be cloned with relative ease, the process of adjusting these hyperparameters can be challenging due to their sheer number and occasionally ambiguous naming conventions. This complexity should not be seen as a deterrent, but rather as an indication of the careful navigation and understanding required when working with this model.

The forthcoming discussion aims to provide a comprehensive guide to the OadTR model, detailing its architecture, implementation, and the specific modifications made for the purpose of this study.

This detailed methodology is intended to serve as a basis for those interested in replicating or building upon these experiments.

OadTR consists of two main components, the encoder and the decoder, both of which are inspired by the standard encoder-decoder architecture of current transformer models. As such they employ multi-headed self attention.

The encoder incorporates a class token to learn global features pertinent to the online action detection task. This is done to ensure that the final feature representation is not disproportionately influenced by the initial value of a specific feature.

Conversely, the class token, via its adaptive interplay with other tokens within the encoder, can yield a semantic embedding more fitting for feature representation and for capturing the global context of the video sequence. Empirical support for these theoretical postulates is provided by experiments showing that the feature similarity for $t = 0$ dramatically diminishes before and after a pass through the network when a class token is employed.

While the encoder's responsibility lies in accurately capturing temporal dependencies and relations, the decoder's function can be unerstood, as predicting the future of $t = 0$, achieved via learnable predictive queries. To optimally leverage semantic information from the encoder, the decoder incorporates the encoder-decoder cross-attention mechanism.

An important difference to the original Vision Transformer is that the OadTR implementation permits the parallelization of the prediction queries' decoding at each decoding layer [30].

The initial implementation of OadTR is tailored to a multiclass problem, rendering it unsuitable for the METEOR dataset. To adapt it to multilabel problems, a few modifications were required. The most significant adjustment involved updating the loss function. Following initial trials, the PyTorch implementation of BCEWithLogitsLoss was employed [98]. As can be seen in equation 3.1, this loss function amalgamates binary cross-entropy loss with a sigmoid layer, offering an additional advantage of eliminating the necessity of the softmax operation implemented by OadTR for the final classification outputs and the supervised training of the decoder.

$$\ell(x, y) = \frac{1}{N} \sum_{n=1}^{N} -w_n \left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \right] \tag{3.1}$$

For optimization, Adam was utilized in its original form, as implemented by PyTorch. A notable divergence from the OadTR implementation was the use of a CosineAnnealingLR scheduler, which allows for dynamic updates of the learning rate during training. This modification was considered necessary as the original OadTR implementation employed a StepLR scheduler, which is significantly influenced by the step-size. This is a hyper-parameter that would have necessitated substantial effort to optimize. The revised approach thus aimed to make the training process more efficient, without compromising on model performance.

In terms of regularization, two distinct strategies were employed. The first involved the use of the *weight_decay* parameter within the Adam optimizer, and the second incorporated dropout layers dispersed throughout the network. Furthermore, the size of the model itself served as an additional regularization factor. This multi-faceted approach to regularization aimed to prevent overfitting and enhance the generalization capabilities of the model.

3.4.2 Colar

The second model employed for online action detection is the Colar Model [56]. This model was chosen for several reasons. Firstly, its novelty ensures that it incorporates the latest advancements in the field. Secondly, it has a lower computational footprint compared to the OadTR model, making it more efficient for processing large amounts of video data. Thirdly, the Colar Model introduces an innovative approach to action detection by consulting exemplars, or representative examples of each action category. Finally, the model is straightforward to implement, which, while not a primary consideration, is a beneficial feature. An overview of the conceptual differences between the OadTR model and the Colar model can be found in figure 3.1

The Colar Model operates by comparing each frame in a video sequence with both previous frames and category exemplars. This dual comparison is facilitated in distinct branches, named dynamic branch and static branch.

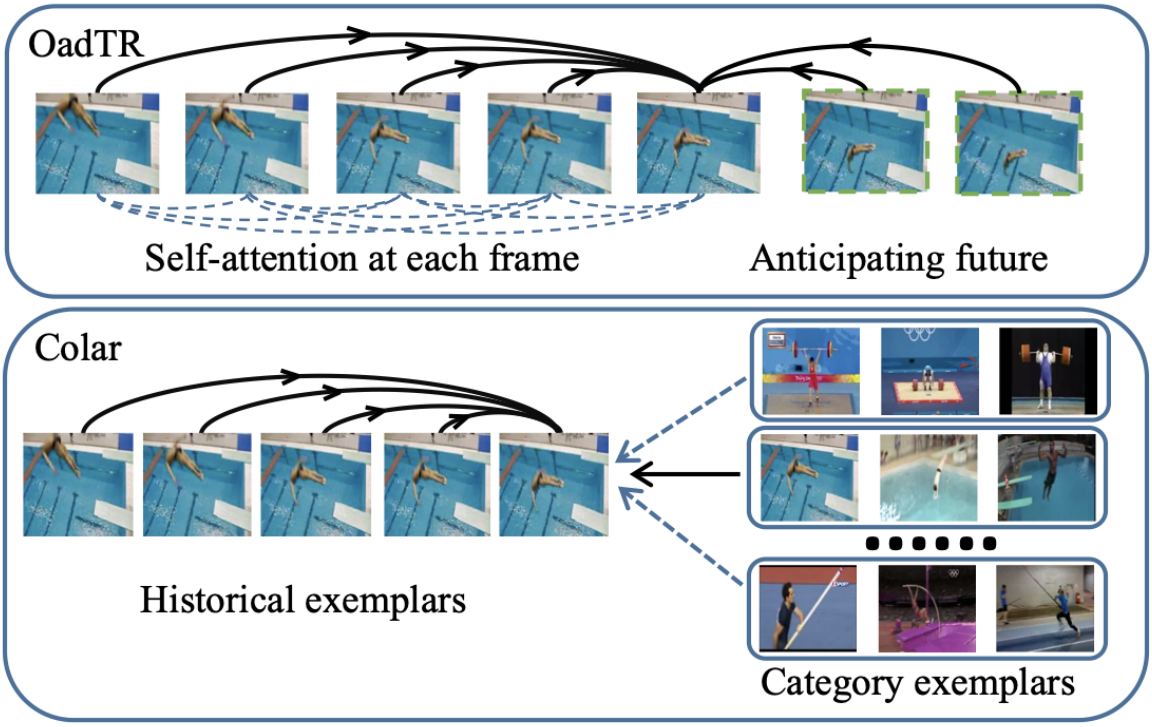


Figure 3.1: Illustration of the conceptual distinction between OadTR [30] and Colar [56]. Adapted from the original representation in [56].

The dynamic branch models the temporal process of actions. It does this by comparing each frame with its predecessors. During this process, local features are aggregated to obtain a combined feature representation of the entire input sequence. To facilitate this aggregation each video feature is transformed into a key space and a value space. The resulting keys are used to compute similarities, while the values are utilised for feature aggregation. The affinity between each frame and its preceding frames is then measured, and these affinity values are used to aggregate the value features of the preceding frames, yielding a historical feature. This historical feature is combined with the value feature of the current frame to perform online action detection.

The static branch, on the other hand, is designed to capture the unique characteristics of each action category. It does this by comparing each frame with a set of representative exemplars for each category. These exemplars are selected based on a simple clustering approach where instances closest to the cluster centers are used as exemplars.

To measure similarity between the current frame and the exemplars, transformations into the key and value space are performed, similarly to the dynamic branch. The affinity between the current frame and each exemplar is then measured, and these affinity values are used to aggregate the value features of the exemplars, yielding a category-specific feature. This category-specific feature is combined with the value feature of the current frame to perform online action detection.

The original Colar implementation fuses predictions from dynamic and static branch according to the following formula: $s = \beta\hat{s}^s + (1-\beta)\hat{s}^d$, where $\hat{s}^s$ are the predictions from the static branch and $\hat{s}^d$ are the predictions from the dynamic branch. Ablation studies found this parameter beta to be optimal at 0.3. However, in order to make the fusion of the final predictions learnable as well, a different implementation was also tested, where the predictions from static and dynamic branch are concatenated along the first dimension and then fed into a MLP to agregate a final prediction score per class.

Since the Colar model was implemented for a multiclass problem and does not natively support the multilabel case, some modifications were nescessary. Similarly to the OadTR model, the most significant of which was the implementation of the BCEWithLogitsLoss [98]. Other modifications regarded the replacement of Softmax Layers with Sigmoid layers, for both the final classification and

the measure of similarity within the static branch.

For optimization, Adam was utilized in its original form, as implemented by PyTorch. To fascilitate regularization, the *weight_decay* parameter of the optimizer was used. As for OadTR, a CosineAnnealingLR scheduler was used to reduce the learning rate over the course of the experiments. This followed the same intuition as for OadTR.

3.4.3 Combining OadTR and Colar

The exploration of a comprehensive approach to online action detection in this study involves the integration of the static branch of the Colar model with the OadTR model. The static branch of the Colar model has demonstrated its proficiency in capturing category-specific features by comparing each frame with a set of representative exemplars. Concurrently, the OadTR model, utilizing the transformer architecture, excels in capturing long-range temporal structure, encoding historical information, and predicting future frame representations.

The integration of these models is designed to harness the unique strengths of each. The static branch of the Colar model provides a robust mechanism for capturing category-specific features, thereby enhancing the model's ability to distinguish between different action categories. The OadTR model contributes its efficient and effective handling of temporal dependencies, enhancing the model's ability to track the evolution of actions over time.

The point of integration is when the features from the encoder and decoder are merged in the OadTR model. In the original OadTR model, the task-related features in the encoder are concatenated with the pooled predicted features in the decoder. This is then passed through a fully connected layer and used for action classification. In the integrated model, the concatenation operation is adjusted to include the predictions from the static branch of the Colar model.

This integration is expected to yield a model that is both powerful and efficient. The computational efficiency of the OadTR model can help balance the computational demands of the static branch of the Colar model, resulting in a model that is capable of processing large amounts of video data without sacrificing performance. This approach is anticipated to provide a more nuanced understanding of online action detection, making it a promising direction for further research and experimentation.

Following the notations from the Colar and OadTR paper, equation 3.2 is derived, explaining how the final prediction p_0 is made. A visual representation of the proposed architecture can be found in figure 3.2

$$p_0 = \text{sigmoid}(\text{Concat}[m_N^{\text{token}}, \tilde{Q}, \hat{s}^s]W_c) \tag{3.2}$$

In this Equation the parameter m_N^{token} represents the task-related features in the OadTR encoder, $\tilde{Q}$ depicts the pooled-predicted features from the OadTR decoder and $\hat{s}^s$ is the output of the static branch of the Colar model. W_c are the parameters of a fully connected layer that is used to combine features.

3.5 Model training and evaluation

As previously elaborated, the process of model training and evaluation involved several steps, starting with feature extraction using various backbones for analysis. The extracted features were stored to enable direct access for the online action detection model, thereby speeding up the training process and allowing for a larger batch size as only the action detection model needed to fit on the GPU.

The dataset was split randomly according to video names using 80% for training and 20% for testing. This approach prevented scenarios where the model would use timesteps for evaluation that were also used for training, thus preventing data leakage. Despite the split being done without regards for the number of frames per category in each video, figure A.2 shows, that the result approximates the intended 80/20 ratio.

The hyperparameters and regularization techniques varied for different models and were discussed in the introduction of the model architectures.

To address the issue of class imbalance among the different labels, class weights were applied in the loss function. This approach assigns a higher penalty for incorrect predictions in the minority class, thereby making the model pay more attention to these examples during training. The different feature extractors use different framerates, and/or combine multiple frames into a single feature vector. Hence,

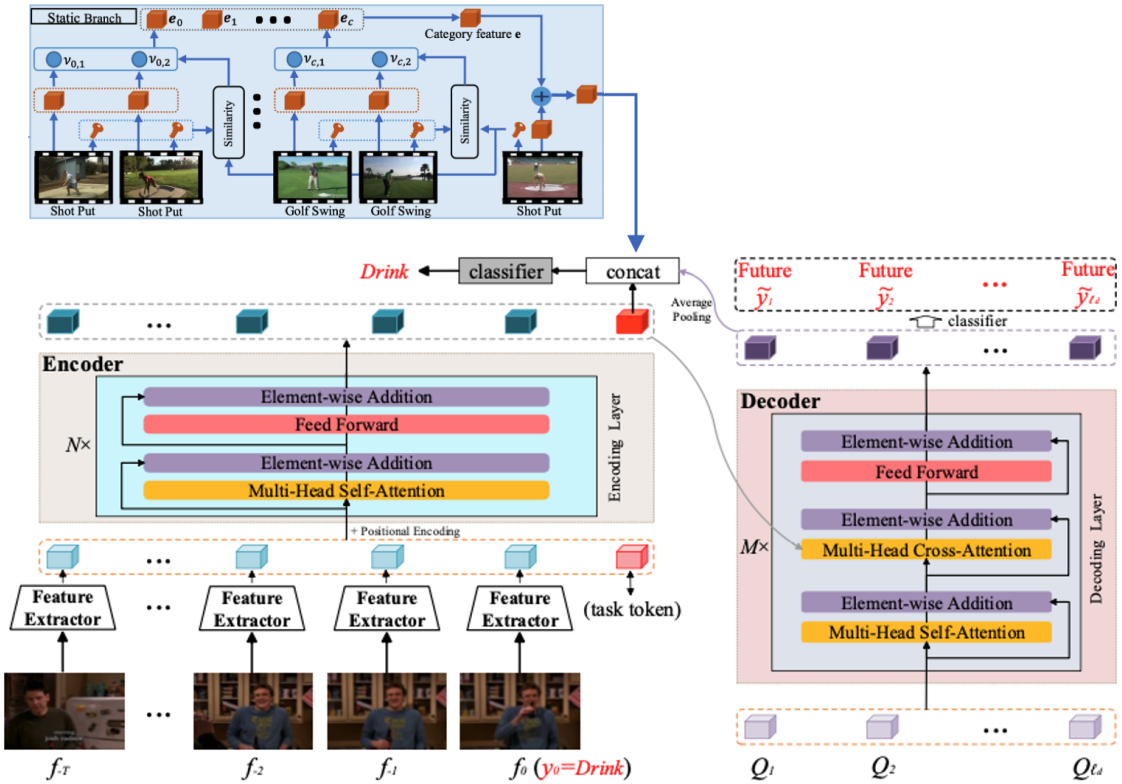


Figure 3.2: Proposed new architecture for combination of OadTR and Colar. Adapted from the original figures in [30] and [56]

requiring feature specific weights. To comply, the weights were calculated dynamically after loading the features, by using this forumla: $\frac{\text{Frames that don't belong to that category}}{\text{Frames in that category}}$, as introduced in the PyTorch documentation [98].

Model performance was evaluated using the ROC-AUC score to assess how much better the model performed compared to random classifier. Additionally, the metrics F1 score, precision and recall are used to compare the performance of different model configurations. These metrics are used as implemented by sklearn [99]. To reduce the impact of the class imbalance on the metric results, the parameter *average* was set to *weighted*. Based on this, the support of each class is used as a weight, so that the average is calculated according to the class distribution.

Additionally, mean average precision (mAP) is used as well. Mean Average Precision (mAP) is a widely used metric, computed by taking the mean of average precision per class.

Furthermore, multilabel confusion matrices were generated and used to evaluate the model performance as well.

Model training and evaluation were performed on a NVIDIA Tesla V100 - 32GB GPU provided by the Data Science Research Infrastructure (DSRI) hosted at Maastricht University.

3.6 Identifying salient cues

The identification of salient cues in video data is a crucial aspect of this study. This process aims to discern the influence of specific regions or agents within the frame on the final classification outcome. To facilitate this, a masking technique is employed, which involves setting parts of the input video to zero, effectively excluding certain agents from the frame.

A total of $n+1$ videos are generated for this process, where n is the number of agents in the frame. One video is left unmodified to serve as a baseline for comparison. For each agent in the frame, a separate video is generated where the agent is masked, i.e., excluded in both spatial and temporal dimensions. Each of these videos is then used to make a prediction.

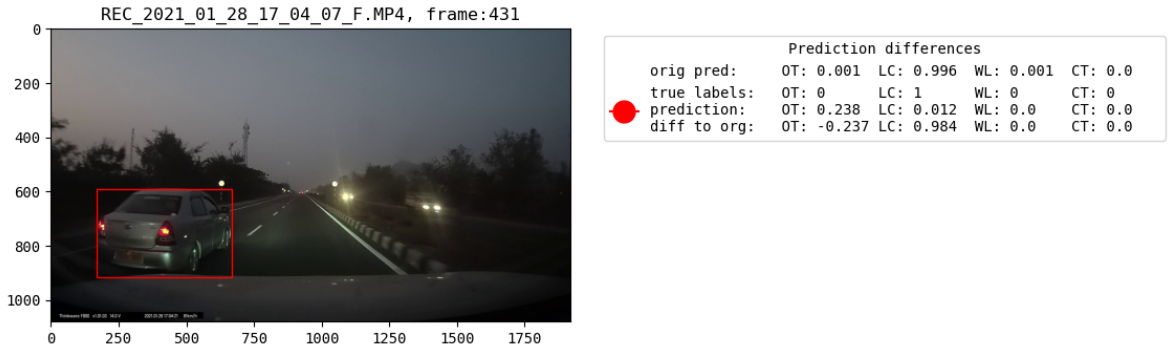


Figure 3.3: Example for the identification of Salient Cues

The prediction results from the masked videos are compared with the baseline prediction. This comparison is performed by subtracting the prediction from the masked video from the baseline prediction. If the result is greater than zero, it implies that the exclusion of the agent makes the classification less likely, suggesting that the agent may be associated with the category in question.

$$\text{Influence}_{\text{agent}_i} = \text{classification(baseline)} - \text{classification(video excluding agent}_i)$$

This approach provides a mechanism to identify salient cues in the video data that contribute to the classification outcome. However, it should be noted that this method may produce misleading results in cases where agents occlude each other, as can occur in the dataset.

This process can be seen in figure 3.3. The original prediction for this scene has LaneChange at 0.996. However, after occluding the red agent, this prediction drops to 0.012, which leads to $\text{Influence}_{\text{agent}_i} = 0.984$.

While the use of occlusion or masking to provide insights into video data and online action detection is not a common approach, it is related to other techniques in the field of computer vision and machine learning. For example, the use of key-frames in action recognition and the use of 3D and 4D data for facial expression recognition aim to capture salient features or cues in the data. This study extends these concepts by applying a masking technique to identify the influence of specific agents on the classification outcome. This occlusion based approach can be used to provide local explanations.

These local explanations afford tangible means to enhance the performance of the model.

Through the identification of specific regions or agents that lead to consistent misclassifications, training data can be enriched with more examples of these instances, thereby improving model comprehension and prediction accuracy. This could not only help uncover biases and errors in the predictions/data, but also allow for a tailored data pre-processing pipeline involving augmentations to synthetically inflate the number of training examples for most error-prone categories.

Besides modification involving the trainig data, local explanations can also help better understand pottential issues in the model architecture and training process. Especially with regards to loss function and regularization techniques.

In conclusion, the local explanations provided by this occlusion-based approach offer invaluable insights for targeted improvement of the model's performance, aligning with the primary goal of building robust and interpretable models for aggressive driving behavior assessment. Additionally, these explanations hold significant potential for applications in the domain of autonomous vehicles, specifically in identifying agents that are responsible for a positive classification.

Chapter 4

Experiments and Results

4.1 Experimental setup

OadTR The objective of maintaining uniform experimental conditions necessitated the use of a consistent hyperparameter framework, which is comprehensively outlined in appendix B. As a different learning rate scheduler was used, the parameters *lr_drop* and *lr_drop_size* are of no importance to the experiment setup. In the interest of ensuring a comprehensive understanding and facilitating seamless replication, these parameters have been reported nonetheless. Certain parameters, exemplified by *weight_values* and *thumos_data_path*, are defined at runtime. Further information about the parameters is available in the *custom_main.py* file located within the attached code repository.

Efficient utilization of memory resources and rapid convergence were achieved by inflating the *batch_size* from 128 to 512. While this adjustment should minimally impact the overall results of training, it significantly enhances computational efficiency [100].

To ensure fair comparison with the Colar model, the *enc_layers* parameter was incremented from 63 to 64. Contrary to what the nomenclature may suggest, this parameter is not indicative of the count of encoder layers within the model. Instead, it is instrumental in defining the number of time-steps that contribute to a given prediction. In the context of frame-level features, this corresponds to the number of video frames. While the OadTR baseline employs 63 time-steps, the Colar model operates with 64.

The *dim_feature* parameter plays a crucial role in enabling swift adaptation to new experimental scenarios by matching the actual dimensionality of the features. In the event of an implementation that incorporates optical flow, the formula *dim_feature = rgb_feature_dim + flow_feature_dim* needs to be applied.

Despite the fact that the lion's share of available hyperparameters were maintained in their original state, certain parameters were introduced specifically to tailor the experiments to the dataset and dataloader. These additions, while beneficial in this context, may not be universally applicable across diverse use cases.

Colar As can be seen in appendix B, there are much fewer hyperparameters for the Colar model than for the OadTR model. The modifications made are specific to dataset and available hardware and bear little importance to the setup of the model.

4.2 Exploring different sampling strategies (RQ1)

In online action detection, a critical aspect of processing involves the extraction of relevant features from video data. This is where the concept of a "sampling strategy" comes into play. Different feature extraction procedures can be viewed as different sampling strategies. Each strategy determines how to "sample" or select parts of the video data to extract features from, which will in turn form the basis for detecting actions. It is important to note however, that sampling strategy does not only refer to a specific feature extraction model, but also covers topics such as framerate or the separate estimation of optical flow. The choice of sampling strategy can significantly impact the performance of the action detection system, influencing factors such as computational efficiency or detection accuracy.

Before delving into the intricate details of our experimental results, it is essential to first understand the underpinning feature extraction models that were used. These models, which are pre-trained on different datasets and vary in numerous other characteristics, constitute a significant part of this analysis.

The **TSN backbone** was utilized, given its application in the OadTR and Colar publications. As inferred from the name **T**emporal **S**egment **N**etwork, this model extracts features based on segments or clips. The implementation utilized in these experiments anticipates a clip length of 6 frames. As previously mentioned, clip-level feature extractors were employed at a framerate of 30 FPS, to prevent a significant reduction in the number of valid training/testing examples. However, these clips are exclusively used in the optical flow arch of the model, which anticipates 5 consecutive optical flow representations. The RGB arch of the implementation only processes a single frame, specifically the central frame (e.g., the one at index position 3 in the clip), in accordance with extant literature [101]. This backbone was pre-trained on the Kinetics-400 dataset. The features procured by this sampling approach possess a dimensionality of $d_{\text{RGB}} = d_{\text{FLOW}} = 2048$. Given that the OadTR model utilizes 64 timesteps (here clips), this sampling approach encompasses $(64 * 6)/30 = 12.8$ seconds. The implementation of this backbone is furnished by mmaction2 [102].

The second sampling pipeline implemented harnesses **attention-based models**. Flow features were estimated using the GMFlowNet [103], which is pre-trained on FlyingThings3D [104]. RGB features are extracted utilizing the swin-transformer-v2 as implemented from PyTorch [105]. Model weights pre-trained on ImageNet-1K are also provided by PyTorch. The dimensionality of the features procured by this approach is $d_{\text{RGB}} = d_{\text{FLOW}} = 1024$. This approach extracts frame-level features. Consequently, given a framerate of 15FPS and the model expecting 64 timesteps, this approach covers $64/15 = 4.26$ seconds. Despite this being substantially less time than the TSN backbone covers, it is sufficient to capture the length of an average action in its entirety based on the METEOR dataset definition.

Thirdly, a **convolutional** counterpart to the attention-based sampling pipeline was employed. This approach exclusively utilizes feature extraction models provided by PyTorch [105]. Optical flow was extracted using the RAFT model, pre-trained on FlyingThings and FlyingChairs [106, 104]. RGB features were extracted using ResNet50, trained on ImageNet-1K. The features obtained from this approach have the following dimensionalities $d_{\text{RGB}} = d_{\text{FLOW}} = 2048$. The time period covered by this sampling approach is equivalent to the attention-based sampling approach.

Lastly, a sampling approach employing the feature extraction model, which yielded the most promising results in the Colar publication, was examined. The feature extraction model used is a **ResNet-50 i3D**, which was pre-trained on the Kinetics dataset [34]. Similar to the TSN feature extractor, this model expects clips of frames as input. The implementation used for the experiments runs on 8 consecutive frames. This sampling approach diverges from the other approaches tested, in that it does not return spatial and temporal features separately, but only one set of features encoding these information together. As with the other clip-wise sampling approach, it was applied to the videos at the original 30FPS, resulting in a coverage of $(64 * 8)/30 = 17.06$ seconds. This is the longest time span covered by any sampling approach employed. Similarly to the TSN model, the backbone implementation and pre-trained weights are provided by mmaction2 [102].

The experiments reported in this section were conducted on the OadTR model. All models were trained for 50 epochs. The only hyperparameter varying between experiments is *dim_feature*, which had to be adapted according to the dimensionality of the features used.

Experiment	**FPS**	**Feature_dim**	**ROC AUC**	**F1**	**Precision**	**Recall**	**mAP**
TSN Backbone	30	2 * 2048	0.614	0.634	0.641	0.638	0.30
Attention Backbone	15	2 * 1024	0.590	0.522	0.604	0.491	0.255
Convolutional Backbone	15	2 * 2048	0.615	0.602	0.620	0.589	0.274
ResNet50-i3D Backbone	30	2048	**0.668**	**0.671**	**0.686**	**0.659**	**0.313**

Table 4.1: Performance metrics represent the weighted class average of the impact of different feature extraction pipelines on a fixed model architecture's performance.

Table 4.1 provides a comprehensive comparison of the four distinct sampling strategies employed in online action detection, specifically focusing on their impact on the performance of a fixed model architecture. The strategies are evaluated based on six key performance metrics: ROC AUC, F1 score,

Precision, Recall, mean average precision (mAP), and calibrated average precision (cAP). Additionally, a confusion matrix per label and strategy was generated. These confusion matrices are visualised in figure C.1 in appendix C. Furthermore, appendix C provides label specific measures, including mAP and cAP, in table C.1.

The TSN Backbone, operating at 30 FPS and a feature dimensionality of 2 * 2048, demonstrates a balanced performance across all metrics, with ROC AUC, F1, Precision, and Recall scores all hovering around the 0.63 mark. The mAP and cAP measures are consistent with this trend, indicating a solid generalisation ability and effectiveness of this strategy. Interestingly, the cAP for the category *Cutting* is quite high despite its infrequent occurrence. This indicates that this backbone's ability to correctly detect cutting actions exceeds its ability to detect more common categories such as *Wrong Lane*. It is noted in the confusion matrices that for all behaviors (all labels except background), there are more false negatives than false positives. Also, it can be seen that the number of true positives deteriorates with a decreasing number of samples per class.

The Attention Backbone, operating at 15 FPS and a feature dimensionality of 2 * 1024, exhibits a slight decrease in performance across all metrics compared to the TSN Backbone. This is particularly noticeable in the F1 score, Recall, and also in the mAP, which is the lowest among all strategies. The confusion matrices for this strategy show that there are more evaluation samples available, which is unsurprising, given the fact that this strategy operates on a frame-level. However, despite a larger number of samples to train with, the model exhibits great uncertainty for the labels *Background* and *OverTaking*.

The Convolutional Backbone, despite operating at the same FPS as the Attention Backbone but with a higher feature dimensionality (2 * 2048), manages to slightly outperform the Attention Backbone in all metrics. This indicates that the increased feature dimensionality may contribute to improved performance, even at a lower FPS. Correspondingly, the mAP measure is notably higher, supporting this observation.

The ResNet50-i3D Backbone, operating at 30 FPS and a feature dimensionality of 2048, stands out as the top performer across all metrics, including mAP. It achieves the highest ROC AUC, F1, Precision, and Recall scores among all strategies, as well as impressive mAP values, thereby indicating its strong detection performance even under calibration. However, a closer look at the confusion matrices suggests that this may only hold for the most prominent labels (namely *Background* and *OverTaking*). Less frequent labels on the other hand are seemingly better captured by the TSN or Convolutional Backbone, as confirmed by table C.1.

In summary, while all strategies demonstrate varying degrees of effectiveness, the ResNet50-i3D Backbone emerges as the most promising sampling strategy for online action detection, given its superior performance across all evaluated weighted metrics. A closer look at less frequent classes, however, reveals that this seeming superiority appears to be mainly due to the good performance on more frequent labels.

Given the limited number of samples obtained, the calculation of correlation between characteristics and metric results, including mAP, was deemed inappropriate and thus refrained from. This decision aligns with the principle of ensuring robust and reliable statistical analysis.

4.3 Testing different model architectures (RQ2)

In the pursuit to answer the second research question, three different architectures were explored. As introduced in section 3, these models are the OadTR model, the Colar model and an integrated model using the OadTR skeleton, but enhancing it by the static branch of the Colar model.

For fair comparison, all three models were trained on features, extracted using the ResNet50-i3D pipeline. The training was conducted for 50 epochs, regardless of wall-clock time. It is noted, that training the Colar model is significantly faster than OadTR. This is due to a multitude of reasons, including the usage of temporal convolution instead of full self attention and the replacement of multi-headed self-attention with one-head attention on the current frame. The configuration of hyperparameter for both models can be found in appendix B.

In compliance with the naming convention introduced by OadTR, the model, integrating concepts from OadTR and Colar was named *VisionTransformer_v4* (the native OadTR implementation is named *VisionTransformer_v3*). Based on this, the experiment names in 4.2 are abbreviated as OadTR_v3 and OadTR_v4 respectively.

Experiment	ROC_AUC	F1	Precision	Recall
OadTR_v3	**0.668**	0.671	0.686	0.659
OadTR_v4	0.661	0.658	0.685	0.633
Colar	0.613	**0.682**	**0.688**	**0.688**

Table 4.2: Performance metrics represent the weighted class average of the impact of different models architectures.

The OadTR_v3 model demonstrated compelling performance across several metrics. Specifically, it achieved the highest ROC_AUC (0.668) and precision (0.686) amongst the three models as per Table 4.2. With a focus on the individual categories (Table C.2), it exhibited a strong performance in the 'Background' category, scoring the highest F1 measure of 0.799. For other categories, the performance was relatively subdued, suggesting that the model could benefit from further optimization.

The OadTR_v4 model, while displaying slight underperformance in terms of overall ROC_AUC and recall, recorded considerable precision of 0.685. In the 'Background' category, it achieved an F1 score of 0.777, suggesting satisfactory performance. The 'LaneChange' category saw the best performance from OadTR_v4 amongst all models, with an F1 score of 0.138. However, the F1 scores for 'WrongLane' and 'Cutting' categories were significantly low, indicating an area for potential improvements.

The Colar model showed strong performance in terms of balanced metrics, achieving the highest F1 and recall scores of 0.682 across all models. In category-specific performance, Colar displayed robustness in handling the 'Background' category with the highest precision of 0.832. Interestingly, it demonstrated a superior recall of 0.682 in the 'OverTaking' category, showcasing its sensitivity. Furthermore, Colar exhibited better F1 scores in 'WrongLane' and 'Cutting' categories than the other two models.

The 'Background' category, the most represented in the training data, showed strong performance in all models. In terms of F1 scores, the models ranged from 0.777 (OadTR_v4) to 0.799 (OadTR_v3). Additionally, Colar showcased superior precision, suggesting its effectiveness in identifying true positives within this category.

Performance in the 'OverTaking' category was slightly lower across all models, but Colar's sensitivity, as indicated by a recall of 0.682, outperformed the others. This suggests that Colar, despite lower precision, was more adept at detecting 'OverTaking' instances, limiting the occurrences of false negatives.

The 'LaneChange' category witnessed sub-optimal performance from all models, reflecting a challenge in accurately predicting this action. However, OadTR_v4 achieved a higher F1 score of 0.138 in comparison to the other models, indicating a more balanced performance in terms of precision and recall.

For the 'WrongLane' category, all models performed poorly, indicating a difficulty in accurately predicting this label. Notably, the Colar model provided slightly better F1 scores, showing potential for improvement with further tuning and optimization.

Finally, the 'Cutting' category, which had the least representation in the training data, saw all models struggling to perform. The Colar model, despite the challenges, showed the highest F1 score of 0.041, indicating some resilience in handling this category.

In summary, all models demonstrated varying performance across different categories, each excelling in certain areas while requiring improvements in others. The Colar model showed balanced performance, OadTR_v3 outperformed in precision and ROC_AUC, and OadTR_v4 exhibited strength in the 'LaneChange' category. Despite the strong performance in the 'Background' category, all models demonstrated a need for further optimization for the less represented categories such as 'Cutting'.

A more visual representation of the model performance can be found in figure C.2.

4.4 Generating salient cues as explanations (RQ3)

The methodology employed to identify salient cues for the elucidation of the model's predictions is detailed in Section 3.6. In these experiments, a masking technique was employed which involved setting parts of the input video to zero, thus effectively 'removing' certain agents from the frame.

This masking approach creates a suite of $n + 1$ videos, where n represents the total number of

agents in the frame. The prediction results from these masked videos are then compared against a baseline prediction from the unmasked video. The comparison provides a measure of influence of each agent on the classification outcome.

In addition to the primary method, an alternative technique utilizing negative masks was also investigated. This technique deviates from the positive masking approach, in that it preserves only the actor of interest within each frame, with all other regions being set to zero.

However, the use of negative masks led to undesirable outcomes, in that the classification results appeared to be random and yielded no noticeable improvements. In fact, the lack of contextual data within the frames due to negative masking, led to significant declines in classification accuracy.

Hence, following initial exploratory trials, the strategy of using negative masks was abandoned due to its detrimental effects on classification performance. However, it is noteworthy that the current codebase is designed to allow for swift adaptation and potential future exploration of negative masks.

The reported values are generated by applying a sigmoid function to the last layer of the neural network. Given the nature of the multilabel classification problem at hand, the outputs of this operation can be interpreted as probabilities. Each output corresponds to the probability of the associated class label being present, as estimated by the model. This is because each output logit represents a separate binary classification problem, independent of the others. It's worth noting that these 'probabilities' do not sum to 1 across classes, reflecting the fact that multiple class labels can be simultaneously present in this multilabel context.

Generally speaking it would be difficult to incorporate video explanations into this report. In order to provide some visualization and allow for a mapping of probability differences and agents, it was decided to use the last frame as a reference. Therefore each explanation consists of a visual representation of the last image next to a legend, which maps the colored bounding boxes to probability estimates and the difference between these new differences and the original ones.

4.5 Ablation Studies

To elucidate the intricate relationships between various hyperparameters and the model's performance, we performed a series of ablation studies. These in-depth explorations allow for the quantification of the effects of individual parameters, providing crucial insights into their role in the model's classification performance.

The scope of these ablation studies includes hyperparameters such as the number of encoder and decoder layers, the dropout probability, and the size of the hidden dimensions. Each model was trained consistently for 20 epochs, utilizing Temporal Segment Networks (TSN) features, coupled with the OadTR_v3 model architecture. This standardized experimental design ensured the effects observed could be directly linked to the manipulated hyperparameters.

The comprehensive results of these ablation studies are provided in Appendix C. This section offers a condensed summary of these findings, highlighting key trends and pivotal insights. Therefore, it paves the way for future model refinement and optimization efforts.

As observed in Table C.3, varying the number of decoder layers results in minor performance fluctuations. The highest performance across all metrics is achieved with four decoder layers, indicating an optimal complexity-performance trade-off at this point. Thus, adding more decoder layers may not yield significant performance gains. Yet, the minute performance differences suggest that the number of decoder layers isn't the most critical performance determinant.

Turning to the number of encoder layers, a more complex interplay is observed (Table C.4). Although the ROC_AUC and Precision peak at five layers, the Recall metric suffers at this configuration. A balanced performance across all metrics occurs with four encoder layers, suggesting a saturation point in model complexity. Beyond this point, added complexity seems to negatively affect some performance aspects, particularly the Recall metric. However, the performance impact of the number of encoder layers, similar to decoder layers, remains marginal, reinforcing the importance of other determining factors.

The dropout probability exhibits a certain level of influence on the model's performance (Table C.5). A dropout probability of 0.5 results in the highest F1 score, Precision, Recall, and second highest ROC_AUC. Interestingly, a dropout probability of 0.1 also yields a similar ROC_AUC score, indicating that the relationship between dropout probability and performance is not strictly linear. Although the Recall metric generally improves with higher dropout rates, the modest metric variations suggest that

dropout probability is not the dominant performance influencer. Therefore, the optimal dropout rate is likely dataset and problem-specific and may vary with other hyperparameters.

Finally, increasing the hidden dimension size generally improves the model's performance (Table C.6). While the ROC_AUC, F1 score, and Precision metrics peak at a hidden dimension size of 1024, the Recall maximizes at a dimension size of 512. A marginal Recall decrease at 1024 suggests a minor trade-off when model complexity surpasses a certain threshold. Despite this, the difference between performance at varying hidden dimension sizes is not drastic, indicating that it is a factor, albeit not the only one, influencing model performance. This highlights the necessity of considering a combination of model configurations and training strategies for optimal performance.

Chapter 5

Discussion

The discussion section seeks to elaborate on the findings presented earlier, provide possible explanations, and propose future research avenues. To aid further discussion, the first part focuses on the generation of salient cues, which is important as these are used in subsequent parts of this discussion. The second part focuses on the implications of four distinct sampling strategies and their impact on online action detection system performance. These strategies were implemented using various pre-trained feature extraction models: the Temporal Segment Network (TSN) Backbone, Attention Backbone, Convolutional Backbone, and the ResNet50-i3D Backbone. Thirdly, the performance of three architecturally different online action detection models will be evaluated. Followed by a brief discusion about the ablation studies conducted.

5.1 Meta-discussion on Salient Cue Generation (RQ3)

While no quantitative observations have been made, several factors appear to qualitatively influence the efficacy of the generated explanations.

Actor Size By definition, larger, more prominent agents have a greater impact on the extracted image features and thus are more likely to cause substantial variation in classification results. This effect is exemplified in Figure D.3, where the explanation for *Cutting* is dominated by the orange agent, which appears to be more prominently positioned than the red or blue agent.

Number of Agents The effectiveness of the explanations seems to be inversely correlated with the number of agents. As the scene becomes more crowded, the explanation is less likely to identify the correct agent with a high level of certainty.

Contrasting Aggressors and Victims A key challenge of this approach is the lack of a reliable way to differentiate between an aggressor and a victim of aggression. In terms of the generated explanations, there is minimal conceptual difference if the aggressor or the victim of aggression is omitted from the scene. Specifically, an overtaking maneuver can only be labeled as aggressive if there is an agent being overtaken aggressively. Removing one of the two from the frame should yield similar results.

Background Light The method of masking agents in black was hypothesized to be less effective in night scenes. However, the examples provided in this thesis and the accompanying repository do not support this assumption. This could be attributed to the model's expectation of black (or dark) pixels primarily appearing in the upper part of videos, while the masks usually occlude the lower part, even during nighttime conditions.

5.2 Discussing Results for different sampling strategies (RQ1)

Dimensionality of Feature Representation The conducted experiments display a correlation between the dimensionality of feature representation and model performance. Specifically, the Con-

volutional Backbone, despite its 15 FPS rate identical to the Attention Backbone, outperforms the latter in every metric, including on less frequent labels such as Cutting. This superiority might be attributed to its higher feature dimensionality (2 * 2048), potentially providing a more comprehensive representation of the video data and subsequently leading to an improved performance. Interestingly, despite its lower feature dimensionality, the ResNet50-i3D Backbone achieves the highest performance across all categories. This apparent contradiction requires further exploration.

Higher vs. Lower Frame Rate The observed performance disparity between models operating at 30 FPS (TSN Backbone and ResNet50-i3D Backbone) and those at 15 FPS (Attention Backbone and Convolutional Backbone) was noteworthy. Lower framerate models, underperform their higher framerate counterparts across all metrics and for all labels, including recall and precision. These results emphasize the role of higher frame rates in capturing nuanced details in video data, which are essential to the task of detecting aggressive driving behavior.

Explicit Usage of Optical Flow The explicit implementation in the original publication of the OadTR model as well as reported early experiments in the Colar publication, highlights its importance in online action detection. Optical flow offers temporal features essential for understanding motion and temporal changes. This appears to be particularly useful in detecting less frequent behaviors like Cutting, where the ResNet-i3D backbone underperforms, which is evident in table C.2. On the other hand the overall performance of the ResNet-i3D backbone was proven superior to the other approaches. This highlights, that there is no need to model optical flow explicitly. A finding, aligning with the way the OadTR model process the features, which is by concatenating them along the first axis. Hence there is no separate arch of the model specifically dedicated to computations involving optical flow.

Time Span Covered The deployed sampling strategies, covering varying time spans, have distinctive impacts on their performance. Notably, the ResNet50-i3D Backbone, which spans the longest time duration, attains the best performance across all weighted metrics. This outcome suggests that longer context capture could enhance the model's capacity to efficiently identify and categorize aggressive driving behaviors. However, the performance of the Convolutional Backbone on *Cutting* implies the relationship between the covered time span and detection performance may be influenced by the duration of the action being detected. This inference is drawn from the observation that *Cutting* is shorter than the other assessed behaviors, as outlined in [91].

Multipurpose Backbone vs. Human Actions Backbone Initial intuition suggested that a multipurpose backbone might outperform a backbone pre-trained on human actions, especially when there are no human actions in the frame. The experimental results partially confirm this presumption. The TSN Backbone, pre-trained on human actions, was surpassed by the multipurpose ResNet50-i3D Backbone in all measures across all labels, implying that a multipurpose backbone, trained on diverse data, might possess a more versatile understanding. It could be particularly beneficial when the actions to be detected substantially differ from human actions. However, the Convolutional and Attention Backbones, which also utilize multipurpose feature extraction models, could not outperform the ResNet50-i3D Backbone. This outcome implies that other elements such as model architecture, feature dimensionality, and frame rate might also significantly influence performance.

Comparison to OadTR As can be seen table C.1, the results obtained by any of the experiments are in no way competitive to the ones obtained in the OadTR publication by the original model on the THUMOS14 dataset [30]. To some extend these differences are expected, given that different data was used that is potentially not as well suited for the task of online action detection. However, not using a fine-tuned feature extraction model definitely added to this discrepancy. While this was expected before conducting experiments, it was still infeasible to locally fine tune on the METEOR dataset.

5.3 Discussion on Results for Different Model Architectures (RQ2)

The results from the experiment examining various model architectures reveal distinct strengths and weaknesses for each model, highlighting potential areas for improvement.

OadTR_v3 The OadTR_v3 model demonstrated notable performance in terms of ROC_AUC and precision, leading the three models with respective values of 0.668 and 0.686. This model exhibited a particular strength in discerning the 'Background' category, attaining the highest F1 score of 0.799. However, its somewhat subdued performance in other categories suggests opportunities for further refinement. The model's suboptimal results in the *LaneChange*, *WrongLane*, and *Cutting* categories underline the need to enhance its proficiency for less represented categories in the dataset.

Colar The Colar model demonstrated robust overall performance, registering the highest F1 and recall scores of all models at 0.682. Its strong proficiency in handling the 'Background' category, denoted by the highest precision score of 0.832, indicates a particular strength in identifying true positives. Additionally, the model showed superior sensitivity in the *OverTaking* category with a recall score of 0.682, surpassing the other two models' performance. The Colar model notably outperformed other models in the *WrongLane* and *Cutting* categories. However, its relatively lower ROC_AUC and precision compared to the OadTR_v3 model suggest potential areas for further refinement. Its comparatively superior performance on underrepresented categories suggests that categories with fewer training/testing examples especially benefit from comparison with representative exemplars.

OadTR_v4 The OadTR_v4 model, a synthesis of OadTR and Colar concepts, exhibits remarkable precision, scoring 0.802 for the 'Background' category. The model however underperforms in overall ROC_AUC and recall. Interestingly, OadTR_v4 outshines all other models in the less frequently encountered 'WrongLane' category, with an F1 score of 0.076 - almost twice the score of its predecessor, OadTR_v3. This suggests that the integration of Colar and OadTR methodologies is particularly effective for rare labels. On the other side, the model failed to classify any instances of 'Cutting', as indicated by zeroes across all metrics for this category. This underscores the need for further refinements in the merging of OadTR and Colar concepts. Despite these shortcomings, the unique strengths displayed by OadTR_v4 imply a potential for the integrated model, particularly when adjustments are made for better assimilation of the constituent models' features.

The comparable performance across the models, particularly in less represented categories, may be more reflective of the characteristics of the dataset rather than the inherent capabilities of the models. This observation suggests that enhancements in the dataset, such as increasing the representation of underrepresented categories, could potentially augment the performance of these models.

5.3.1 Utilizing Visual Explanations

To facilitate understanding of the inner workings of individual model configurations, some generated explanation examples are presented in D. A major challenge in comparing these explanations is the dependency of the ground truth (*orig pred* in the legend) on the model generating the explanations. To gain a better understanding, one example per action category will be discussed, and additional explanations can be found in the accompanying repository.

In the first example depicted in Figure D.1, all models incorrectly interpret the action *OverTaking* as *WrongLane* or *LaneChange*. This confusion might arise from the fact that *OverTaking* often involves elements of both these actions. However, with regards to *OverTaking* explanations, it is noteworthy that the v4 model provides the most persuasive explanations. The discrepancy from the original prediction is greatest when excluding the green agent, which carries the positive label. Despite the confusion regarding the nature of aggression depicted, explanations for the v3 and v4 models correctly attribute the act of aggression to the green agent. While this information is not directly relevant to prediction accuracy, it is significant for understanding which agent might pose a threat to the *EgoVehicle*'s safety.

The second example, presented in Figure D.2, differs from Figure D.1 in two primary aspects. Firstly, the scene is more crowded, containing two additional agents compared to D.1. This leads

to the second, more complex difference: partial occlusion among these agents. For instance, in the last frame, the green agent, partially occluded by the purple agent, in turn obscures the red and blue agents. This implies a significant challenge for the masking approach, which inherently relies on occlusion. All models exhibit high values for the *LaneChanging* labels, with Colar also recording a high value for *OverTaking*. However, all explanations show the steepest drop when masking the green agent, indicating that this agent could be performing the aggressive action. The aggressive behavior of the purple agent is not adequately represented in the explanations, highlighting a limitation of the masking method.

Lastly, Figure D.3 presents arguably the most challenging example discussed here. Not only does it involve 11 agents—a significantly higher number than previous examples—but it also contains two distinct actions. The red agent, obscured in the final frame, is *Cutting* another agent, while the bright green agent is *OverTaking*. The models *Colar* and *OadTR_v3* correctly assign the highest class probabilities to these two behaviors, whereas *OadTR_v4* struggles to correctly identify *Cutting* in the scene.

Considering *OverTaking*, all models misattribute the responsibility. Even though the light green agent is overtaking, the largest difference is observed when the overtaken agent is occluded. This is consistent across all models, albeit the exact agent varies. Remarkably, some degree of occlusion is observed. For example, in the final frame, the turquoise agent is temporarily almost entirely occluded by the orange agent.

The explanations for *Cutting* fail to correctly attribute the action to the agent either performing the cut or being cut. All three models report the largest difference when masking the orange agent. While it is the red agent that performs the *Cutting*, it seems implausible that the orange agent is being cut off due to their noticeable distance in the final frame. While the generated explanations fail to fully capture the actual complexity of the scene, they nonetheless provide insights into the relative importance of a specific region.

As is emergent from reviewing these visual explanations, the models perform fairly similar compared to each other. The explanations for the different models are fairly similar to each. Hence, they face similar problems like a degradation in goodness of explanation as the number of agents increases, a difficulty to correctly identify the aggressor and a dependence on the initial classification scores. Regarding these, the findings from the visual explanations support the metric scores outlined in tables 4.2 and C.2.

5.4 Discussing ablation studies

The ablation studies presented in the previous section allow for the exploration and quantification of the contributions of individual hyperparameters to the model's performance, thereby informing future adjustments and fine-tuning.

Decoder Layers Varying the number of decoder layers (Table C.3) had a minor impact on performance. The model performed optimally with four layers, demonstrating a delicate balance between model complexity and performance. However, the narrow performance variations suggest that the number of decoder layers is not a significant determinant of the overall performance. Thus, while a model with four decoder layers is optimal in our scenario, it is also evident that the performance improvement gained by optimizing this hyperparameter is relatively limited.

Encoder Layers Adjusting the number of encoder layers (Table C.4) yields a slightly more complex relationship. The highest ROC_AUC and Precision were observed with five layers. However, the model's Recall decreased at this configuration, suggesting a potential trade-off between different performance aspects. It seems that while additional complexity enhances the model's precision, it may compromise its ability to capture all relevant patterns, as indicated by the lower Recall. Similar to the decoder layers, while four encoder layers seem to offer a balance, the actual performance gain or trade-off is minimal, indicating other factors may be more influential.

Dropout Probability As observed in Table C.5, the impact of dropout probability on the model's performance is noticeable, albeit not drastic. The F1 score and recall peaked at a dropout probability

of 0.5, indicating that a more generalized model, achieved by a higher dropout rate, was better at balancing precision and recall and capturing all the true positive instances.

However, the highest ROC_AUC and Precision were observed at a dropout probability of 0.1, suggesting that a less regularized model (lower dropout rate) was able to better discriminate between classes and more accurately predict positive instances.

The observed relationship between dropout rate and performance metrics was not linear, suggesting the optimal dropout rate may indeed be task-specific and could vary depending on the specificities of the dataset, the model architecture, and the presence of other hyperparameters. Moreover, the relatively small variations across different dropout rates imply that while dropout is a significant consideration in model tuning, it is not the sole dominant factor impacting the model's performance.

Hidden Dimension Size Increasing the size of the hidden dimensions (Table C.6) generally improves the model's performance. The ROC_AUC, F1, and Precision peaked at a hidden dimension size of 1024, while the Recall was maximized at a dimension size of 512. The minor drop in Recall at a dimension size of 1024 suggests a potential trade-off beyond a certain threshold of model complexity. Although the size of the hidden dimensions appeared to influence model performance to some extent, the variations across different sizes were not drastic, emphasizing that model performance hinges on a combination of factors and configurations.

Taken together, these ablation studies underline the importance of understanding the nuanced interplay of hyperparameters in complex models. While individual parameters can influence performance, no single parameter was found to have a dramatic effect in isolation. This highlights the necessity of treating model refinement as a holistic process, where multiple factors and their interactions should be considered. Furthermore, the results underscore the importance of task-specific tuning, as the optimal configuration can vary based on the nature of the task and dataset.

Chapter 6

Conclusion

In the realm of road safety, aggressive driving behavior has long been a subject of concern and scrutiny. Through an intricate blend of attention-based models, feature extraction techniques, and a rich dataset, this research has paved the way for a more comprehensive understanding of aggressive driving behavior.

The METEOR driving dataset, which served as the backbone of this research, was meticulously chosen for its extensive annotations and representation of rare and aggressive behaviors. The dataset's richness was further enhanced through label refinement, where behaviors were mapped based on established definitions of aggression. This refinement was crucial in ensuring that the dataset was tailored to the specific needs of this research.

Feature extraction was a cornerstone in this study. The decision to employ pre-trained backbones was not taken lightly. The research acknowledged the potential benefits of custom pre-training but ultimately opted for pre-trained backbones due to practical challenges and the desire for compatibility with existing literature. This decision underscores the importance of adaptability and pragmatism in research.

In evaluating different feature extraction pipelines, the study delved into the nuances of frame-level and clip-level features. Frame-level features, which generate a feature vector for each input frame, were juxtaposed with clip-level features that yield a single feature vector for a sequence of frames. This distinction is critical as clip-level features encode more temporal context, while frame-level features capture spatial information.

The experiments conducted were a testament to the research's commitment to rigor and thoroughness. Through these experiments, the research illuminated the relationship between the dimensionality of feature representation and model performance. It was observed that higher dimensionality often translated into more comprehensive representations of video data, which in turn contributed to improved performance.

Another revelation from the experiments was the pivotal role of frame rates. Models operating at higher frame rates were adept at capturing the subtle details that are often the difference between accurately detecting aggressive driving behavior and missing it altogether. This finding is particularly significant as it highlights the importance of temporal resolution in video analysis.

In terms of novelty, this research presented two main contributions. Firstly, the innovative step of integrating the OadTR model with the Colar model resulted in a novel hybrid that can exploit historical exemplars while predicting future frames, incorporating all this information into a comprehensive prediction. While global metrics did not reveal superiority over the individual models, there were distinct improvements, particularly for lesser-represented labels. This suggests, that the combined approach holds potential, a finding that invites further exploration.

Secondly, the generation of salient cues surfaced as an innovative aspect of this research. While the study did not make quantitative observations regarding salient cues, it did offer qualitative insights. Multiple factors were identified as influencers of the efficacy of the generated explanations. This aspect of the research is particularly intriguing as it delves into the interpretability of the models. Contrasting to main research directions in this field, the approach used in this study is free of optimization. Thus, generating reproducible, transparent and fast explanations while only requiring minimal computational resources. These characteristics make them especially interesting for deployment in areas with restricted access to computing power and the need for real-time interpretability, as is usually the case for autonomous vehicles.

A significant challenge encountered during the course of this research related to the utilization of the provided infrastructure. The expertise typically associated with the role of a data scientist primarily involves working with models, theories, and abstract concepts. Nevertheless, this research necessitated a substantial extension into the realm of systems management. Tasks such as setting up and maintaining a CUDA environment, along with resolving version conflicts among software dependencies, were integral to this process. Despite these tasks being somewhat tangential to the core objectives of the research, they were indispensable for ensuring the operability and efficiency of the tested models. This technical hurdle, although demanding and time-consuming, provided valuable insights. It highlighted the crucial role of robust and reliable computational infrastructure when undertaking data-intensive research, an aspect that extends beyond the mere application of data science techniques.

Through its innovative use of attention-based models, meticulous data refinement, and rigorous experiments, it has shed light on the multifaceted nature of aggressive driving behavior and the challenges involved in its assessment. The research has also opened the door to new possibilities and questions, which will undoubtedly be the focus of future studies.

One of the key takeaways from this research is the importance of data quality and representation. The METEOR dataset, with its rich annotations, proved to be invaluable. Future research should continue to seek out or develop datasets that are representative of real-world driving behaviors, especially those that are underrepresented in existing datasets.

Additionally, the role of technology in road safety cannot be understated. Attention-based models represent just one of the many tools available for assessing driving behavior. As technology continues to evolve, so too will the tools and methods available for road safety research. Future research should remain adaptable and open to incorporating new technologies as they emerge.

Moreover, collaboration between academia, industry, and regulatory bodies is essential. The insights gained through research such as this can inform policy and contribute to the development of safer vehicles and road systems. Engaging with stakeholders across different sectors can ensure that the research has a tangible impact on road safety.

In closing, this research has made a significant contribution to the understanding of aggressive driving behavior through the lens of attention-based models. It has navigated the complexities of feature extraction, model interpretability, and data representation with rigor and innovation. The findings and insights gleaned from this study serve not only as a testament to the capabilities of attention-based models but also as a call to action for continued exploration and collaboration in the pursuit of safer roads.

Chapter 7

Future Work

As this research concludes, it is imperative to recognize that the journey of understanding and assessing aggressive driving behavior is far from over. The insights and findings of this study pave the way for future research endeavors that can further enhance the efficacy and applicability of attention-based models in assessing aggressive driving behavior. This section outlines potential directions for future work, with an emphasis on the incorporation of a customized backbone.

Customized Backbone One of the critical aspects that future research should focus on is the incorporation of a customized backbone for feature extraction. In this study, pre-trained backbones were employed due to practical challenges and the desire for compatibility with existing literature. However, the use of pre-trained backbones may not be optimally tailored to the specific characteristics and nuances of aggressive driving behavior.

Developing a customized backbone can allow for more specialized feature extraction that is attuned to the intricacies of driving behavior. This can potentially lead to more accurate and robust models. Customized backbones can be trained specifically on driving datasets, allowing them to capture features that are particularly relevant to driving scenarios.

Enhanced Model Interpretability While this study made strides in generating salient cues for explanations, there is room for improvement in model interpretability. Future research should focus on developing models that not only accurately assess aggressive driving behavior but also provide interpretable and understandable explanations for their assessments. This is crucial for practical applications, especially in legal and regulatory contexts where explanations may be required.

Diverse and Representative Datasets The METEOR dataset played a significant role in this research. However, future research should seek to incorporate more diverse and representative datasets. This includes datasets that capture driving behavior in different geographic locations, under various weather conditions, and across a range of driving cultures. Such datasets can help in developing models that are more generalizable and applicable in a broader range of scenarios.

Real-time Applications Developing models that can operate effectively in real-time is another avenue for future research. This includes optimizing models for computational efficiency without compromising accuracy. Real-time assessment of aggressive driving behavior has practical applications in traffic management, law enforcement, and driver assistance systems.

Ethical and Privacy Considerations As models become more advanced and capable, ethical and privacy considerations become increasingly important. Future research should consider the ethical implications of assessing driving behavior, particularly in terms of data privacy, consent, and the potential for bias in model assessments.

Collaboration and Stakeholder Engagement Engaging with stakeholders, including policymakers, automotive manufacturers, and road safety advocates, is essential for ensuring that the research

has practical and tangible impacts. Collaboration can facilitate the translation of research findings into policies, technologies, and interventions that contribute to road safety.

Appendix A

Additional dataset statistics

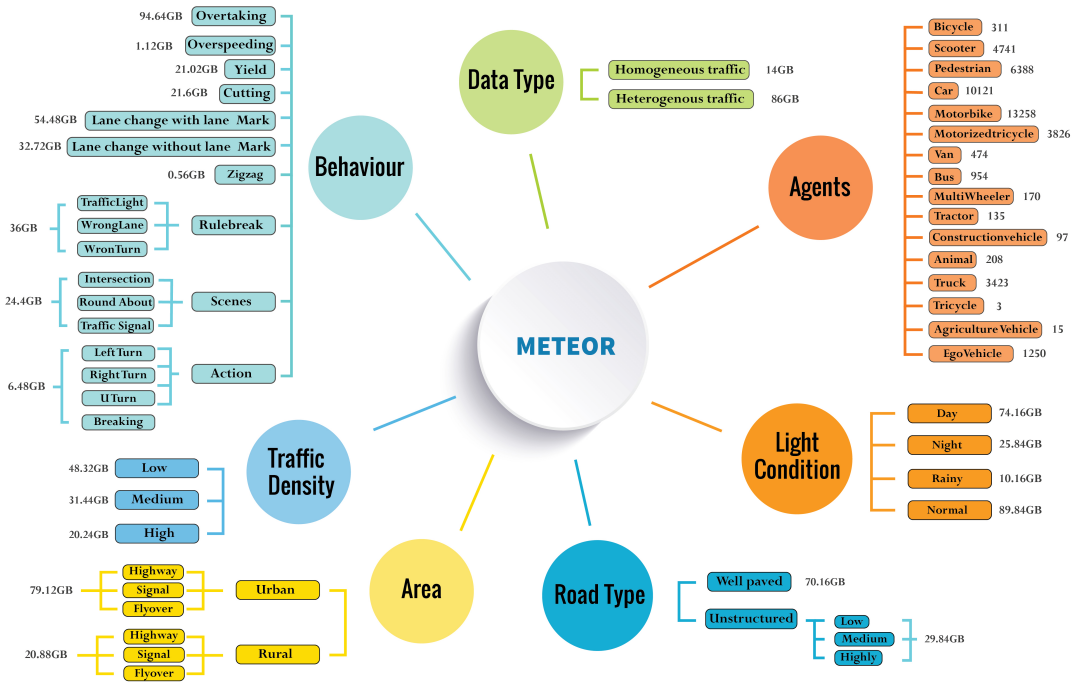


Figure A.1: Overview of Labels in METEOR Dataset [91]

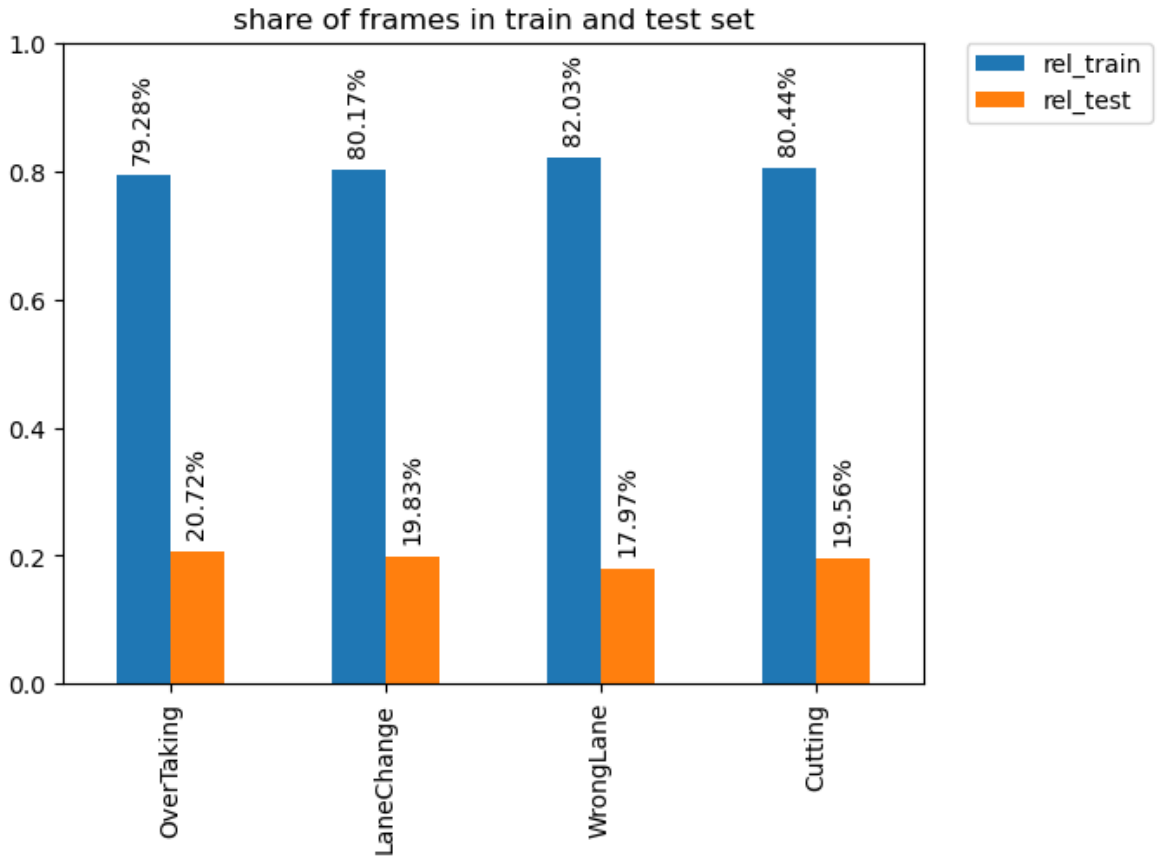


Figure A.2: Share of frames used for training and testing

	OverTaking*	OverSpeeding	LaneChanging(m)*	LaneChanging*	TrafficLight	WrongLane*	WrongTurn	Cutting*
EgoVehicle	227225	9	165866	37371	32	23733	187	4595
MotorizedTricycle	66556	209	9974	3800	7	4985	2	1233
Car*	304322	87	88686	28558	208	8515	24	5534
Van	6501	0	929	820	0	268	0	0
MotorBike*	319702	285	14271	6917	727	40863	115	4994
Pedestrian	45	0	0	0	0	377	4	5180
Bicycle	852	0	18	0	0	2762	1	142
MultiWheeler	0	0	0	0	0	332	0	0
Truck	63728	89	18889	4376	0	4532	0	3868
Scooter*	130465	7	7237	2897	220	15440	281	2289
Tractor	57	0	0	0	0	2598	0	0
Bus	33756	10	9263	3296	0	4852	0	817
ConstructionVehicle	802	0	245	0	0	0	0	21
Animal	0	0	0	0	0	0	0	390
AgricultureVehicle	0	0	0	0	0	0	0	0
Pedestrain	0	0	0	0	0	0	0	0
Tricycle	0	0	0	0	0	0	0	0

Table A.1: Number of Frames where a certain actor is annotated as showing a specific aggressive behaviour. Categories and Actors that are part of the final configuration are marked with a *

Appendix B

Model Descriptions

Argument	Default Value
exp_name	'ColarMETEOR'
data_root	'/workspace/pvc-meteor/features/features_i3d.pkl'
dataset_file	'/workspace/pvc-meteor/features/METEOR_info.json'
kmean	'/workspace/pvc-meteor/features/colar/exemplar.pickle'
checkpoint	'./checkpoint/THUMOS-TSN-Kinetics.pth'
seed	20
lr	3e-4
weight_decay	1e-4
cuda_id	0
lr_drop	1
input_size	2048
enc_layers	64
numclass	5
batch_size	512
overlap	1
num_workers	8
start_epoch	1
epochs	20
output_dir	'checkpoint'
clip_max_norm	1.0
feature_type	'METEOR'
command	'kinetics'

Table B.1: List of Colar hyperparameters and their default values

Argument	Default Value
lr	1e-4
batch_size	512
weight_decay	1e-4
epochs	60
resize_feature	False
lr_drop	1
lr_drop_size	0.5
clip_max_norm	1.0
dataparallel	None
removelog	None
use_flow	True
version	'v3'
query_num	8
decoder_layers	4
decoder_embedding_dim	1024
decoder_embedding_dim_out	1024
decoder_attn_dropout_rate	0.4
decoder_num_heads	4
classification_pred_loss_coef	0.5
enc_layers	64
lr_backbone	1e-4
feature	'3D_Resnet'
dim_feature	2048
patch_dim	1
embedding_dim	1024
num_heads	8
num_layers	3
attn_dropout_rate	0.4
positional_encoding_type	'learned'
hidden_dim	512
dropout_rate	0.4
numclass	22
classification_x_loss_coef	0.3
classification_h_loss_coef	1
similar_loss_coef	0.1
margin	1.0
weighted_loss	'True'
weight_values	0
dataset_file	'/workspace/pvc-meteor/features/METEOR_info.json'
frozen_weights	None
thumos_data_path	'/home/dancer/mycode/Temporal.Online.Detection/ Online.TRN.Pytorch/preprocess/'
thumos_anno_path	'data/thumos_{}_anno.pickle'
remove_difficult	None
device	'cuda'
binary_label	False
output_dir	'models'
seed	20
resume	"
start_epoch	1
eval	None
num_workers	8
use_frequent	'True'
use_infrequent	'False'
pickle_file_name	'METEOR.pickle'
world_size	1
dist_url	'tcp://127.0.0.1:12342'

Table B.2: List of OadTR hyperparameters and their default values

Appendix C

Additional experiment results

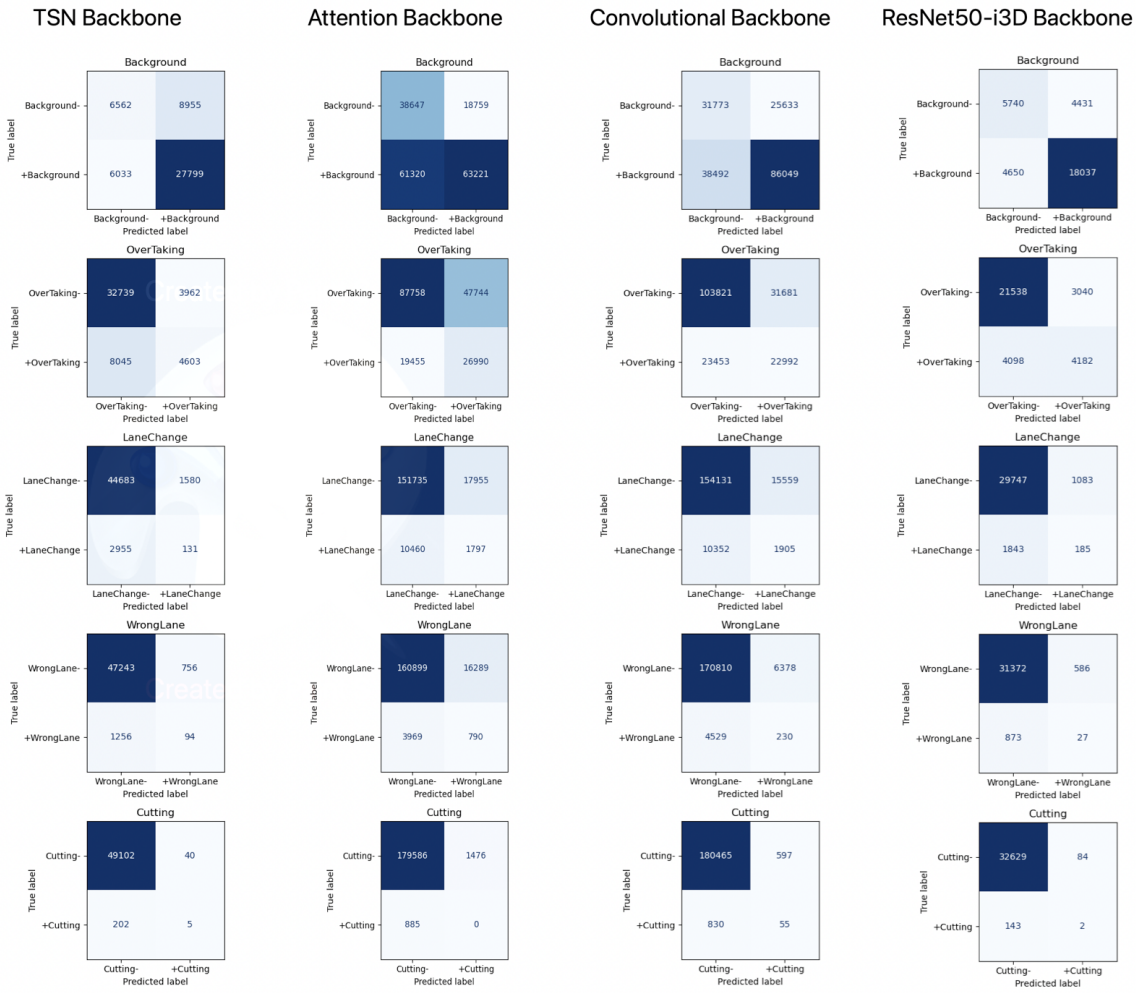


Figure C.1: Multilabel Confusion Matrices for Different Sampling Strategies

	category	F1	recall	precision	mAP
TSN Backbone	Background	0.788	0.822	0.756	0.835
	OverTaking	0.434	0.364	0.537	0.496
	LaneChange	0.055	0.042	0.077	0.09
	WrongLane	0.085	0.070	0.111	0.062
	Cutting	0.040	0.024	0.111	0.019
Attention Backbone	Background	0.612	0.508	0.771	0.768
	OverTaking	0.445	0.581	0.361	0.389
	LaneChange	0.112	0.147	0.091	0.079
	WrongLane	0.072	0.166	0.046	0.033
	Cutting	0.000	0.000	0.000	0.005
Convoluational Backbone	Background	0.729	0.691	0.770	0.801
	OverTaking	0.455	0.495	0.421	0.421
	LaneChange	0.128	0.155	0.109	0.096
	WrongLane	0.040	0.048	0.035	0.032
	Cutting	0.072	0.062	0.084	0.015
ResNet50-i3D Backbone	Background	0.799	0.795	0.803	0.858
	OverTaking	0.540	0.505	0.579	0.542
	LaneChange	0.112	0.091	0.146	0.107
	WrongLane	0.036	0.030	0.044	0.046
	Cutting	0.017	0.014	0.023	0.012

Table C.1: Measures for different sampling strategies, specific for each label

	category	F1	recall	precision	mAP
OadTR_v3	Background	0.799	0.795	0.803	0.858
	OverTaking	0.540	0.505	0.579	0.542
	LaneChange	0.112	0.091	0.146	0.107
	WrongLane	0.036	0.030	0.044	0.046
	Cutting	0.017	0.014	0.023	0.012
OadTR_v4	Background	0.777	0.754	0.802	0.849
	OverTaking	0.532	0.493	0.578	0.546
	LaneChange	0.138	0.142	0.133	0.1
	WrongLane	0.076	0.091	0.065	0.041
	Cutting	0.000	0.000	0.000	0.077
Colar	Background	0.790	0.753	0.832	0.875
	OverTaking	0.581	0.682	0.507	0.574
	LaneChange	0.200	0.314	0.147	0.125
	WrongLane	0.054	0.061	0.048	0.045
	Cutting	0.041	0.044	0.039	0.029

Table C.2: Measures for different model architectures, specific for each label

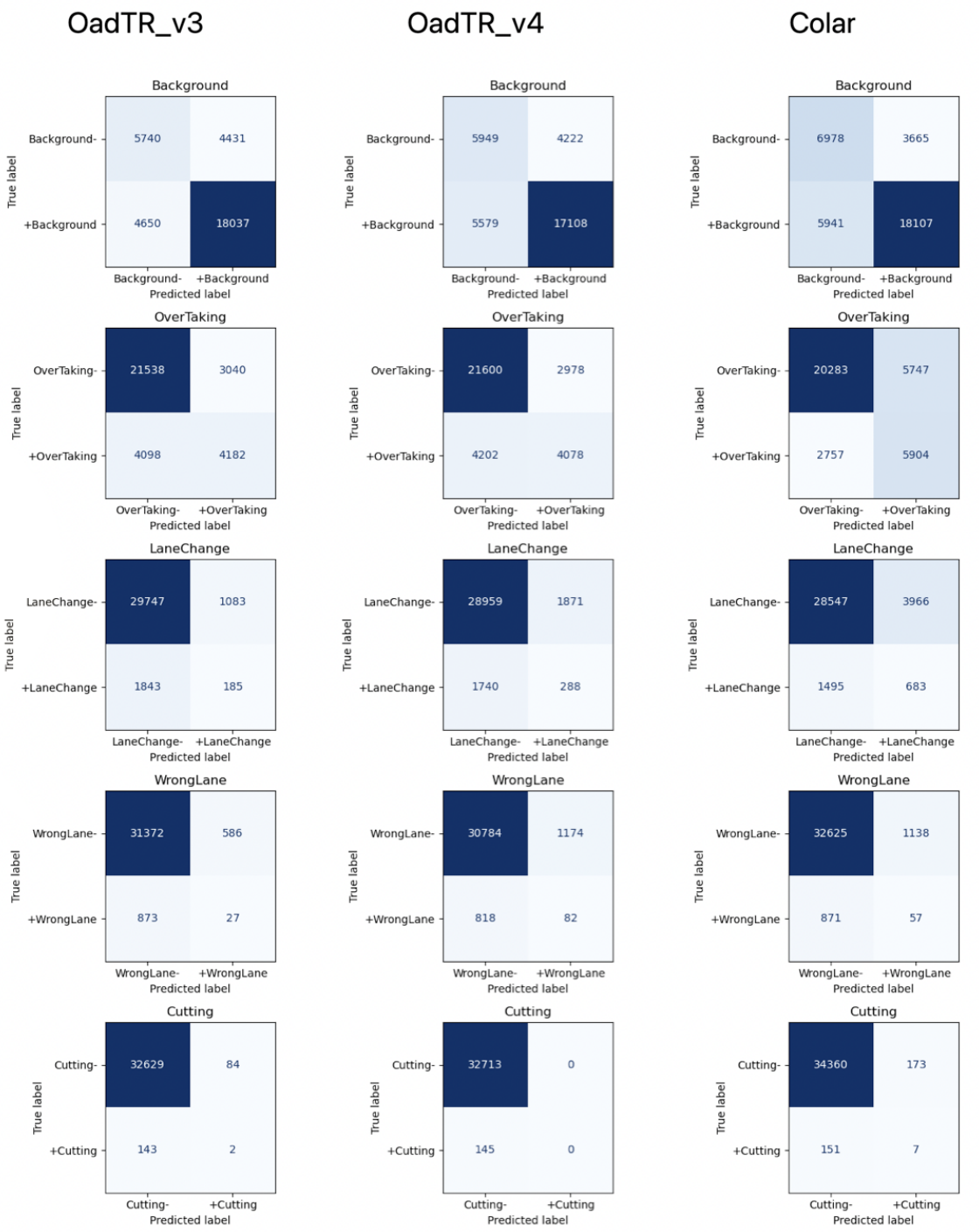


Figure C.2: Multilabel Confusion Matrices for Different Model Architectures

Metric	3	4	5	6
ROC_AUC	0.690	**0.697**	0.686	0.691
F1	0.676	**0.679**	0.644	0.666
Precision	0.676	**0.678**	0.673	0.675
Recall	0.699	**0.707**	0.664	0.691
mAP	0.303	**0.307**	0.296	0.301

Table C.3: Classification performance of models with different numbers of decoder layers.

Metric	2	3	4	5
ROC_AUC	0.695	0.693	0.698	**0.700**
F1	**0.679**	0.671	0.675	0.651
Precision	0.676	0.678	0.679	**0.685**
Recall	**0.709**	0.693	0.700	0.671
mAP	**0.316**	0.305	0.305	0.302

Table C.4: Impact of number of encoder layers on classification performance.

Metric	0.1	0.2	0.3	0.4	0.5
ROC_AUC	**0.702**	0.698	0.694	0.690	0.701
F1	0.662	0.666	0.667	0.676	**0.682**
Precision	**0.684**	0.681	0.676	0.676	0.681
Recall	0.681	0.687	0.689	0.699	**0.711**
mAP	**0.312**	0.309	0.311	0.303	0.309

Table C.5: Impact of dropout probability on classification performance.

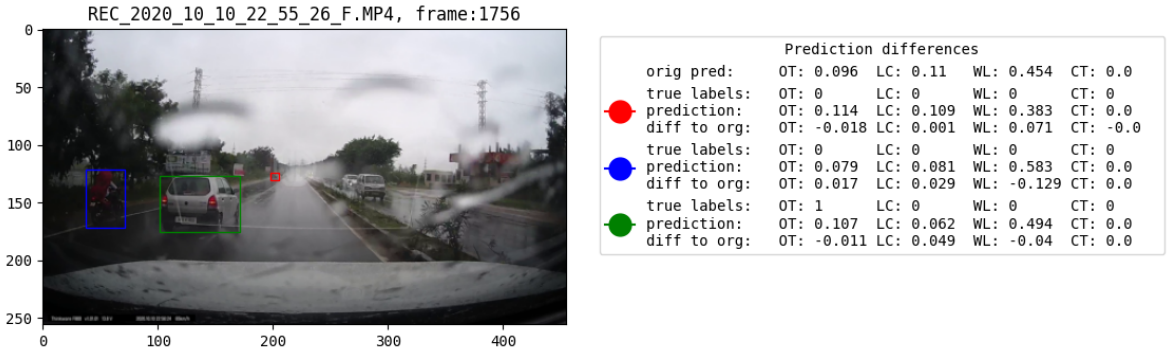
Metric	128	256	512	1024
ROC_AUC	0.680	0.682	0.690	**0.691**
F1	0.657	0.658	**0.676**	0.674
Precision	0.668	0.670	**0.676**	0.674
Recall	0.679	0.677	**0.699**	**0.699**
mAP	**0.306**	0.305	0.303	0.305

Table C.6: Impact of hidden dimension size on classification performance.

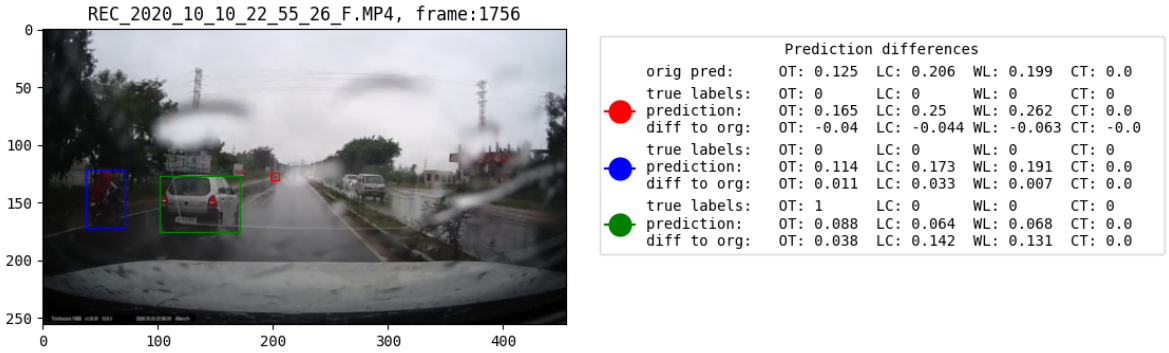
Appendix D

Visual explanations

(a) Colar



(b) OadTR_v3



(c) OadTR_v4

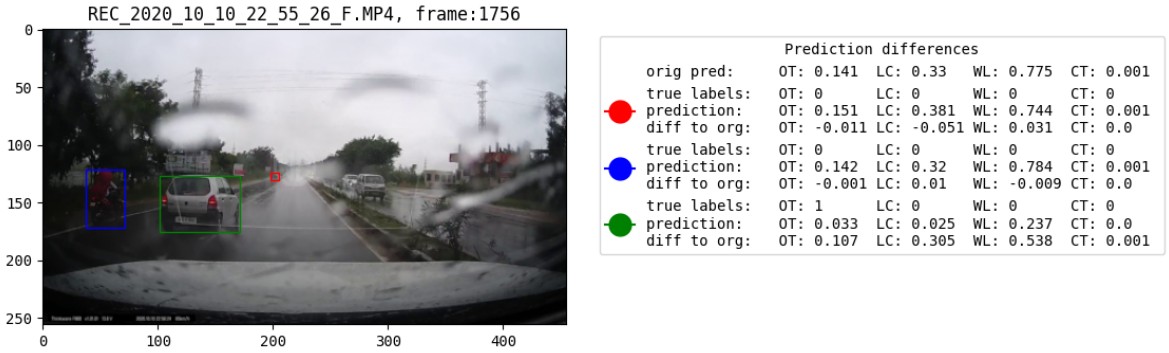
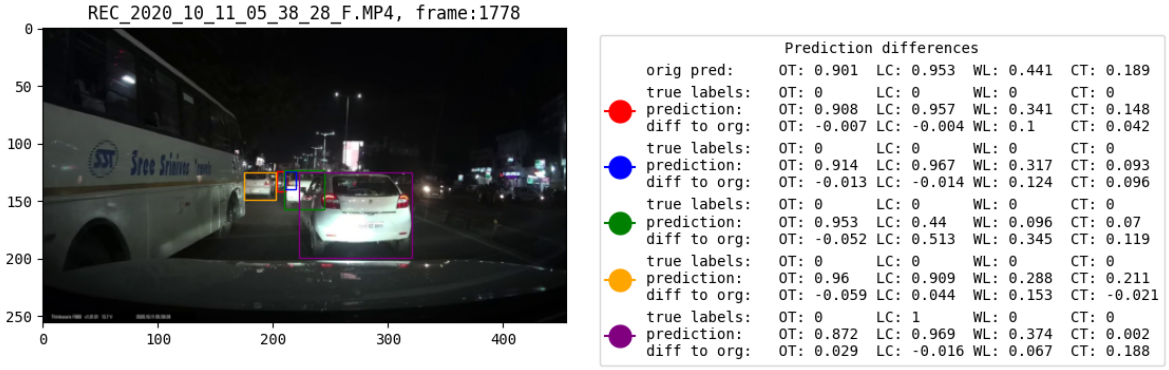
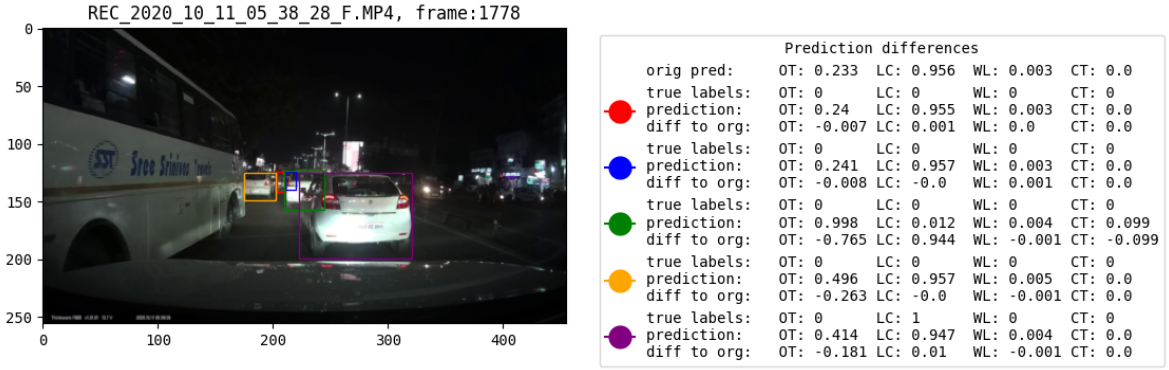


Figure D.1: Explanations for each of the three model configurations example 1

(a) Colar



(b) OadTR_v3



(c) OadTR_v4

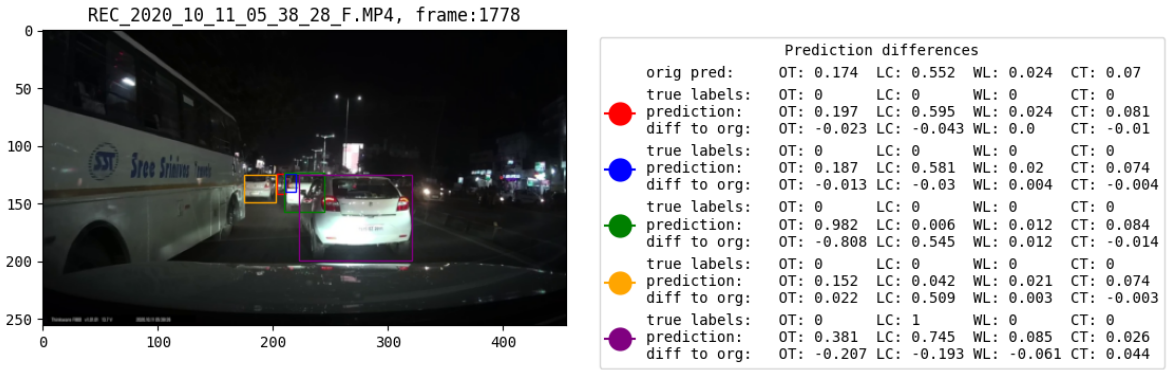


Figure D.2: Explanations for each of the three model configurations example 2

(a) Colar

Prediction differences

orig pred:	OT: 0.994	LC: 0.007	WL: 0.003	CT: 0.983
true labels:	OT: 0	LC: 0	WL: 0	CT: 1
prediction:	OT: 0.99	LC: 0.008	WL: 0.0	CT: 0.937
diff to org:	OT: 0.003	LC: -0.001	WL: 0.002	CT: 0.046
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.995	LC: 0.013	WL: 0.001	CT: 0.96
diff to org:	OT: -0.002	LC: -0.006	WL: 0.001	CT: 0.022
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.993	LC: 0.012	WL: 0.001	CT: 0.948
diff to org:	OT: 0.001	LC: -0.005	WL: 0.002	CT: 0.035
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.729	LC: 0.001	WL: 0.001	CT: 0.001
diff to org:	OT: 0.265	LC: 0.006	WL: 0.002	CT: 0.981
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.997	LC: 0.003	WL: 0.002	CT: 0.996
diff to org:	OT: -0.004	LC: 0.004	WL: 0.0	CT: -0.014
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.996	LC: 0.003	WL: 0.002	CT: 0.986
diff to org:	OT: -0.002	LC: 0.004	WL: 0.001	CT: -0.003
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.998	LC: 0.003	WL: 0.001	CT: 0.976
diff to org:	OT: -0.004	LC: 0.004	WL: 0.001	CT: 0.006
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.995	LC: 0.004	WL: 0.001	CT: 0.985
diff to org:	OT: -0.002	LC: 0.003	WL: 0.002	CT: -0.003
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.965	LC: 0.029	WL: 0.003	CT: 0.061
diff to org:	OT: 0.029	LC: -0.022	WL: -0.0	CT: 0.921
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.998	LC: 0.01	WL: 0.005	CT: 0.997
diff to org:	OT: -0.005	LC: -0.003	WL: -0.002	CT: -0.014
true labels:	OT: 1	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.991	LC: 0.005	WL: 0.018	CT: 0.916
diff to org:	OT: 0.003	LC: 0.002	WL: -0.016	CT: 0.067

(b) OadTR_v3

Prediction differences

orig pred:	OT: 0.994	LC: 0.009	WL: 0.007	CT: 0.802
true labels:	OT: 0	LC: 0	WL: 0	CT: 1
prediction:	OT: 0.992	LC: 0.009	WL: 0.005	CT: 0.658
diff to org:	OT: 0.002	LC: -0.0	WL: 0.001	CT: 0.144
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.992	LC: 0.008	WL: 0.006	CT: 0.563
diff to org:	OT: 0.002	LC: 0.001	WL: 0.001	CT: 0.239
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.994	LC: 0.012	WL: 0.013	CT: 0.536
diff to org:	OT: -0.0	LC: -0.003	WL: -0.006	CT: 0.265
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.585	LC: 0.0	WL: 0.012	CT: 0.099
diff to org:	OT: 0.409	LC: 0.009	WL: -0.005	CT: 0.702
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.96	LC: 0.003	WL: 0.022	CT: 0.822
diff to org:	OT: 0.034	LC: 0.006	WL: -0.015	CT: -0.021
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.993	LC: 0.007	WL: 0.006	CT: 0.668
diff to org:	OT: 0.0	LC: 0.002	WL: 0.0	CT: 0.134
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.994	LC: 0.007	WL: 0.008	CT: 0.549
diff to org:	OT: -0.001	LC: 0.002	WL: -0.001	CT: 0.253
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.993	LC: 0.006	WL: 0.004	CT: 0.658
diff to org:	OT: 0.0	LC: 0.003	WL: 0.003	CT: 0.144
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.188	LC: 0.0	WL: 0.007	CT: 0.143
diff to org:	OT: 0.806	LC: 0.009	WL: 0.0	CT: 0.659
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.994	LC: 0.011	WL: 0.01	CT: 0.724
diff to org:	OT: -0.001	LC: -0.002	WL: -0.004	CT: 0.078
true labels:	OT: 1	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.998	LC: 0.034	WL: 0.008	CT: 0.193
diff to org:	OT: -0.004	LC: -0.025	WL: -0.002	CT: 0.608

(c) OadTR_v4

Prediction differences

orig pred:	OT: 0.991	LC: 0.038	WL: 0.005	CT: 0.039
true labels:	OT: 0	LC: 0	WL: 0	CT: 1
prediction:	OT: 0.99	LC: 0.045	WL: 0.004	CT: 0.027
diff to org:	OT: 0.001	LC: -0.007	WL: 0.001	CT: 0.012
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.987	LC: 0.022	WL: 0.005	CT: 0.033
diff to org:	OT: 0.004	LC: 0.016	WL: -0.001	CT: 0.007
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.987	LC: 0.022	WL: 0.006	CT: 0.031
diff to org:	OT: 0.003	LC: 0.016	WL: -0.002	CT: 0.008
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.74	LC: 0.001	WL: 0.008	CT: 0.001
diff to org:	OT: 0.251	LC: 0.037	WL: -0.003	CT: 0.038
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.979	LC: 0.023	WL: 0.012	CT: 0.062
diff to org:	OT: 0.012	LC: 0.015	WL: -0.008	CT: -0.023
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.988	LC: 0.028	WL: 0.005	CT: 0.038
diff to org:	OT: 0.002	LC: 0.01	WL: -0.0	CT: 0.001
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.99	LC: 0.028	WL: 0.005	CT: 0.036
diff to org:	OT: 0.0	LC: 0.01	WL: -0.001	CT: 0.003
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.989	LC: 0.027	WL: 0.004	CT: 0.032
diff to org:	OT: 0.002	LC: 0.011	WL: 0.001	CT: 0.008
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.939	LC: 0.005	WL: 0.015	CT: 0.012
diff to org:	OT: 0.052	LC: 0.033	WL: -0.01	CT: 0.028
true labels:	OT: 0	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.994	LC: 0.064	WL: 0.008	CT: 0.075
diff to org:	OT: -0.004	LC: -0.026	WL: -0.003	CT: -0.035
true labels:	OT: 1	LC: 0	WL: 0	CT: 0
prediction:	OT: 0.988	LC: 0.048	WL: 0.006	CT: 0.01
diff to org:	OT: 0.002	LC: -0.01	WL: -0.001	CT: 0.03

Figure D.3: Explanations for each of the three model configurations example 3

Bibliography

[1] B. Krahé and I. Fenske, "Predicting aggressive driving behavior: The role of macho personality, age, and power of car," *Aggressive Behavior*, vol. 28, no. 1, pp. 21–29, 2002.

[2] J. Leeming, G. Mackay, K. Pole, and P. Fitzgerald, "Road accidents: prevent of punish?," 1969.

[3] L. Tasca, "A review of the literature on aggressive driving research," 2000.

[4] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[6] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis, *et al.*, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[9] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *CoRR*, vol. abs/2103.14030, 2021.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[11] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[14] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, PMLR, 09–15 Jun 2019.

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[16] M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin, R. Jenatton, L. Beyer, M. Tschannen, A. Arnab, X. Wang, C. Riquelme, M. Minderer, J. Puigcerver, U. Evci, M. Kumar, S. van Steenkiste, G. F. Elsayed, A. Mahendran, F. Yu, A. Oliver, F. Huot, J. Bastings, M. P. Collier, A. Gritsenko, V. Birodkar, C. Vasconcelos, Y. Tay, T. Mensink, A. Kolesnikov, F. Pavetić, D. Tran, T. Kipf, M. Lučić, X. Zhai, D. Keysers, J. Harmsen, and N. Houlsby, "Scaling vision transformers to 22 billion parameters," 2023.

[17] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[18] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," *arXiv preprint arXiv:1712.00409*, 2017.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[20] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[21] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *2011 International conference on computer vision*, pp. 2556–2563, IEEE, 2011.

[22] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[23] R. Ghosh, "Deep learning for videos: A 2018 guide to action recognition," *URL: http://blog. qure. ai/notes/deep-learning-for-videos-actionrecognition-review (visited on 13/05/2020)*, 2018.

[24] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," *Advances in neural information processing systems*, vol. 2, 1989.

[25] J. Xiao, D. Ma, and S. Yamane, "Optimizing 3d convolution kernels on stereo matching for resource efficient computations," *Sensors*, vol. 21, no. 20, p. 6808, 2021.

[26] Y. Zhang, X. Li, C. Liu, B. Shuai, Y. Zhu, B. Brattoli, H. Chen, I. Marsic, and J. Tighe, "Vidtr: Video transformer without convolutions," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 13577–13587, 2021.

[27] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM computing surveys (CSUR)*, vol. 54, no. 10s, pp. 1–41, 2022.

[28] Y. Zhang, B. Li, H. Fang, and Q. Meng, "Current advances on deep learning-based human action recognition from videos: a survey," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 304–311, IEEE, 2021.

[29] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman, "Video action transformer network," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 244–253, 2019.

[30] X. Wang, S. Zhang, Z. Qing, Y. Shao, Z. Zuo, C. Gao, and N. Sang, "Oadtr: Online action detection with transformers," 2021.

[31] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*, pp. 20–36, Springer, 2016.

[32] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*, pp. 20–36, Springer, 2016.

[33] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–970, 2015.

[34] J. Carreira, E. Noland, C. Hillier, and A. Zisserman, "A short note on the kinetics-700 human action dataset," *CoRR*, vol. abs/1907.06987, 2019.

[35] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7794–7803, 2018.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[37] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[38] C. Pinhanez and A. Bobick, "Human action detection using pnf propagation of temporal constraints," in *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, pp. 898–904, 1998.

[39] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1473–1488, 2008.

[40] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[41] D. Gavrila and V. Philomin, "Real-time object detection for "smart" vehicles," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1, pp. 87–93 vol.1, 1999.

[42] M. Xu, L.-Y. Duan, C. Xu, M. Kankanhalli, and Q. Tian, "Event detection in basketball video using multiple modalities," in *Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint*, vol. 3, pp. 1526–1530 vol.3, 2003.

[43] I. Laptev, "On space-time interest points," *International journal of computer vision*, vol. 64, pp. 107–123, 2005.

[44] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, Ieee, 2005.

[45] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[46] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model.," in *CVPR*, vol. 92, pp. 379–385, 1992.

[47] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *Advances in neural information processing systems*, vol. 27, 2014.

[48] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[49] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[50] R. De Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars, "Online action detection," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pp. 269–284, Springer, 2016.

[51] J. Gao, Z. Yang, and R. Nevatia, "Red: Reinforced encoder-decoder networks for action anticipation," *arXiv preprint arXiv:1707.04818*, 2017.

[52] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International journal of computer vision*, vol. 103, pp. 60–79, 2013.

[53] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of the IEEE international conference on computer vision*, pp. 3551–3558, 2013.

[54] A. Clark, "Whatever next? predictive brains, situated agents, and the future of cognitive science," *Behavioral and brain sciences*, vol. 36, no. 3, pp. 181–204, 2013.

[55] M. Xu, M. Gao, Y.-T. Chen, L. S. Davis, and D. J. Crandall, "Temporal recurrent networks for online action detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[56] L. Yang, J. Han, and D. Zhang, "Colar: Effective and efficient online action detection by consulting exemplars," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3160–3169, 2022.

[57] D. Castelvecchi, "Can we open the black box of ai?," *Nature News*, vol. 538, no. 7623, p. 20, 2016.

[58] R. Goebel, A. Chander, K. Holzinger, F. Lecue, Z. Akata, S. Stumpf, P. Kieseberg, and A. Holzinger, "Explainable ai: the new 42?," in *Machine Learning and Knowledge Extraction: Second IFIP TC 5, TC 8/WG 8.4, 8.9, TC 12/WG 12.9 International Cross-Domain Conference, CD-MAKE 2018, Hamburg, Germany, August 27–30, 2018, Proceedings 2*, pp. 295–303, Springer, 2018.

[59] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[60] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

[61] J. Aechtner, L. Cabrera, D. Katwal, P. Onghena, D. P. Valenzuela, and A. Wilbik, "Comparing user perception of explanations developed with xai methods," in *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–7, IEEE, 2022.

[62] H. Chefer, S. Gur, and L. Wolf, "Transformer interpretability beyond attention visualization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 782–791, 2021.

[63] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," *arXiv preprint arXiv:1605.01713*, 2016.

[64] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International conference on machine learning*, pp. 3319–3328, PMLR, 2017.

[65] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," *arXiv preprint arXiv:1706.03825*, 2017.

[66] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

[67] L. Hiley, A. Preece, Y. Hicks, S. Chakraborty, P. Gurram, and R. Tomsett, "Explaining motion relevance for activity recognition in video deep learning models," *arXiv preprint arXiv:2003.14285*, 2020.

[68] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern recognition*, vol. 65, pp. 211–222, 2017.

[69] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.

[70] J. Gu, Y. Yang, and V. Tresp, "Understanding individual decisions of cnns via contrastive backpropagation," in *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pp. 119–134, Springer, 2019.

[71] B. K. Iwana, R. Kuroki, and S. Uchida, "Explaining convolutional neural networks using softmax gradient layer-wise relevance propagation," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 4176–4185, IEEE, 2019.

[72] R. Fong, M. Patrick, and A. Vedaldi, "Understanding deep networks via extremal perturbations and smooth masks," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2950–2958, 2019.

[73] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[74] L. Wang, Z. Li, and M. Li, "Impact of personality traits on driving behavior: A systematic review," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 55, pp. 342–358, 2018.

[75] A. T. McCartt, D. R. Mayhew, K. A. Braitman, S. A. Ferguson, and H. M. Simpson, "Effects of age and experience on young driver crashes: review of recent literature," *Traffic Inj. Prev.*, vol. 10, pp. 209–219, June 2009.

[76] N. Harré, J. Field, and B. Kirkwood, "Gender differences and areas of common concern in the driving behaviors and attitudes of adolescents," *Journal of Safety Research*, vol. 27, no. 3, pp. 163–173, 1996.

[77] M. Wiberg, "Gender differences in the swedish driving-license test," *Journal of Safety Research*, vol. 37, no. 3, pp. 285–291, 2006.

[78] E. R. S. Observatory, "Driver distraction," 2018.

[79] M. Kilpeläinen and H. Summala, "Effects of weather and weather forecasts on driver behaviour," *Transportation research part F: traffic psychology and behaviour*, vol. 10, no. 4, pp. 288–299, 2007.

[80] S. M. Simmons, J. K. Caird, F. Sterzer, and M. Asbridge, "The effects of cannabis and alcohol on driving performance and driver behaviour: a systematic review and meta-analysis," *Addiction*, vol. 117, no. 7, pp. 1843–1856, 2022.

[81] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proceedings of the National Academy of Sciences*, vol. 116, no. 50, pp. 24972–24978, 2019.

[82] S. Anthony, "Self-driving cars still can't mimic the most natural human behavior." Quartz, Aug 2017. Accessed: 2023-03-10.

[83] A. Aljaafreh, N. Alshabatat, and M. S. N. Al-Din, "Driving style recognition using fuzzy logic," in *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, pp. 460–463, IEEE, 2012.

[84] W. Wang, J. Xi, A. Chong, and L. Li, "Driving style classification using a semisupervised support vector machine," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 5, pp. 650–660, 2017.

[85] J. L. Deffenbacher, E. R. Oetting, and R. S. Lynch, "Development of a driving anger scale," *Psychological reports*, vol. 74, no. 1, pp. 83–91, 1994.

[86] R. Chandra, U. Bhattacharya, T. Mittal, A. Bera, and D. Manocha, "Cmetric: A driving behavior measure using centrality functions," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2035–2042, IEEE, 2020.

[87] A. Mavrogiannis, R. Chandra, and D. Manocha, "B-gap: Behavior-guided action prediction and navigation for autonomous driving," *arXiv preprint arXiv:2011.03748*, 2020.

[88] U. N. E. C. for Europe, "Aggressive driving behaviour: Background paper," 2018. Accessed on [January 12, 2023].

[89] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8748–8757, 2019.

[90] V. Ramanishka, Y.-T. Chen, T. Misu, and K. Saenko, "Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7699–7707, 2018.

[91] R. Chandra, M. Mahajan, R. Kala, R. Palugulla, C. Naidu, A. Jain, and D. Manocha, "Meteor: A massive dense & heterogeneous behavior dataset for autonomous driving," *arXiv preprint arXiv:2109.07648*, 2021.

[92] V. Ramanishka, Y.-T. Chen, T. Misu, and K. Saenko, "Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[93] S. Dokania, A. Hafez, A. Subramanian, M. Chandraker, and C. Jawahar, "Idd-3d: Indian driving dataset for 3d unstructured road scenes," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 4482–4491, 2023.

[94] P. Cong, X. Zhu, F. Qiao, Y. Ren, X. Peng, Y. Hou, L. Xu, R. Yang, D. Manocha, and Y. Ma, "Stcrowd: A multimodal dataset for pedestrian perception in crowded scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19608–19617, 2022.

[95] H. Quispe, J. Sumire, P. Condori, E. Alvarez, and H. Vera, "I see you: A vehicle-pedestrian interaction dataset from traffic surveillance cameras," *arXiv preprint arXiv:2211.09342*, 2022.

[96] R. Chandra, A. Bera, and D. Manocha, "Using graph-theoretic machine learning to predict human driver behavior," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2572–2585, 2021.

[97] M. Baptista-Ríos, R. J. López-Sastre, F. Caba Heilbron, J. C. Van Gemert, F. J. Acevedo-Rodríguez, and S. Maldonado-Bascón, "Rethinking online action detection in untrimmed videos: A novel online evaluation protocol," *IEEE Access*, vol. 8, pp. 5139–5146, 2020.

[98] PyTorch, "Pytorch vision models," 2022.

[99] "Scikit-learn: Machine learning in Python." Accessed: 2023-06-11.

[100] V. Godbole, G. E. Dahl, J. Gilmer, C. J. Shallue, and Z. Nado, "Deep learning tuning playbook," 2023. Version 1.

[101] S. Lee, H. Eun, J. Moon, S. Choi, Y. Kim, C. Jung, and C. Kim, "Learning to discriminate information for online action detection: Analysis and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–17, 2022.

[102] M. Contributors, "Openmmlab's next generation video understanding toolbox and benchmark." https://github.com/open-mmlab/mmaction2, 2020.

[103] S. Zhao, L. Zhao, Z. Zhang, E. Zhou, and D. Metaxas, "Global matching with overlapping attention for optical flow estimation," 2022.

[104] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.

[105] PyTorch, "Pytorch vision models," 2022.

[106] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.