UNIVERSITÀ
DEGLI STUDI
FIRENZE

PhD Program in Smart Computing
Dipartimento di Ingegneria dell'Informazione (DINFO)

Deep learning methods for safety-critical driving events analysis

Matteo Simoncini

Dissertation presented in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Smart Computing

PhD Program in Smart Computing
University of Florence, University of Pisa, University of Siena

Deep learning methods for safety-critical driving events analysis

Matteo Simoncini

Advisor:

Prof. Fabio Schoen

Head of the PhD Program:

Prof. Paolo Frasconi

Evaluation Committee:
Prof. Stefano Ghidoni, *University of Padova*
Prof. Miguel A. Perez, *Virginia Tech*

XXXIV ciclo — October 2022

To Pierina and my family.

There is no shame in not having solved a challenging problem yet.
- Andrew D. Bagdanov

Acknowledgments

I would like to express my gratitude to my supervisor Prof. Fabio Schoen, who supported and helped me during my Ph.D. His wisdom and experience many times over the three years contribute to steer from the non-proficuos path I was undergoing for a more proficuos one. I would like to thank my Supervisory Committee, Prof. Andrew Bagdanov and Prof. Samuele Salti, who reviewed my work and gave me extremely useful insights, that helped to improve my research. Finally, I would like to thank Verizon Connect for giving me the great opportunity of undergoing this journey, which made me grow as a researcher and as a person. In particular, I would like to thank Dr. Francesco Sambo, Dr. Leonardo Taccari, Dr. Douglas Coimbra de Andrade, Dr. Alessandro Lori and all the Verizon Connect Research team in Florence for their support and useful feedback over the course of all three years.

Abstract

Road scene understanding, with a particular focus on road safety, is crucial in applications aiming at preventing car crashes and severe injuries on the road. In this thesis, we propose to study the data of crash and near-crash events, collectively called safety-critical driving events. Such data include a footage of the event, acquired from a camera mounted inside the vehicle, and the data from a GPS/IMU module, *i.e.*, speed, acceleration and angular velocity.

We introduce a novel problem, that we call *unsafe maneuver classification*, that aims at classifying safety-critical driving events based on the maneuver that leads to the unsafe situation. The set of dangerous maneuvers is not known a priori, as the scenarios ultimately resulting in a dangerous situation for the driver are extremely diverse, thus we first propose a taxonomy that groups the maneuvers in macro-classes. We grounded the classes definition on a Naturalistic Driving Study (NDS) and, thus, they are representative of the real distribution of unsafe events.

To address unsafe maneuver classification, we first propose a two-stream neural architecture based on Convolutional Neural Networks (CNNs). Such architecture is formed of three blocks: a video processing backbone, a sensor processing module and a two-stream module, that performs sensor fusion between the first two and addresses the classification task. We propose a fine-tuning strategy for the video processing backbone that achieves the state-of-the-art on the problem.

Then, we propose to integrate the output of an object detector in the classification task, to provide the network explicit knowledge of the entities in the scene. We design a specific architecture that leverages a tracking algorithm to extract information of a single real-world object over time, and then uses attention to ground the prediction on a single (or a few) objects, *i.e.*, the dangerous or in danger ones, leveraging a solution that we called Spatio-Temporal Attention Selector (STAS).

Finally, we propose to address video captioning of safety-critical events, with the goal of providing a description of the dangerous situation in a human-understandable form. We use an encoder-decoder architecture, where the encoder is the same architecture used for the unsafe maneuver classification task. As decoder, we test multiple architectures that are recommended in the video captioning literature and find an attention-based hierarchical decoder, *i.e.*, having two nested asynchronous recurrent networks, respectively responsible for the next paragraph and the next word generation, to be the one yielding the best results.

Contents

Chapter 1

Introduction

Car crashes are one of the major problems of the modern era. According to a report of the World Health Organization[†], in 2016 the number of road traffic death reached 1.35 million worldwide, becoming the 8th leading cause of death among people of all the age groups and the leading cause of death for people aged 5-29 years. To cope with this phenomenon, the various administrations have adopted a tight legislation to enforce best practices while driving, as well as accounting for safety aspects when designing, planning and maintaining roads. On the vehicle side, instead, the usage of in-vehicle electronic safety features, such as Electronic Stability Control (ESC) and Advanced Emergency Braking Systems (AEBS), has shown to actively contribute to reduce road traffic deaths and to mitigate the severity of injuries. Moreover, in case a severe accident occurs, the timely activation of post-crash care for the injured can save lives and mitigate the consequences of the crash.

In this scenario, research aimed at getting a better understanding of the road scene with regard to safety and, for instance, in the detection or anticipation of crashes and near-crashes (collectively called safety-critical events) is thus extremely important, as it might further contribute to mitigate the problem. We can roughly divide the works in this field into two categories: online algorithms, as forward collision warning systems and ADAS (Advanced Driver Assistance Systems), to be executed inside the vehicle and that aim at directly act in case of a critical situation, and offline (or after-the-fact) algorithms, used to get some form of understanding of the event. Such knowledge can be used for post-crash service automatic activation, but also to selectively record the dangerous events and use them in order to coach the driver and improve their driving skills, or to provide evidence in case of a traffic-related crash.

[†] `https://www.who.int/publications/i/item/9789241565684`

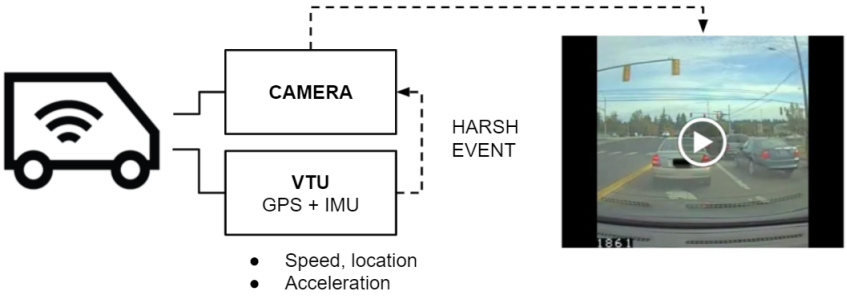


Figure 1.1: Connected vehicles context: a dashcam and a VTU are installed inside the vehicle. Once an harsh event is detected, the system records a footage of a few seconds before and after the event as well as the sensor data.

Such algorithms may use different sources of data, including sensors installed inside the vehicle, *e.g.*, an accelerometer or a gyroscope, collectively called Inertial Motion Unit (IMU), a Global Positioning System (GPS) device, or data from the diagnostic port of the vehicle, accessing information as revolution per minute (RPM), pedal position and steering wheel position. In recent years, car manufacturers and private car-owners have begun to install dashboard cameras (*dashcams*) to acquire footage of the road scene in front of the driver and/or of the situation inside the cabin. The availability of such videos unlocked the possibility to study new safety-related algorithms, that have an awareness of the road scene around the vehicles and of the situation inside the cabin, *e.g.*, forward collision warning and drowsy driver detection. In parallel, the capability to process and analyze image and sensor data made a huge leap forward, experiencing a true revolution in the past 10 years, with the advent of Deep Learning. Such algorithms obtained outstanding results and outperformed the previous approaches in most of the problems of the machine learning literature, including classification, regression and generation problems, and on various types of data, such as images, audio, video and other sensors. Deep Learning algorithms, though, require a massive amount of data and a fair amount of computational resources to be trained: this became possible only with the recent technological advancements in the storage and transmission of the data and on the widespread diffusion of hardware capable to parallelize the mathematical operations.

1.1 Goals of this thesis

In this thesis, we aim to tackle the analysis of safety-critical events, seeking for a better understanding of the events in the road scene, that could lead to applications used to increment the overall level of safety. However, our first challenge is to define the notion of unsafety and, by extension, of safety-critical events in order to

detect and analyze them. In fact, in contrast with online algorithms that could be executed on the vehicle if equipped with the proper computation hardware, for the after-the-fact methods to stream to a processing server and to analyze every portion of driving in order to seek for dangerous situations is generally both unfeasible, as it would require a massive amount of resources, and useless, as the vast majority of the processed driving segments would be of safe driving. Thus, it is a common practice in the research literature and in the industry to detect safety-critical events via kinematic triggers, as reported in Figure 1.1. A system is installed inside a vehicle, comprehensive of a forward-facing camera mounted on the dashboard of the vehicle and a Vehicle Tracking Unit (VTU), composed of a GPS and an IMU module, to have access to the speed, acceleration and angular velocity of the vehicle. The GPS/IMU module acts both as a recording device and as a trigger. Once a peak in acceleration is detected, *e.g.,* a spike of a given magnitude for a given amount of seconds, a so-called *harsh event* is triggered and the relative data gets recorded. However, this method produces a fair amount of false positive and false negative events. For instance, an animal running across the vehicle with no attempt of brake by the driver, that is missed by a small margin would not trigger an event. On the other hand, a harsh acceleration in a parking lot, far away from all the other vehicles would trigger an event. Thus, despite this method being good to acquire potential dangerous events, further processing is required.

A trend in the literature is to consider the observable consequences of the harsh driving event, for instance Taccari et al. (2018) considered a classification task, distinguishing between crashes, near-misses and safe events. Instead, as highlighted by Dozza and Gonzalez (2012), the dangerousness of a driving event depends on the full chain of events that lead to the crash or the near-crash and is intrinsically related to change related to the environmental condition of the road scene and the perception of the danger of the driver. For this reason, in this thesis, we aim at studying the safety-critical events by considering the maneuver that leads to the dangerous situation, by tackling a novel task that we called unsafe maneuver classification.

Furthermore, to try to get a better understanding of the reasons leading to a dangerous situation, we follow a recent trend and focus on explainability, *i.e.,* to not just provide a good classification score but also to provide the motivations behind a given output. We address this aspect in a two-fold way. First, we study an architecture that leverages the output of an object detection algorithm on the video to perform the classification focussing solely on the relevant objects in the scene, *i.e.,* the dangerous ones or the ones in danger. Second, we address the video captioning task, which provides a description of the sequence of actions ultimately leading to the safety-critical situation.

1.2 Contributions and Organization

This thesis is structured as follows.

- In Chapter 2 we present the unsafe maneuver classification task. Such task is a multi-class classification problem that considers as input the video, acquired from a dashcam, centered around the safety-critical event and the data from the GPS/IMU sensor and as output a label representing the maneuver that leads to the dangerous situation. We consider maneuvers performed by both the ego-vehicle and other vehicles, together with other unsafe situations. We grounded the definition of our classes on a dataset from a Naturalistic Driving Study (SHRP2 NDS, described in Hankey et al. (2016a)), a collection of safety-critical events acquired by installing an acquisition system like the one described above on a large number of vehicles over multiple years. Thus, our labels consider the real distribution of the unsafe events on the road. We released the annotations we used to the scientific community for reproducibility of the results and to push forward the research in this direction.

- In Chapter 3 we introduce a novel end-to-end architecture based on convolutions that tackles the unsafe maneuver classification task by performing sensors fusion between the video and the sensor stream. Also, we studied and tackled several practical problems when handling such data, such as asynchronicity between the video stream and the sensors stream, missing data due to GPS occlusions, and sensors drift.

- In Chapter 4 we expand the architecture presented in Chapter 3 in two ways. First, we include the output of an object detector. The motivations behind this choice are to boost the overall classification performance by leveraging pretrained knowledge (*i.e.*, the objects in the scene) that could be hard to get on a relatively small sample base, but also to be able to bound the final prediction to a given object into a given set of frames. To do this, we introduce a module that we called Spatio-Temporal Attention Selector (STAS), that combines the attention mechanism with a tracking algorithm to extract and select features relative to a real-word object (*e.g.*, a vehicle) over time and select the most relevant one, *i.e.*, the dangerous one or the one in danger, doing a step toward explainability. Second, we used 2D depthwise separable convolutions over the sensors stream, extracting features that maintain the semantic meaning of the individual sensor, confirming that their effectiveness, already proven in the image and audio literature, extends to our application domain. We show how the proposed novelties contribute to a new *state-of-the-art* in the unsafe maneuver classification task.

- In Chapter 5 we address video captioning of safety-critical events. We annotate and release to the scientific community the SHRP-X dataset, a collection of 3000 multi-sentence annotations describing each individual and relevant action, *i.e.*, maneuver, in the video that ultimately leads to the crash or the near-crash. We propose an encoder-decoder architecture, using as encoder the architecture of Chapter 4, pre-trained on the unsafe maneuver classification task. As decoder, we tested various architectures that proved effective in the video-captioning literature. Furthermore, we show how the usage of task-specific attributes could give a boost to the overall classification performance.
- In Chapter 6 we present the conclusions we draw from this thesis and present future lines of research.

Please note that, while most of my work was on the safety-critical analysis, that is reported in this thesis, during my Ph.D. I have been involved in other research topics, most of which resulted in scientific papers. A list of such papers can be found in Appendix A.

Chapter 2

Unsafe maneuver classification [†]

In this Chapter, we present the novel unsafe maneuver classification task, that aims at classifying the safety-critical driving events on the action that leads to the dangerous situations starting from a video acquired from a dashcam and the GPS/IMU sensors. Such maneuver could be performed by the subject vehicle or by other vehicles in the scene, and just one or more vehicles could be involved. The definition of the unsafe maneuver classification classes are grounded on a Naturalistic Driving Study (NDS) and, thus, are representative of the real distribution of the dangerous events.

Dashcam data and, optionally, the sensors data from the GPS/IMU module have been used to tackle various problems in applications like insurance, fleet management and self-driving vehicles, mainly to detect and classify (Taccari et al. (2018); Yao et al. (2019); Zhu et al. (2019)) or to anticipate (Chan et al. (2016), Suzuki et al. (2018)) car crashes and to address driving maneuver detections (Peng et al. (2018); Zekany et al. (2019); Deo et al. (2018)).

Crash detection aims at detecting the dangerous event based on the presence of a collision (crash) between the subject vehicle and another entity on the road scene. Such entity could be another vehicle, but also an object or a part of the road itself, like a road bump or a shoulder. Here, the classification task address only the final result of the dangerous situation, and the causes that lead to the dangerous situation or the notion of unsafety is not addressed. In fact, approaches that do not consider

[†] Part of the content of this chapter was published in:

- M. Simoncini, D. Coimbra de Andrade, S. Salti, L. Taccari, F. Schoen, F. Sambo, "Two-stream neural architecture for unsafe maneuvers classification from dashcam videos and GPS/IMU sensors", *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages:1–6, 2020.

the video at all but just the sensor data have proven to be effective, as in Kubin et al. (2021). In the crash anticipation, instead, the concept of unsafety has to be taken into account, but is done in a general way, aiming at predicting a car crash and doesn't explicitly take into consideration the causes, *i.e.,* the model trained on this task would detect the presence of a dangerous situation but it will be hard to understand why. In the maneuver detection field, instead, the aim is to detect and classify the maneuvers performed either by the subject vehicle or by other vehicles (with respect to the subject vehicle), *e.g.*, change of lanes, turns at intersection, accelerations and decelerations. In this case, only the subject maneuvers or the other drivers' maneuvers are taken into consideration but not both and, again, the unsafety of the scene is never taken into account.

We propose, instead, to address what we called the unsafe maneuver classification problem, with the goal of classifying safety-critical driving events based on the maneuver that leads to the dangerous situation. In the task, we consider both crashes and near-crashes, and the key to perform a correct classification is to identify the action, among the set of maneuvers in a driving scene, that ultimately lead to the dangerous situation. Moreover, we consider maneuvers performed both by the subject (*subject vehicle* `SV` maneuvers, sometimes also referred to as *ego-vehicle maneuvers*) and by other vehicles (*non-subject vehicle* `NSV` maneuvers). Furthermore, we consider both maneuvers involving multiple vehicles (*e.g.*, improper lane change, turning) and single-vehicle maneuvers (*e.g.*, loss of vehicle control, vehicle over the edge of the road). To the best of our knowledge, no work has addressed unsafe maneuver classification in such a general way.

2.1 Unsafe maneuvers taxonomy

One of the first requirements is to define a taxonomy for safety-critical events based on the maneuver that led to the dangerous situation and, thus, to answer to the question *which are the maneuver that could lead to a dangerous situation?*. This first simple step is challenging on its own, as the variety of maneuvers leading to a dangerous situation in a road environment is extremely broad, *e.g.,* short distance to another object, colliding trajectories, violation of the right of way or other road laws (Yao et al. (2019)). Most of the existing works based their results on manually generated crowd-sourced datasets (*e.g.,* acquired from YouTube videos, as in Chan et al. (2016); Yao et al. (2019); Zhu et al. (2019)). However, this approach is not ideal, as not all the possible maneuvers are equally likely to be depicted in videos uploaded online: for instance, a driver may decide not to upload videos where he is at fault. Moreover, the definition of safety-relevant events, and by extension of unsafe maneuvers leading to them, is not unambiguous and is both related to the environmental condition of the road scene and the perception of danger of the driver,

Table 2.1: The proposed unsafe maneuvers taxonomy. In the first column, the maneuver identifier; in the second one, the maneuver description; in the third one, the total number of maneuvers of that type in the dataset.

Class	Description	Total
SL	*Subject lane change*. The subject performs an improper lane change, potentially from an adjacent lane, an acceleration or deceleration lane or from a parallel parking spot, drawing dangerously close to another vehicle in another lane, being it in front of the vehicle, behind the vehicle and/or with potential sideswipe threat. Alternatively, the subject invaded the lane of a car coming in the opposite direction.	283
ST	*Subject turn*. The subject performs an improper turn, potentially at an intersection, from a driveway or from a perpendicular parking spot, invading the lane or space of another vehicle proceeding in the same or opposite direction of the vehicle.	271
NSL	*Non-subject lane change*. As SL but with another vehicle being the one performing the unsafe maneuver.	1507
NST	*Non-subject turn*. As ST but with another vehicle being the one performing the unsafe maneuver.	954
SB	*Subject brakes*. The subject vehicle brakes to avoid the collision with another vehicle in the same lane and going in the same direction, potentially performing an evasive maneuver.	3115
SOE	*Subject over edge*. The subject vehicle runs over the edge of the road or collides with road boundaries.	1069
SLC	*Subject lost control*. The subject vehicle looses control due to road condition, excessive speed or other causes.	196
SO	*Subject other maneuver*. Other unsafe maneuvers performed by the subject vehicle.	167
NSO	*Non-subject other maneuver*. As SO but with another vehicle being the one performing the unsafe maneuver.	183
O	*Other*. Collision or near-collision with animals, pedestrians, pedal-cyclists or other objects.	742

as highlighted in Dozza and Gonzalez (2012). To counteract this problem, multiple reviewers should be used and clear definitions to label events should be agreed upon. Finally, in contrast with previous work focusing only on ego-vehicle maneuvers (Peng et al. (2018); Zekany et al. (2019)) or other vehicles maneuvers (Deo et al. (2018); Breuer et al. (2019)), our aim is to define a broad categorization that considers all possible reasons leading to safety-critical events, *i.e.,* ego-vehicle maneuvers, other vehicle maneuvers but also poor road condition, the presence of objects in the roadway, animals, *etc*.

Therefore, to create a taxonomy that is representative of the real distribution of safety-critical events while addressing the above concerns, we propose to base it on a large Naturalistic Driving Study (NDS), in particular the SHRP2 dataset (Hankey et al. (2016b)). The SHRP2 NDS dataset is a collection of more that 8800 safety-critical events, gathered by more than 3300 drivers between 2010 and 2013. These events have been manually annotated with event-, driver- and environment-related variables, for a total of 75 labels Hankey et al. (2016b). Multiple round of reviews were performed to validate the annotations, and careful and unambiguous definitions of all the labels attached to maneuvers and events are provided: this greatly reduces the inherent ambiguity of the derived taxonomy.

In particular, each safety-critical event in the dataset has been labeled with the start and the end of the event and the so-called *precipitating event*, *i.e., "The state of environment or action that began the event sequence under analysis"*, answering the question *"but for this action, would the crash or near-crash have occurred?"*, for a total of 64 different annotations. By using these annotations as our starting point, we define a set of classes, aggregating similar SHRP2 annotations and manually relabelling the ones not falling perfectly into a category. The resulting classes among with an in-depth description for each of them are reported Table 2.1.

It is worth mentioning, as highlighted in Yao et al. (2019), that the distribution of the safety-critical events has *a long tail*, thus it is intrinsically an umbalanced problem. For instance, as reported in Table 2.1, `SB` is the most common maneuver by far. In contrast, some of the precipitating events have too few examples to constitute a statistically significant sample size. To cope with this problem, we created the classes `SO` and `NSO` containing the remaining unsafe maneuvers performed respectively by the subject vehicle and by other vehicles.

The result is a ten-class classification problem. It is interesting to observe that some of the classes are relative to the same maneuver but differ on the entity performing it, like `SL` and `NSL`, `ST` and `NST`, `SO` and `NSO`. It is important to remember, though, that the point-of-view is always that of the subject vehicle. Thus, although the couples of maneuvers are the same if observed from a bird-eye view perspective, they are totally different samples in the dataset. For instance, in the `NST` samples the subject driver is often going straight or performing another unrelated maneuver

(*e.g.*, a lane change) when another vehicle turns across his path, for instance from an intersection. In this case, the background and the vehicle going in the same direction are moving in a steady way toward the image plane and basically, the only things that do not follow that flow are the vehicle coming in the opposite direction the vehicle performing the unsafe turn. Moreover, the GPS/IMU data does not hold generally any relevant information but the subject breaking and an eventual evasive maneuver. In the ST samples, instead, the whole background is moving as the vehicle starts the turn. The other vehicle, *i.e.*, the one in danger, might be visible at the very last moment or might fall only partially in the video, being, for instance, on the subject vehicle side. Here, instead, the GPS/IMU data plays a key role: the turn is clearly detectable via the gyroscope that is followed often by a brake or an evasive maneuver.

Finally, let us highlight how all the maneuvers can fall into three macro-category: maneuvers performed by the subject that affect other vehicles, maneuvers performed by other vehicles that affect the subject vehicle, maneuvers performed by the subject on its own. Thus, the proposed taxonomy could be used as a proxy to assess the fault of the of the unsafe situation.

We released the labels obtained from the SHRP dataset according to our taxonomy to the scientific community to push forward the research on this topic*.

*https://github.com/mattsim/shrp2-unsafe-maneuver

Chapter 3

Two-stream neural architecture for unsafe maneuvers classification †

In this chapter, the unsafe maneuver classification task is addressed introducing a novel end-to-end architecture based on convolutional neural networks (CNNs). The architecture combines the data from the GPS/IMU module inside the vehicle and the video recorded from the dashcam performing sensor fusion. We discuss the challenges relative to handling such heterogeneous types of sensors. We also introduce a simple but effective methodology to increase the benefit of fine-tuning the backbone network. Moreover, we perform an exhaustive literature review, showing how the task differs from traditional video classification tasks and presenting the related works of the relevant transportation literature.

The unsafe maneuver classification task, introduced in Chapter 2, addresses the classification of safety-critical events (crashes and near-crashes) based on the maneuver that leads to the dangerous situation, starting from the video acquired from a dashcam and the GPS/IMU data. While this task could be addressed using just the video as an input, as is has been explored in closely-related fields of research, we believe that the usage of GPS/IMU data would have a great impact in the classification of the unsafe maneuvers performed by the subject. On the other hand, the

† Part of the content of this chapter was published in:

- M. Simoncini, D. Coimbra de Andrade, S. Salti, L. Taccari, F. Schoen, F. Sambo, "Two-stream neural architecture for unsafe maneuvers classification from dashcam videos and GPS/IMU sensors", *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages:1–6, 2020.
- M. Simoncini, D. Coimbra de Andrade, L. Taccari, S. Salti, L. Kubin, F. Schoen, F. Sambo, "Unsafe maneuver classification from dashcam video and GPS/IMU sensors using Spatio-Temporal Attention Selector", *IEEE Transactions on Intelligent Transportation Systems*, 2022.

video stream is crucial for the detection of the maneuver performed by the other vehicles and relative to other entities on the road.

Thus, we propose to address the aforementioned problem with a novel end-to-end deep learning approach and a new two-stream architecture that has two roles. First, it aligns the two heterogeneous inputs and performs sensor fusions; in fact, generally, the sensors and the video has different sampling rate and the two sampling clocks are misaligned. Second, it automatically extracts features from the two streams to perform the classifications. The video stream is processed using a 2D Convolutional Neural Network (CNN) applied to each frame. We test both the usage of a pre-trained backbone from the literature and we propose a fine-tuning methodology on the unsafe maneuver classification labels. The sensor stream is instead processed using a 1D CNN, that is then merged to the video stream via concatenation and is finally fed to another 1D CNN that extracts features on the combination of the two sensors and performs the final classification.

3.1 Related works

To the best of our knowledge, the problem proposed in this chapter has not been addressed before. The most closely related works in the literature are in the field of *Accident detection / anticipation* and *Driving maneuver detection*. Moreover, in this Section, we provide a review of the methods in the video classification literature, trying to highlight which directions we believe to be best suited to address the unsafe maneuver classification task.

Video classification

The first attempts to perform video classification extended the results obtained on the image classification tasks. Karpathy et al. (2014) applied a 2D CNN over each frame of the video independently to address activity recognition. They experimented with different aggregation over time strategies, including single-frame processing, early fusion, which concatenates multiple frames at the pixel level, late fusion, which concatenates the backbone outputs of multiple frames, and slow fusion, a combination of the latter two that works on multiple frames and progressively merges the activations through the network layers. From their results, the slow fusion approach is outperforming the single-frame one by a small margin, showing that in this task the temporal information is less crucial. Yue-Hei Ng et al. (2015) extended this approach by first using two CNNs for feature extraction: the first one working on the appearance RGB stream and the second one on the optical flow computed on the video. Again, they tried different aggregation strategies, introducing also simple max-pooling over time, time-domain convolution over the resulting frames

and the usage of an LSTM over the resulting feature vectors for each frame to explicitly model the temporal evolution in the video. In their experiments, the LSTM approach was outperforming the remaining approaches, closely followed by simple pooling. Feichtenhofer et al. (2017) extended further the two-stram approach by adding residual connection between the appearance CNN and the flow CNN.

In parallel, a set of works in the literature used 3D CNNs to tackle video classification, as they are a more natural fit. Tran et al. (2015) proposed a CNN based on 3D convolutions, composed of a set of $3 \times 3 \times 3$ convolutional operations with an incremental number of filters and pooling operations, that they named C3D. Their network, though, requires a lot of data to be trained. Thus, they randomly select a 16-frames snippet from their dataset to enlarge the actual number of samples available. Then, they use their network as feature extractor to actually perform the classification, *e.g.,* using an SVM classifier. Carreira and Zisserman (2017) proposed to overcome this issue by re-using the weight learned on the image classification tasks, by inflating the 2D convolutions into 3D. They applied this idea to Inception V1, obtaining a 3D CNN that they called I3D. Among their findings, however, they show that if a larger amount of data is available, *e.g.,* , for pertaining, the overall results improve.

Such networks, while obtaining *state-of-the-art* results, are computational demanding. Compared to image classification, they often have a number of parameters 2-3 times larger and they require to process tensors that can easily be an order of magnitude larger than the image classification counterpart. Moreover, not every frame is relevant: often the content between two consecutive frames changes slightly in two consecutive frames for most of the frame pairs of the video, while it can change a lot for the salient part of the video. Thus, this extra processing, is in part, wasted.

To overcome this limitation, some recent work in the literature tried to reduce the memory footprint and the number of parameters. Two notable works are R(2+1)D, introduced in Tran et al. (2018, 2019), that proposes to split the 3D convolutions into a 2D convolution shared over time followed by a point-wise 1D convolution over time, reducing the number of parameters and the complexity of the operation needed, and SlowFast, introduced in Feichtenhofer et al. (2019), that proposes to introduce two pathways a slow one and a fast one: the fast one process the video at a high framerate by using convolutions with a small filter size, while the slow one process the video at low framerate but with convolutions with a higher number of filters.

Nevertheless, as we hypothesize the full dynamic of the safety-critical event to be crucial, we need to process the full video and not just a small snippet. Also, in contrast with other domains, acquiring more relevant samples for this task is challenging, as described in Chapter 2, and, thus, the number of samples in the dataset is limited. For these reasons, and following a trend in the accident detection and

anticipation literature, in this thesis we considered 2D CNNs as feature extractors for each frame and aggregate the temporal dimension after.

Accident detection and anticipation

In the context of accident anticipation, Chan et al. (2016) proposed a system for anticipating traffic accidents from dashcam videos. They used an object detection algorithm to extract the objects in the scene and used a pre-trained VGG neural network (Simonyan and Zisserman (2014)) on Places-365 (Zhou et al. (2017)) to extract the appearance features on the detections. Then, they introduced a Dynamic Spatial Attention (DSA) system in combination with a Long short-term memory (LSTM, Hochreiter and Schmidhuber (1997)) network and a custom loss to predict the car crash as early as possible. They evaluated their performance on the novel DAD dataset that, however, is formed mostly of accidents not involving the ego-vehicle. Suzuki et al. (2018) improved the previous architecture by using a Quasi-Recurrent Neural Network (QRNN, Bradbury et al. (2016)) and an adaptive custom loss. They also evaluated their performance on the broader NIDB dataset, which is composed mostly of ego-vehicle accidents. Moreover, they propose a fine-tuning strategy for the appearance feature extractor, with an auxiliary task on the type of subject involved in the scene (*e.g.*, crash with a vehicle, pedestrian or a safe driving scene).

In the context of accident detection, Yao et al. (2019) addressed the problem in an unsupervised way, by training a network to predict the position of objects in the scene in the next frame and by detecting anomalies with respect to the actual position. Taccari et al. (2018) designed a system based on object detection and Random Forest to classify safety-critical events into crashes, near-crashes and safe events.

All the aforementioned approaches heavily rely on object detection to perform the classification or the prediction and do not generalize to events involving only the subject vehicle (*e.g.*, loss of control), where no other vehicle is visible. Nevertheless, these approaches involving object detection as additional input and how they could be extended to tackle the unsafe maneuver classification problem will be discussed in depth Chapter 4.

Recently, some works have investigated the driving attention (*i.e.*, the driver eye fixation) prediction task Palazzi et al. (2018) in the context of safety-critical events, under the hypothesis that such information can provide useful insights for accident detection and prediction (Xia et al. (2018); Fang et al. (2019a,b)). Zhu et al. (2019) leveraged this idea to detect safety-critical events in driving videos, addressing it as an anomaly detection problem. They used the eye fixation salience map to extract anomaly candidates from the full clips and used an architecture based on isolation forests to extract the spatio-temporal safety-critical regions. They also proposed a mechanism based on image segmentation to compute a narrative (*e.g.*, *car hit motor-*

bike or *car hit ego-vehicle*). While this approach has shown potential, it cannot readily be applied at scale as the one we propose, due to the limited availability and adoption of solutions to capture driver attention (Xia et al. (2018); Fang et al. (2019b)).

Driver maneuver detection

In the context of ego-vehicle maneuver detection, Peng et al. (2018) considered both video and GPS/IMU as inputs. Video frames were fed to a pre-trained VGG network on Places-365, while handcrafted features were extracted from the GPS/IMU data. The two streams were then fed to an LSTM model. The authors proposed to process only frames sampled on a uniform spatial basis (*i.e.*, a frame per meter) instead of a temporal one, which they proved to be beneficial for ego-maneuver detection. Their approach doesn't extend to general maneuver detection, though, since maneuvers performed by other vehicles while the subject is not moving are ignored. Zekany et al. (2019) proposed a method to classify subject maneuvers from videos, using a pre-trained model (DeepV2D) to extract depth from video and the camera motion information (and, thus, the trajectory performed by the subject). Then, they leveraged Dynamic Time Warping (DTW, Müller (2007)) distances between trajectories to perform the classification. However, in our case, we're not interested in detecting the subject maneuver alone, but rather in classifying the maneuver with respect to its context. In the context of other vehicle maneuvers detection, Deo *et al.* Deo et al. (2018) designed a framework based on the detection of road scene objects and applied tracking and motion detection.

3.2 Two-stream architecture based on convolutions

The two-stream architecture proposed in this Chapter leverages both the appearance (*i.e.*, the RGB images) and the GPS/IMU information. It is formed by three main modules: a video features extractor, a sensors feature extractor and a classifier combining the two streams. A schematic representation can be found in Figure 3.1.

Video features extractor

The video information is processed using a pretrained ResNet-50 (He et al. (2016)) on the Places-365 dataset (Zhou et al. (2017)), from here on also referred as *backbone*. ResNet is a widely used architecture based on residual connections that has shown superior performances on the Imagenet image classification challenge. Although it has been outperformed in recent years by efficient networks (Tan and Le (2019)) and transformer-based architecture (Zhai et al. (2021)) on the ImageNet challenge, it remains a widely used choice as feature extractor in many computer vision applications.

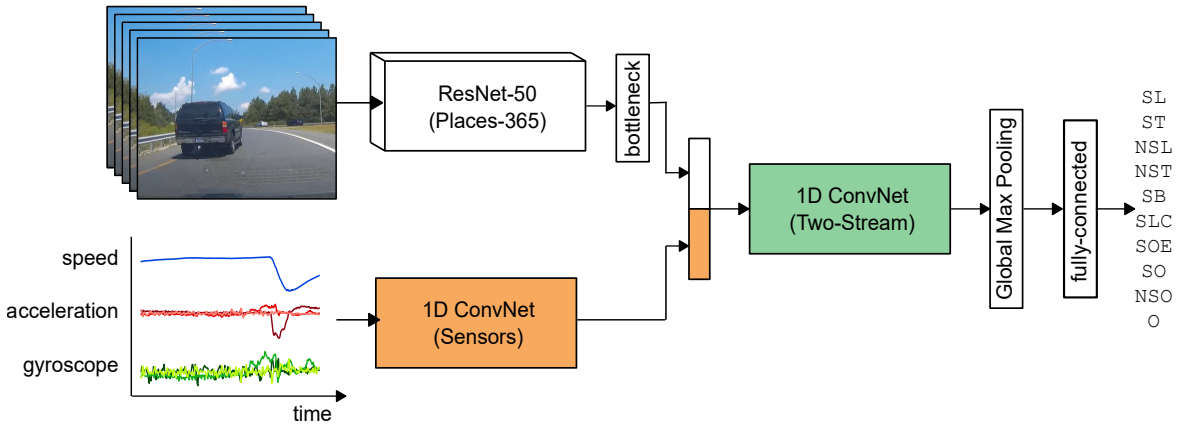


Figure 3.1: Two-stream architecture for unsafe maneuver classification. In white, the video feature extractor, composed by a ResNet-50 backbone pre-trained on Places-365 and a bottleneck layer, applied to each frame of the video, in orange, the sensors feature extractor, a 1D convolutional neural network, and in green the two-stream classifier, again a 1D convolutional neural network.

Places-365 is a dataset that addresses the classification of various types of scenes, with the goal of producing high-level features to be used for visual understanding tasks. It has 365 labels, ranging from indoor scenes, *e.g.*, basement, bathroom, toy shop, to outdoor scenes, *e.g.*, mountain path, desert, soccer field. Among the outdoor classes, there are some that are suited for road scenes, like cross-walk, parking garage, driveway and various road types, and are, thus, a reasonable choice to perform feature extraction from road images. Moreover, similar networks pre-trained on Places-365 have shown good results on tasks closely related to our problem (Suzuki et al. (2018); Peng et al. (2018)).

Formally, each video V is a sequence of frames $\{V_{t_0}, V_{t_1}, \ldots, V_{t_T}\}$ with $\{t_0, t_1, \ldots, t_T\}$ the video frames timestamps, V_{t_i} the 3-channel RGB frame at time t_i of size $W \times H$. Such sequential formulation is converted to a tensor representation in order to be fed to the neural architecture, thus, each video is represented as a tensor of size $T \times 3 \times H \times W$. The backbone is applied to each frame and, as a result, the output is a tensor of size $T \times V_{out}$, which is then reshaped (via transposition) to a size of $V_{out} \times T$, with V_{out} the number of channels of the last convolutional filter of the network. In the case of ResNet-50, $V_{out} = 2048$.

GPS/IMU features extractor

We consider seven type of GPS/IMU measurements: *speed*, *three-axis accelerations* and *three-axis angular velocity*, since they are the most common and broadly available. Such signals have, in general, different sampling frequencies. Moreover, in a general

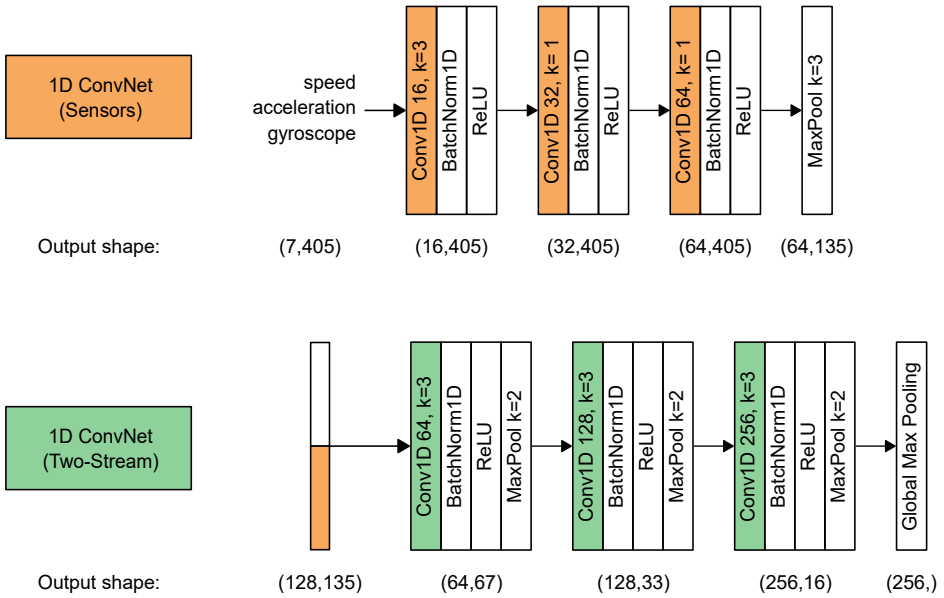


Figure 3.2: Detailed overview of the sensors module (top row) and the two-stream module (bottom row), as an expansion of what presented in Figure 3.1. In the example $\theta = 3$, $f^s = 16$, $N^s = 3$, $f = 64$, $N = 3$ and $B = 64$ with an input of size $T = 135$.

setting, the sensors providing them might not be aligned between each other and with the video frame timestamp.

To cope with these problems, we resample the signals before processing them, so that they have the same number of samples, and this number is a multiple θ of the number of video timestamps. We do not immediately downsample the sensors to the same framerate of the videos, as that might lead to retain as much information as possible from the original signals for processing. Yet, resampling them at a multiple rate of the video framerate makes it easier to temporally align the extracted features with the video features after having processed the sensor streams. Therefore, this module aim is twofold: to extract some high-level representation of the data, similar to what is done for the video stream and to temporally align the video and the sensors information.

Formally, each signal s is a sequence of θT samples $\{s_{\hat{t}_0}, s_{\hat{t}_1}, \ldots, s_{\hat{t}_{\theta T}}\}$ with $\hat{t}_{\theta i} = t_i \; \forall i \in \{1, \ldots, T\}$ and with $s_{\hat{t}_i} \in \mathbb{R}^7$. Similarly to what we propose for the video stream, we represent each signal as a tensor of size $7 \times \theta T$ and we feed this tensor to a 1D convolutional neural network formed by several stacked 1D convolution filters. The network applies N^s convolutional operations. First, a convolution with kernel size θ and with f^s filters is applied, followed by $N^s - 1$ 1D convolutions with kernel size 1 and with twice the filters of the previous layer. Each convolution is followed by a 1D batch normalization and a ReLU activation function. Finally, a max-pooling layer of size θ is applied, which temporally aligns the video and the sensors streams

as required.

The intuition behind the usage of the point-wise operations is that we want the network to extract local features. Ideally, each frame should be aligned with a feature vector representative of the sensor data between the previous one and the following one and, thus, retain its temporal semantic meaning. The fusion between different frames and sensor data over time will be performed lately in the stage. A schematic representation of the described module can be found in Figure 3.2. The output is a tensor of size $s_{out} \times T$ with $s_{out} = f^s \cdot 2^{N^s-1}$.

Two-stream classifier

This module first combines the outputs of the video and sensors feature extractors, *i.e.*, two tensors of size $V_{out} \times T$ and $s_{out} \times T$ respectively, by concatenation on the temporal dimension. However, simple concatenation may not be the best strategy to combine features as typically $V_{out} \gg s_{out}$. Further processing them so that V_{out} and s_{out} have comparable size may help in correctly leveraging the GPS/IMU information at the classifier stage.

Indeed, we observed experimentally that applying a *bottleneck layer* to the video features improves the overall performance. This layer is formed by a fully-connected layer of size B, followed by a 1D batch normalization and a ReLU activation function. Thus, the resulting concatenation is a tensor of size $(B + s_{out}) \times T$.

Such tensor is then fed to a 1D convolutional network, formed by N stacked 1D filters. Each operation is formed by a 1D convolution with kernel size 3, a 1D batch normalization and a max-pooling of size 2 and stride 2. The number of filters applied in each layer is doubled with respect to the previous one, with the first convolution having f filters, while the temporal span of the data is halved. In this way, the network is forced to learn higher-level representations of the underlying data.

The output of the aforementioned filters is a tensor of size $\left(f \cdot 2^{N-1}\right) \times T'$, where T' is the temporal span after all the max-pooling layers, which is fed to a 1D Global Max Pooling layer. In our tests, such layer performed better than the commonly used Global Average Pooling layer. One possible reason is that the unsafe maneuver will occur only in a few, or even a single, temporal sample among the processed T' and, thus, the network performs better if it bases its classification only on it, without taking into account features related to safe driving. Finally, a fully-connected layer of size 10, as the number of unsafe maneuvers considered, is applied, with a *softmax* activation function to perform the classification.

A schematic representation of the described module can be found in Figure 3.2. It is worth noticing that the proposed architecture can be used on data streams with arbitrary resolutions and number of frames, since it is formed mainly by convolutions and spatial or temporal pooling layers. The only fully connected layers are the

final classifier and the bottleneck, which however do not require a fixed input size as they act after global pooling operations.

It is important to highlight that the architecture was designed to be agnostic to the moment in time in which the safety-critical event occurs. In particular, as anticipated in Chapter 1 and Chapter 2, the data relative to a safety-critical driving event are acquired by recording a fixed number of seconds around a detected peak in acceleration (kinematic trigger). Thus, generally, such data are biased, as the event always occur in a given temporal segment. The only use of convolutional operations along the temporal dimension, combined with the usage of a pooling, and, thus, position agnostic, operation prevents the network from being affected by such bias and allows the generalization beyond the events used during the model training.

3.3 Experimental results

All the experiments are conducted on the SHRP2 NDS dataset leveraging the unsafe maneuver annotations, described in Chapter 2. In addition, we consider and annotate the frame in which the event begins and end. The latter two annotations are made with the following rule: it is marked as *eventStart* the frame in which the first maneuver that starts the sequence of events leading to the dangerous situation occurs; it is marked as *eventEndt* the frame in which the last evasive maneuver or action occur, that concludes the sequence of events that are part of the dangerous situation.

The dataset is composed of videos at 15 fps with resolution 480×356, while the GPS-related sensors have a sampling frequency of 1 Hz and the IMU of 10 Hz. The videos have variable length, going from a minimum of 150 to a maximum of 692 frames, however the vast majority are 30 seconds videos of 450 frames. To uniformize the dataset, we cap all the videos to 450 frames length by removing the initial frames and we align the GPS/IMU information and the event start and end to the crops. Then, as the dataset was acquired in a naturalistic environment, presents samples with missing or miscalibrated data. To cope with the first one, we discard every video with missing speed or accelerometer data, while considering a constant zero signal in case of missing gyro data. This choice is motivated by the fact that the first two are a small minority, while the third one is a phenomenon occurring way more frequently, in roughly 30% of the samples. Finally, we remove the corrupted or occluded videos, ending up with a dataset of 8497 events, that we stratified split into train, validation and test with a 80/10/10% proportion. The splits, discard flag and event start and end data are released along with the unsafe maneuver annotations, for reproducibility purposes.

Since the backbone model has been trained on images with shape 224×224, we adjusted our input frames accordingly, maintaining the aspect ratio and having

the smaller side of 224 pixels. However, to be able to perform data augmentation, we resized every video to 346×256 and picked a random 314×224 crop during training. At inference time, we only considered the central crop.

As for the GPS/IMU data, as mentioned above, the accelerometer and the gyroscope occasionally present miscalibration artifacts. To cope with them, we first re-scale such data to have zero mean in each example. Then, we use a robust scaling strategy, scaling the $25th$ and $75th$ percentiles of each sensor in the range $[-0.5, 0.5]$. This approach has empirically shown better results than the more common $[0, 1]$ scaling. Intuitively, these sensors presents relatively small variations in terms of magnitude during regular driving, and severe peaks in correspondence of a hard braking or a crash. Thus, by scaling the peaks, the content of the regular driving might get lost. In contrast, the normalization strategy we used retains the dynamic of the full driving segment while normalizing the data. Furthermore, Gaussian Butterworth Noise was applied, only during training, as data augmentation.

All tests are conducted minimizing the cross-entropy loss with class weight, to cope with class unbalance, and with Adam optimizer (Kingma and Ba (2014)) with an initial learning rate $lr = 10^{-3}$, reduced to $lr = 10^{-4}$ after 30 epochs and to $lr = 10^{-5}$ after 40 epochs. Moreover, we use a weight decay $wd = 5 \cdot 10^{-3}$. The training process takes 20 minutes on a V100 GPU by pre-computing and storing locally the backbone outputs.

Finally, as evaluation metric we use the mean average precision (mAP), which is equivalent to computing the mean area under the precision-recall curve for each class and, thus, takes into accounts both precision and recall and is robust to class unbalance. We set up two sets of tests. First, we evaluate the architecture on small video footage around the event, showing how the performance of the model changes with the different architecture parameters and the relative benefits of the various choices made in its definition. Second, we test our architecture on the full videos, that contain a good portion of safe driving before the safety-critical event, showing the capability of the proposed architecture to focus on the safety-critical part of the clip. The two setups are schematically reported in Figure 3.3

Experiments: small clip around the event

The first set of experiments are conducted on a small clip containing only the safety-critical event, in order to prove the capability to distinguish different types of unsafe maneuvers of the proposed architecture. To do this, we consider the $2/3$ of the $[eventStart, eventEnd]$ segment as event reference point, and the clip that goes from σ_{start} frames before to σ_{end} frames after such point. We empirically found that $\sigma_{start} = 90$ (6 seconds) and $\sigma_{end} = 45$ (3 seconds) let us exclude most of the footage of safe driving, while retaining the entire relevant maneuver. The motivation behind the choice of the reference point is relative to the observation that the maneuvers

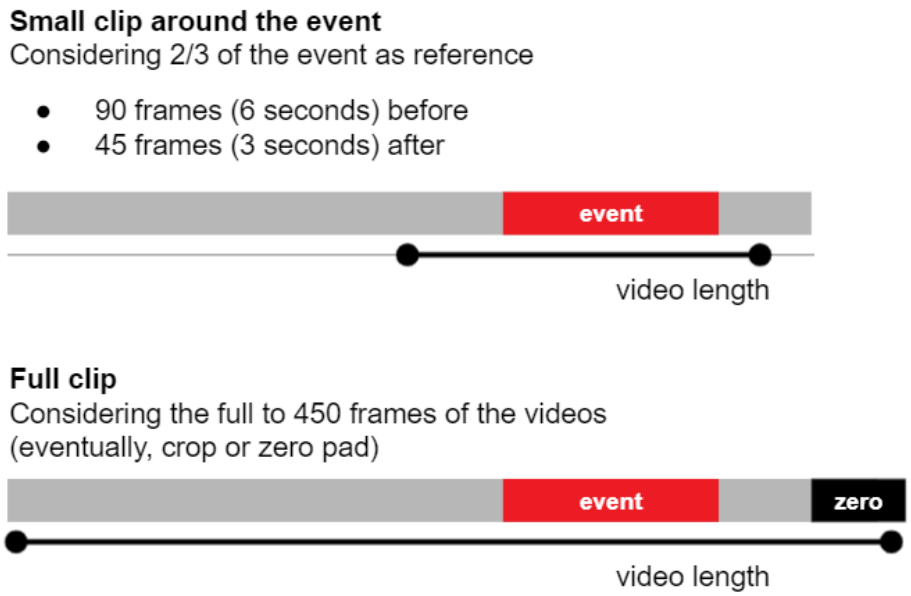


Figure 3.3: The two experimental setups we considered. On the top, the small clip around the event setup, that considers 135 frames around the dangerous event. On the bottom, the full clip setup, that considers the full 450 frames of the video, zero-padding the shorter samples or removing the initial frames of the longest ones.

preceding (and leading to) the dangerous event have generally a wider time span than the evasive and following ones.

Table 3.1: Hyperparameters search space

Variable	Parameters
θ	$\{1,3\}$
N^s	$\{1,2,3\}$
f^s	$\{16,32,64\}$
B	$\{32,64,128\}$
N	$\{3,4,5,6\}$
f	$\{32,64,128,256\}$

The proposed architecture, as described in Section 3.2, has many hyperparameters to be tuned. Specifically, θ, N^s, f^s, N, f and B. Intuitively, we expect joint dependencies between the parameters in their effect on the network performance. For instance, according to the best practice of gently increasing the number of filters while decreasing the dimension of the data, a larger B might require a larger f. Fine-tuning the parameters one by one might thus lead to a suboptimal solution. On the other hand, testing out all possible combinations is a demanding job in terms

of resources (time and computational power) required. For this reason, we decided to use a Random Search (Bergstra and Bengio (2012)) strategy on the parameters space, optimizing the validation *mAP*. Parameters and the range use during Random Search are reported in Table 3.1. We run a total of 60 experiments and found the best results with the following setting: $\theta = 3$, $f^s = 32$, $N^s = 2$, $B = 32$, $f = 32$ and $N = 4$, with a *mAP* on the validation set of 0.653 and a *mAP* on the test set of 0.635.

Table 3.2: Confusion matrix for the best configuration on small clip around event settings.

		Predicted maneuvers										**total**	**recall**
		SL	ST	NSL	NST	SB	SOE	SLC	SO	NSO	O		
True maneuvers	SL	**12**	1	8	0	4	1	1	0	2	1	30	.40
	ST	1	**15**	0	4	0	2	0	2	0	1	25	.60
	NSL	11	2	**84**	22	20	0	0	1	3	6	149	.56
	NST	2	8	6	**52**	5	2	1	6	1	14	97	.54
	SB	11	1	30	7	**239**	3	2	3	10	5	311	.77
	SOE	2	2	0	3	0	**86**	6	1	0	7	107	.80
	SLC	0	1	0	1	0	3	**12**	0	2	0	19	.63
	SO	0	1	0	1	0	2	0	**13**	0	0	17	.76
	NSO	0	0	1	1	2	1	0	1	**13**	0	19	.68
	O	5	12	5	4	1	0	2	3	1	**41**	74	.55
precision		.27	.35	.63	.55	.88	.86	.50	.43	.41	.55	848	-

The confusion matrix on the test set is reported in Table 3.2. Thanks to the presence of the class weight in the training loss function, all the classes, including the minority ones, have overall reasonable precision, recall and AP, with two notable exceptions. The SL class tends to be confused with NSL and SB. We believe that the first case is due to a natural ambiguity of the two classes, *e.g.*, in a narrow road, maybe without a lane separation line, it is sometimes hard to say whether the subject invaded the other lane or the opposite, while the second one is mostly due to the classifier focusing on the wrong vehicle to detect the safety-critical event. The ST class tends to be confused with NST and 0. The first case is composed by events in which both vehicles are turning, into the same or opposite direction and, thus, could be disambiguated only considering the road laws. In the second one, instead, the network seems not to take into consideration the pedestrian or pedal-cyclist in the scene. This phenomenon might be influenced by adverse lighting conditions.

To prove the effectiveness of the proposed approach, we compared the model described above, which include the video stream and the sensors one and that is referred as VS, with several variants:

- VS+L: A baseline classifier, working on the same features, but deploying a 2-Stacked LSTM layer, as proposed in related works Chan et al. (2016); Peng et al. (2018).
- VS-B: The proposed architecture, without the the bottleneck layer.
- V: The proposed architecture, without the GPS/IMU stream.
- S: The proposed architecture, without the video stream.

Results are reported in Table 3.3. Clearly, processing the two streams is key to achieve high performance, showing that the two sources of information complement each other in the solution to our problem. In this sense, while the video stream generally performs worse than the sensor one, it is better at detecting maneuvers involving other vehicles (*e.g.*, NST, O) while the opposite is true for the ego-vehicle ones (*e.g.*, SL, ST, SOE, SLC). Furthermore, the introduction of the bottleneck layer is beneficial for the classification, validating our hypothesis that to align the dimension of the different inputs before performing the sensor fusion is beneficial and prevents an input to prevaricate on the other one. Finally, even when deploying both feature streams, there is a large gap in mAP between the proposed classifier based on convolutions and pooling layers and the LSTM layers typically used in the state-of-the-art.

Table 3.3: Results of the proposed approach and baselines

model	**average precision (AP)**										**mAP**
	SL	ST	NSL	NST	SB	SOE	SLC	SO	NSO	O	
VS	.28	.54	.61	.60	.91	.92	.60	**.68**	.62	.59	**.635**
VS-B	.20	.48	**.66**	**.61**	**.93**	.91	.55	.57	**.70**	**.66**	.627
VS+L	.17	.46	.57	.47	.89	.90	**.64**	.67	.31	.62	.569
S	**.35**	**.55**	.57	.30	.80	**.94**	.59	.65	.59	.23	.556
V	.10	.27	.51	.42	.84	.59	.51	.31	.07	.51	.414

Experiments: full video

As a second set of experiments, we applied the same architecture to the full 450 frames (30 seconds) videos, to prove the ability of the model to focus on the safety-critical event among several other safe maneuvers. Both streams of the videos shorter than such dimension were zero-padded. We retained the optimal architecture parameters from the previous experiments and the same *train*/*validation*/*test* split, but the architecture was retrained from scratch. As a result, we obtained a validation *mAP* of 0.648 and a test *mAP* of 0.634. The confusion matrix obtained on the test set

Table 3.4: Confusion matrix with full video as samples

		Predicted maneuvers										**total**	**recall**
		SL	ST	NSL	NST	SB	SOE	SLC	SO	NSO	O		
True maneuvers	SL	**9**	0	9	2	7	2	0	0	0	1	30	.30
	ST	1	**12**	0	6	0	2	0	1	0	3	25	.48
	NSL	8	1	**95**	23	17	1	0	0	0	4	149	.64
	NST	0	7	9	**64**	3	1	1	3	2	7	97	.66
	SB	14	2	25	15	**241**	3	0	1	4	6	311	.77
	SOE	0	1	0	3	0	**91**	4	1	0	7	107	.85
	SLC	0	0	1	0	1	3	**13**	0	1	0	19	.68
	SO	0	1	0	3	0	2	0	**9**	1	1	17	.53
	NSO	2	0	2	2	1	2	0	0	**10**	0	19	.53
	O	3	8	6	6	2	3	3	2	2	**39**	74	.53
precision		.24	.38	.65	.52	.89	.83	.62	.53	.50	.57	848	-

is reported in Table 3.4 Such results are comparable with the ones obtained with the clip around the event setup, showing the capability of the proposed architecture to focus on the relevant safety-critical part of the video, an important trait in a practical deployment of our solution.

3.4 Backbone fine-tuning

While using a pre-trained backbone on Places-365 as feature extractor showed good results in our experiments and in similar tasks in the relevant literature (Peng et al. (2018); Chan et al. (2016)), to fine-tune the backbone would almost certainly improve the performance. However, when working with videos, it is hard to fine-tune the backbone directly on the task, especially if the videos are long or have a high *fps*, as it generally requires larger datasets and more GPU memory than available. Thus, many works propose to fine-tune their backbone on auxiliary tasks that are close to the original one.

Suzuki et al. (2018) propose to use a per-frame classification task on four classes, indicating whether in the frame there is an imminent collision with a pedestrian, a cyclist, a vehicle or none of the above (a frame from the safe driving). Kim et al. (2018) propose to train a small convolutional network that predicts the vehicle steering and acceleration. We propose to fine-tune the backbone on the same unsafe maneuver classification task, but by considering solely a smaller version of the video as input, with a lower frame-rate and duration.

We considered video segments of 32 frames at 5 *fps*, *i.e.*, 6.4 seconds, randomly choosing such segments during the training phase under the constraint that at least

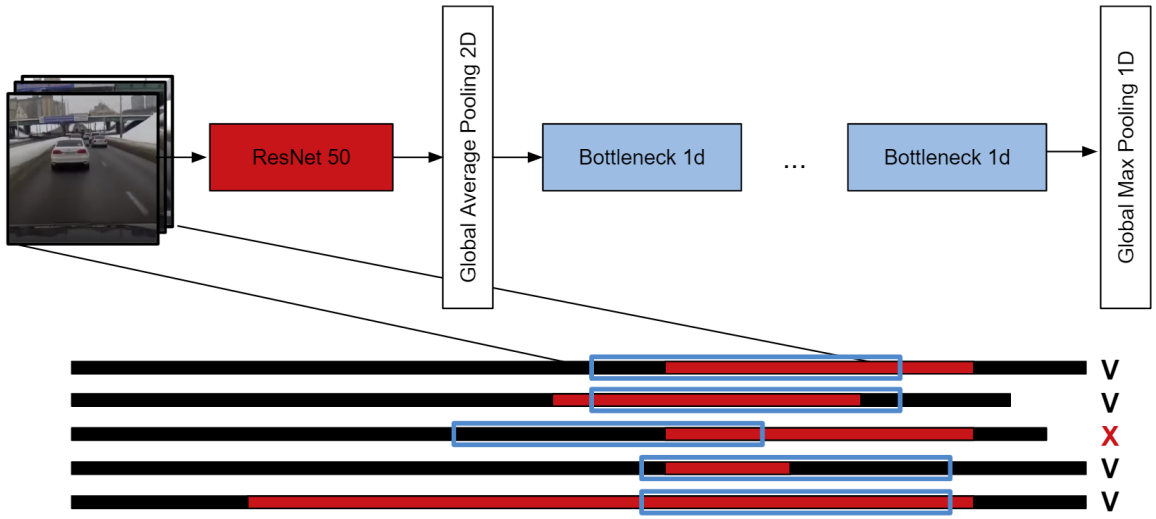


Figure 3.4: The architecture used to fine-tune the backbone. Below, a set of examples used for fine-tuning, satisfying (V) or not satisfying (X) the constraint. In blue, the selected segment, in red, the safety-critical event, in black the full video.

75% of the safety-critical event should be contained in the segment or that all the frames should be safety-critical event frames. This approach was possible only by knowing the beginning and the end of the safety-critical event in each video. Also, the choice of lowering the frame-rate was made to consider a wider temporal footprint that should contain the full evolution of the event, as by considering 32 frames at 15 *fps*, *i.e.*, 2.1 seconds, only a portion of the event might have been fed into the network, for instance, the beginning of the event or the evasive maneuvers, making the classification task difficult or impossible. During the validation and test phase, we are considering all segments satisfying the above conditions for each segment, with a minimum overlap of 50%.

The architecture used to fine-tune the backbone is reported in Figure 3.4. We first fed the video to the backbone and applied a 2D global average pooling layer. After these operations, the output tensor has shape $F \times C$, with $F = 32$ numbers of frames and C backbone output channels. In our specific case, we considered ResNet-50, having $C = 2048$. Then, in line with the residual structure of the backbone, we considered $N = 4$ 1D adaptations of the Bottleneck layers proposed in He et al. (2016) of size C, each of which applies a point-wise 1D convolution with $C/4 = 512$ filters, a 1D convolution of size 3 and $C/4 = 512$ filters and point-wise 1D convolution with $C = 2048$ filters, warping everything with a residual connection between the input of the three convolutions and the output. In addition to this, we would want to reduce gradually the temporal dimension. Thus, the first point-wise operation of each block has a stride $s = 2$ and each residual connection is equipped with an additional point-wise operation, to adjust the number of channels accordingly. Finally,

Table 3.5: Confusion matrix on the unsafe maneuver auxiliary task for backbone fine-tuning, with an overall *mAP* of 0.424

		Predicted maneuvers										**total**	**recall**
		SL	ST	NSL	NST	SB	SOE	SLC	SO	NSO	O		
True maneuvers	SL	**11**	0	42	0	19	1	4	5	11	5	98	.11
	ST	4	**30**	3	8	0	11	3	14	2	3	78	.38
	NSL	24	4	**390**	12	34	7	3	1	17	18	510	.76
	NST	13	26	80	**71**	26	22	8	12	5	47	310	.23
	SB	49	4	96	3	**607**	3	2	4	133	15	916	.66
	SOE	9	43	14	14	6	**105**	42	23	6	60	322	.33
	SLC	0	3	1	5	5	4	**26**	6	2	2	54	.48
	SO	0	11	2	5	0	0	1	**28**	2	1	50	.56
	NSO	1	0	7	4	14	2	3	4	**14**	4	53	.26
	O	7	17	16	13	11	17	6	14	5	**139**	245	.57
precision		.09	.22	.60	.53	.84	.61	.27	.25	.07	.47	2636	-

a 1D global max-pooling layer is applied over the reduced temporal dimension and the resulting vector is used for classification. The weights of the backbone are initialized with the Places-365 weights, while the residual bottlenecks are trained from scratch.

We train the network minimizing the cross-entropy loss with class weight and Adam optimizer, with a fixed learning rate of $lr = 10^{-4}$. In addition to randomly selecting the temporal segment, as described above, we are using Random Crop, *i.e.*, we randomly select a 314×224 crop of the video, and Color Jittering, *i.e.*, randomly changing the video brightness, contrast and saturation. Moreover, as the first layers of the networks learns to detect general-purpose low-level descriptors (*e.g.*, the presences of edges and color patches) while the one in-depth aggregates such descriptor into high-level ones (*e.g.*, the presence of a car) and given the relatively small size of the samples available for fine-tuning, we choose to not train (*i.e.*, keep frozen) the first 22 layers of the backbone.

The results of the training on the auxiliary task are reported in Table 3.5. We obtained a *mAP* of 0.424. Although the number of samples is different, as we are considering multiple segments for each video as described above, the results are comparable and, for some classes, outperforming the one obtained by the video only backbone in Table 3.3 in terms of precision and recall, while being obtained on segments with a significantly shorter time span and frame-rate. Plugging the fine-tuned backbone into the two-stream architecture proposed in Section 3.2, we obtain a *mAP* of 0.690 of, with a boost of roughly 0.06 points in the small clip setup. In Table 3.6 we report the resulting confusion matrix..

Table 3.6: Confusion matrix of the two-stream architecture with fine-tuned backbone, with an overall *mAP* of 0.690

		Predicted maneuvers										**total**	**recall**
		SL	ST	NSL	NST	SB	SOE	SLC	SO	NSO	O		
True maneuvers	SL	**15**	1	9	1	2	2	0	0	0	0	30	.50
	ST	0	**16**	0	5	1	2	0	0	0	1	25	.64
	NSL	6	0	**122**	10	2	1	1	0	0	7	149	.82
	NST	1	3	7	**59**	7	1	0	5	2	12	97	.61
	SB	15	4	19	7	**255**	1	0	0	7	3	311	.82
	SOE	0	2	1	1	0	**87**	6	1	2	7	107	.81
	SLC	0	1	0	1	2	1	**12**	1	0	1	19	.63
	SO	0	0	0	2	0	2	0	**12**	1	0	17	.71
	NSO	0	0	1	1	2	0	0	2	**12**	1	19	.63
	O	0	8	5	4	3	3	1	1	2	**47**	74	.64
precision		.41	.46	.74	.65	.93	.87	.60	.55	.46	.59	848	-

3.5 Conclusions

In this Chapter, we have introduced the novel unsafe maneuver categorization problem and have shown how a novel neural architecture, processing both videos captured by a dashcam as well as GPS/IMU sensor data, can be used to tackle it. After a broad calibration phase, exhaustive tests have shown the capability of the proposed architecture to distinguish the various unsafe maneuver types and also to correctly identify the time interval in which an unsafe event occurred within a recording mostly composed of normal behavior. Also, we proposed a fine-tuning strategy for the backbone CNN based on a simplified version of the task, that has shown a significant boost in the overall performance.

Chapter 4

Unsafe maneuver classification using Spatio-Temporal Attention Selector †

We propose a novel deep learning architecture to classify unsafe driving maneuvers from dashcam and GPS/IMU data. Such architecture processes the output of an object detection algorithm in combination with raw video frames and GPS/IMU data. At the core of the architecture there is a novel Spatio-Temporal Attention Selector (STAS) module, which (1) extracts features describing the evolution of each object in the scene over time and (2) leverages multi-head dot product attention to select the relevant ones, i.e., the dangerous ones or the ones in danger, to perform classification. Our method is shown to achieve higher performance than other approaches in the literature applying attention over single frames.

In this Chapter, we propose an alternative approach to the one presented in Chapter 3 to tackle the unsafe maneuver classification problem by proposing two major improvements: first, we integrate the output of an object detection algorithm in the pipeline, to provide the network with explicit information about the entities on the road and let it learn high-level representations of the interactions between them. Second, we leverage attention Bahdanau et al. (2014); Vaswani et al. (2017) to let the network focus on the relevant objects in the scene, *i.e.*, the one involved in the unsafe maneuver, and on the relevant temporal segments.

† Part of the content of this chapter was published in:

- M. Simoncini, D. Coimbra de Andrade, L. Taccari, S. Salti, L. Kubin, F. Schoen, F. Sambo, "Unsafe maneuver classification from dashcam video and GPS/IMU sensors using Spatio-Temporal Attention Selector", *IEEE Transactions on Intelligent Transportation Systems*, 2022.

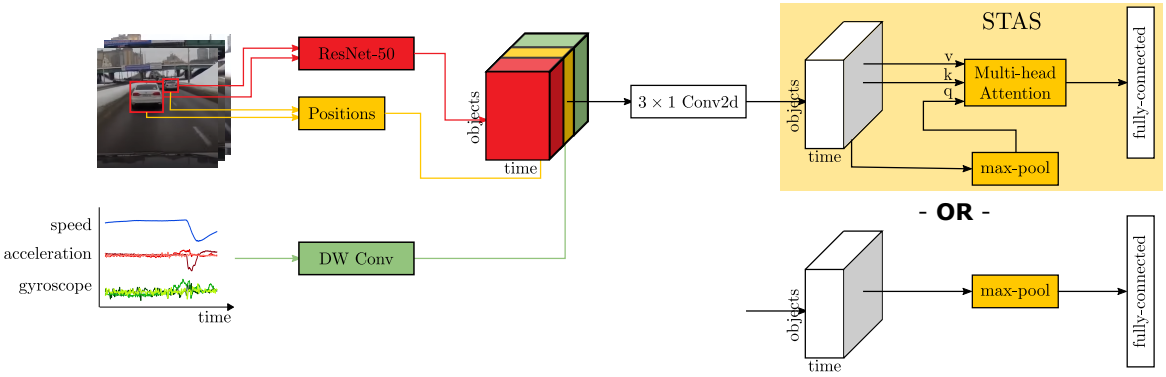


Figure 4.1: An overview of the proposed architecture. In red, the appearance features extracted from the object detector output, in yellow the features relative to the boxes positions, in green the GPS/IMU features, in white the per-object feature extracted. Such features are either pooled with a max pooling layer or fed to the STAS module, composed of a max pooling layer as query vector and multi-head attention.

While the usage of attention is motivated by the extremely good results it has obtained in various fields of the scientific literature (de Andrade et al. (2018); Chan et al. (2016); Bahdanau et al. (2014); Vaswani et al. (2017); Suzuki et al. (2018); Xu et al. (2015); Anderson et al. (2018a); Herdade et al. (2019a); Cultrera et al. (2020); Woo et al. (2018); Hu et al. (2018)), it also has an interesting by-product: attention forces the network to *focus* on a particular portion of the input and to have the resulting output that mainly depends on that portion. Thus, it provides an explanation on the reason behind a given prediction by learning features with an associated semantic meaning, as reported in Gunning (2017), and that allow the network to intrinsically explain itself (Gilpin et al. (2018)). This aspect in particular is known in the scientific literature as eXplainable Artificial Intelligent (XAI) (Abdul et al. (2018); Gunning (2017); Gilpin et al. (2018)) and it is of crucial importance when the output of the model is presented to or evaluated by a human (in the so-called human-agent systems), where a motivation behind the prediction might be necessary to convince of the correctness of the prediction or to understand the reasons behind a misclassification (Rosenfeld and Richardson (2019)).

In the road-safety domain, and, specifically, to address the accident anticipation problem, previous works in the literature attempted to integrate attention with the output of an object detection algorithm, using Dynamic Spatial Attention (DSA), in particular Chan et al. (2016) and Suzuki et al. (2018). Such module applies attention to all the detected objects of a single frame, in order to have the network focus on the most relevant object at a given moment in time, and then uses a recurrent module to extract the temporal dependencies between the learned features. In contrast, our aim is to use attention to select the most relevant object and the most relevant temporal segment to correctly classify the maneuver, by explicitly extract-

ing features describing the evolution of each object in a given set of frames and then selecting the most relevant one to perform the prediction. We refer to this approach as Spatio-Temporal Attention Selector (STAS).

Moreover, and as an alternative to the use of the attention mechanism, we propose to leverage a max-pooling layer to extract relevant information from the object features. While such layer is more opaque than attention, *i.e.*, it is harder to understand the reasons behind a prediction, it showed slightly superior performance compared to the attention-based ones in our experiments, suggesting that model explainability comes with a cost. An overview of the proposed architecture is reported in Figure 4.1.

4.1 Related works

In this section we discuss the most relevant papers in the literature, focusing on closely related tasks using object detection and the usage of the attention mechanism in the computer vision literature.

Attention mechanism

The attention mechanism was first introduced in Bahdanau et al. (2014) in the Neural Machine Translation (NMT) literature, with the aim of giving the decoder the ability to dynamically *focus* on parts of the input sentence that are relevant to predict a target word, instead of being forced to encode the source sentence into a fixed-length vector. This is achieved by projecting each word in an embedding space and computing a similarity measure between words, represented as a fully-connected operation. Vaswani et al. (2017) generalized the mechanism proposed in Bahdanau et al. (2014) (also referred to as *soft attention*) by computing the similarity measure using the dot product operation, introducing what is referred as *dot-product attention*, and by computing a set of embedding and attention operations in parallel instead of a single one, that get then merged together into a single final output, that goes under the name of *multi-head dot product attention*.

Later on, the idea of letting the network focus on a specific part of the data was applied in other fields of research, *e.g.*, computer vision, mainly in two ways. Xu et al. (2015) proposed to use the attention mechanism in an encoder-decoder architecture for video captioning, having the network focus on features extracted from a specific part of the image. Other works generalized the approach by considering the features from the output of an object detector Anderson et al. (2018a); Herdade et al. (2019a). Xu et al. (2015) proposed to use the attention mechanism in an encoder-decoder architecture for video captioning, having the network focus on features extracted from

a specific part of the image. Other works generalized the approach by considering the features from the output of an object detector Anderson et al. (2018a); Herdade et al. (2019a).

More recently, the attention mechanism has been used to increase network representation capability of over singe-sequence data, to focus on important features and suppressing unnecessary ones as well as learn features that relates different positions of the sequence. This approach generally goes under the name of *self-attention*, and was firstly introduced in Vaswani et al. (2017). Woo et al. (2018) applied self-attention to images, introducing the Convolutional Block Attention Module (CBAM). Such block is a composition of modules, a channel attention module and a spatial attention one. Starting from a 2D feature tensor, the first module uses a combination of max-pooling and average-pooling operations over the spatial dimension, to get a descriptor that is applied back to the input tensor via element wise multiplication; the second one performs a similar operation over the channel dimension. Similarly, Hu et al. (2018) proposed the Squeeze-and-Excitation (SE) block, that computes a pooling operation over the spatial dimension (squeeze), applies a channel-wise feed-forward operation to learn inter-channel dependencies and uses the output to scale the input tensor (excitation). Finally, attention weights have been also used to achieve model explainability Xu et al. (2015); Kim et al. (2018). As the attention output is a weighted sum of the input (values)

Accident anticipation

The usage of attention for accident anticipation from dashcam videos was first proposed in Chan et al. (2016). They used an object detection algorithm to extract the objects in the scene and computed the features of a pre-trained convolutional neural network on Places-365 Zhou et al. (2017) on their locations. Then, they introduced the DSA system in combination with an LSTM and a custom loss to predict the car crash as early as possible. The DSA system proposes to apply soft attention to features extracted from the object in the scene on each frame, in order to build dynamically at each frame a feature vector representative of the relevant objects. Performance was evaluated on the novel DAD dataset that, however, is mostly composed of accidents not involving the ego-vehicle. Suzuki et al. (2018) improved the previous architecture by using Quasi-Recurrent Neural Network (QRNN) and an adaptive custom loss. Also, they used a fine-tuned backbone on per-frame risk factor classification and background classification task. They evaluated their performance on the broader NIDB dataset, which is mostly composed of ego-vehicle accidents.

4.2 Object-aware feature extraction for unsafe maneuver classification

In order to address the unsafe maneuver classification task, we propose an architecture that leverages the video information and the GPS/IMU data. We first use an object detector to extract the position and types of all the objects from each frame and extract appearance and positional features for each object. Then, the object features are enriched with features extracted from the GPS/IMU data. We propose a novel feature extraction method that leverages depth-wise separable convolutions (Chollet (2017); Howard et al. (2017)) to preserve each sensor semantic meaning. Then, use a tracking algorithm based on the detection class, confidence and positions to link the same real object in different frames. Finally, we apply a set of convolutional operations to each object, in order to extract high-level descriptors for each of them and to reduce the temporal dimensionality of the data.

Object detection and feature pooling

We use a Faster-RCNN (Ren et al. (2015)) with ResNet-101 (He et al. (2016)) backbone as object detector on each frame, and extracted object positions, classes and detection confidence. Faster-RCNN is a popular object detector that has shown good results in terms of mAP while retaining an acceptable inference time.

At this point, we would like to associate an appearance feature vector to each object. One could consider the output of the detector backbone or consider other backbones, that could be pre-trained on other tasks, as in Chan et al. (2016); Peng et al. (2018) and as shown in Chapter 3, or fine-tuned on an auxiliary task, as in Suzuki et al. (2018) and in Chapter 3. In this second case, it is common to extract the crop of the image for each object and run it through the backbone (Chan et al. (2016); Suzuki et al. (2018)). This might result, however, in prohibitive inference times. For instance, to compute the features of a single frame with ten objects detected, one would need to compute ten backbone forward passes. During training, it is possible to speed up the process by storing locally the backbone outputs. However, when there are a lot of detections for each frame, this might also results in high storage costs. In contrast,inspired by Cultrera et al. (2020) and as widely shown effective in the Computer Vision community, we are using a RoI pooling layer, firstly introduced in Girshick (2015), that allow us to compute the object features with a single backbone forward pass both during training and inference.

The RoI pooling layer starts from the full frame feature, *i.e.*, a tensor of size $C \times H \times W$ obtained by applying the backbone on a given frame, and a list of regions of interest (in our case the detected objects), *i.e.*, a set of tuples (x, y, w, h) indicating respectively the top left corner coordinates and the box width and height. It converts

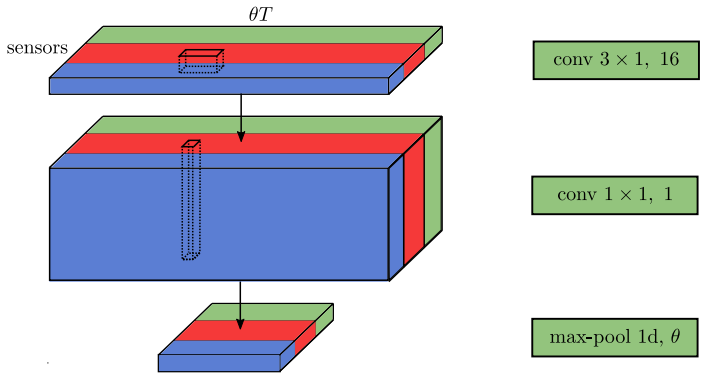


Figure 4.2: Schematic representation of the GPS/IMU module leveraging depthwise separable convolution applied to each sensor. Different colors represent different sensors.

the features inside any region of interest into another feature map with a fixed spatial extent of $p_h \times p_w$. This is done by first identifying the RoI region on the full frame feature map, dividing such region into a $p_h \times p_w$ grid of sub-regions of approximate sizes of $h/p_h \times w/p_w$ and then max-pooling the values independently on each feature and on each cell. We consider $p_h = p_w = 7$. The whole operation requires a single backbone evaluation and, thus, is efficient in terms of inference time and does not require feature storage. Furthermore, it allows the usage of image augmentation techniques.

GPS/IMU module

The data coming from the GPS/IMU sensors is used as input to the GPS/IMU module, as shown in Figure 4.1. Such data are pre-processed as described in Chapter 3. We first resample the signals via interpolation, so that they have the same number of samples, and this number is a multiple $\theta = 3$ of the number of video timestamps. Then, we apply a set of convolutional operations and, finally, we use a max pooling operation of size θ, to align the sensors and the video stream.

Instead of using 1D convolutions over the temporal dimension, though, in this Section we propose to apply the same convolutional operation to process each signal independently, with the idea to learn filters to be applied to a generic signal and to extract features describing its temporal evolution, preserving the individual signal semantic meaning. To accomplish this, we use 2D depthwise separable convolutions (Chollet (2017); Howard et al. (2017)).

Regular 2D convolutions attempts to learn filters in a 3D space, having a 2D spatial extension over the input, namely width an height, and a channel dimension. Thus, a single convolution is responsible to both learn correlations across channels and spatial correlation. Depthwise convolutions propose to split this two aspect by

performing two convolutional operations: fist, a convolution over the spatial dimension performed independently on each channel, in order to learn spatial correlation; then, a pointwise, *i.e.*, 1×1, operation that mixes the learnt spatial dependencies over different channels. In this way, depthwise separable convolutions greatly reduce the computational and model size due to the factorization of the operations, that leads to a regularization effect.

In order to apply this idea to sensors data, we follow what has been proposed in the audio processing literature (de Andrade et al. (2018); Zhang et al. (2017)). Starting from an input tensor of shape $\theta T \times s$, with T number of frames and s total number of sensor signals, we fist add an extra dimension, changing the shape of the input tensor to $1 \times \theta T \times s$, with the first dimension representing the number of channels. Then, we apply a 2D convolution with kernel size $k = 3 \times 1$ and with $f^s = 16$ output channels (*i.e.*, filters). The output tensor, using padding over the temporal dimension to maintain the same spatial extent, has shape $f^s \times \theta T \times s$. Then, we apply a second 2D convolution with kernel size $k = 1 \times 1$ and with 1 output channel, that get then removed to go back to a tensor of shape $\theta T \times s$. While one could stack multiple of these operations, to learn richer information as in Chollet (2017) and Howard et al. (2017), we choose to use a single pair of convolutions, so that each element of the output sensor retains the temporal receptive field, *i.e.*, is computed only on the sensor information around the single frame. A schema of the GPS/IMU module is reported in Figure 4.2.

Object preprocessing

Let us consider the set of objects detected in frame $t \in \{1 \dots T\}$

$$\mathbf{o}_t = \{o_{t,1}, \dots, o_{t,N_t}\} \quad \text{with} \quad o_{t,i} = (a_{t,i}, p_{t,i}) \tag{4.1}$$

with N_t the total number of objects detected in frame t and with $a_{t,i}$ and $p_{t,i}$ are respectively the appearance features and position of the i-th object detected in frame t. Without loss of generality, let us assume $o_{t,i}$ to be relative to the same real object (*e.g.*, the same vehicle) for each frame t, with $(a_{t,i}, p_{t,i})$ vectors of zeros if the i-th object is not present or not detected in the frame t. Also, instead of considering the maximum number of detections N_t for each frame t, we can think of having a fixed number of detection N_{objs} for each frame, considering as zeros the extra objects for each videos and discarding the exceeding ones.

As a heuristic to decide which objects to keep among the detected ones, we propose to consider the top N_{objs} objects according to detection total volume, *i.e.*, sum of the detected area for each object $o_{t,i}$ for each frame t, and we kept the N_{objs} objects with the largest volumes. This simple rule turned out effective in our experiments, as the relevant objects are close to the subject vehicle for a large number of frames

and, thus, are among the objects with the largest volume. On the other hand, the background objects are generally small, *e.g.*, a vehicle far ahead in an adjacent lane, or appear for few frames.

In this setup, we can express the set of detected objects as a matrix O, as

$$O = \begin{bmatrix} o_{0,0} & \cdots & o_{0,N_{objs}} \\ \vdots & \ddots & \vdots \\ o_{T,0} & \cdots & o_{T,N_{objs}} \end{bmatrix} \tag{4.2}$$

of size $T \times N_{objs}$, with each element of the matrix $o_{t,i} = (a_{t,i}, p_{t,i})$. In order to build the matrix O, it is necessary to link the same real-world object in two consecutive frames, $o_{t,i}$ and $o_{t+1,i}$. To this end, we propose to use a tracking algorithm on the detected objects.

We utilize a greedy tracking algorithm that uses only object positions, detections confidence and class information, that have shown to be effective in Taccari et al. (2018). In particular, starting from frame $t = 0$, we assign a unique tracking *id* to each object $o_{t,i}$ with confidence $c_{t,i} \geq 0.6$. Then, iteratively for each following frame t:

- we compute the matching between the detections in frame $t-1$ and the ones in frame t and assign to each matched object the same *id*;
- we assign a new unique *id* to each unmatched objects in frame t with confidence $c_{t,i} \geq 0.6$;
- we discard all the remaining detections in frame t.

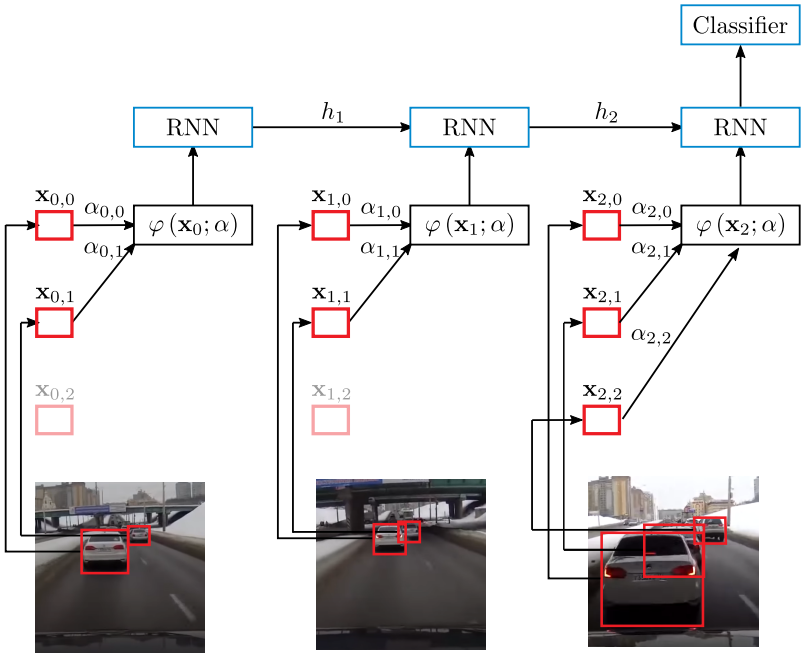
The matching is again a greedy algorithm that first generates a set of candidate detection pairs of the same class and with Intersection Over Union (IOU) greater than a threshold $\theta_{IOU} = 0.2$; then iterates over such set assigning a matching for the pairs with the highest IOU values, removing the matched detections from the candidate set and iterating, so that each object in frame $t-1$ could be a match for at most one object in frame t (maximum bipartite matching).

Object feature extraction

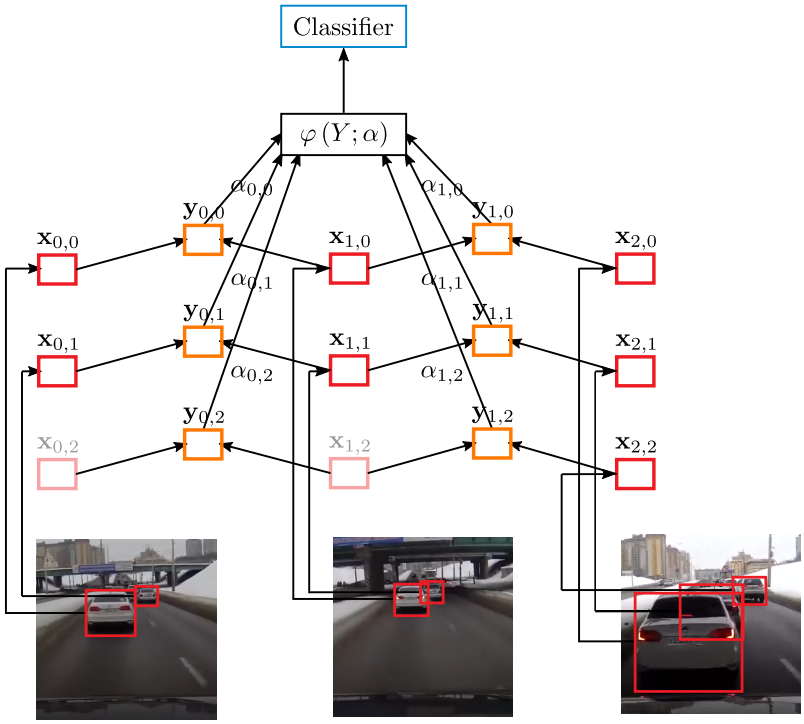
Starting from matrix O and in order to perform the classification, we build a matrix X of shape $T \times N_{objs}$, where each element $\mathbf{x}_{t,i}$ is the concatenation of three feature vectors

$$\mathbf{x}_{t,i} = \left[\mathbf{x}^a_{t,i} \mid \mathbf{x}^p_{t,i} \mid \mathbf{x}^g_{t,i} \right]. \tag{4.3}$$

The feature vector $\mathbf{x}^a_{t,i}$ is relative to the appearance of the object and is obtained feeding the output of the RoI pooling layer of the i-th object and of frame t to a bottlenck



(a) The DSA module as proposed in Chan et al. (2016). The feature vector $\mathbf{x}_{t,i}$ are extracted from each frame (in red) and the attention mechanism is used to produce the attention map $\varphi(\mathbf{x}_i;\alpha)$ and the attention weights $\alpha_{t,i}$. Then, the map is fed to a RNN.



(b) The STAS module proposed in this thesis. The feature vector $\mathbf{x}_{t,i}$ are extracted for each object and for each frame (in red). Then, multiple feature vectors of the same object are aggregated, into a feature vector $\mathbf{y}_{\tilde{t},i}$ representing the evolution of the object (in orange, in the example two objects were considered). Finally, the attention map $\varphi(Y;\alpha)$ and the attention weights $\alpha_{\tilde{t},i}$ are computed on each object and on each temporal segment.

Figure 4.3: Comparision between the DSA module and the STAS module. The figures are displaying single-head attention.

layer, *i.e.*, a linear layer followed by a 1D batch normalization layer and a ReLU activation, in order to reduce the dimensionality of the data and make it comparable with the other stream, as it was proven beneficial in Chapter 3. The feature vector $\mathbf{x}^p_{t,i}$ is relative to the position and class of the detection. It contains:

- the coordinates of the top left corner of the box, normalized in $[0, 1]$;
- the normalized width and height of the box;
- the confidence of the detection;
- a one-hot encoded vector indicating the class of the object.

Finally, $\mathbf{x}^g_{t,i}$ is the output of the GPS/IMU module for frame t, as described above, replicated for each object. In addition, we add a flag indicating whether the box i has been detected in frame t or not. In the latter case, $\mathbf{x}^a_{t,i}$, $\mathbf{x}^p_{t,i}$ and $\mathbf{x}^g_{t,i}$ are zeros. Finally, we add an extra, always present *dummy* object to the object matrix, with the GPS/IMU information $\mathbf{x}^g_{t,i}$ only. Such object will cope with the fact that a video might not have objects other than the subject vehicle equipped with the dashcam, and will forward GPS/IMU information to the final classifier.

In order to extract features that link together the objects in two consecutive frames $\mathbf{x}_{t,i}$ and $\mathbf{x}_{t+1,i}$, some works in the literature propose to use a DSA module (Chan et al. (2016); Suzuki et al. (2018)). Such module uses the attention mechanism to select the relevant object at each time step that are then feed to a recurrent layer (*e.g.*, an LSTM), as showed in Figure 4.3a.

Formally, let $\mathbf{x}_t = \{\mathbf{x}_{t,0}, \dots, \mathbf{x}_{t,N_{objs}}\}$ be the set of all the detected objects in frame t. Then, for each frame t, it can be computed a dynamic weighted-sum of the objects in that frame as

$$\varphi(\mathbf{x}_t; \alpha) = \sum_{i=1}^{N_{objs}} \alpha_{t,i} \mathbf{x}_{t,i} \tag{4.4}$$

with $\sum_{i=1}^{N_{objs}} \alpha_{t,i} = 1$ and where $\alpha_{t,i}$ is computed each frame starting from the previous hidden representation of the recurrent layer $\mathbf{h}_{t-1}$ as

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{N_o bjs} \exp(e_{t,j})} \tag{4.5}$$

where

$$e_{t,i} = \mathbf{w}^\top \sigma \left(W_e \mathbf{h}_{t-1} + U_e \mathbf{x}_{t,i} + \mathbf{b}_e \right) \tag{4.6}$$

is a feed-forward layer to be trained that produces some form of relevance (or similarity) between $\mathbf{h}_{t-1}$ and $\mathbf{x}_{t,i}$.

In the above module, a connection between the features of the same object $\mathbf{x}_{t,i}$ and $\mathbf{x}_{t+1,i}$ occurs only if the attention weights $\alpha_{t,i}$ and $\alpha_{t+1,i}$ on the same object are

high and might get diluted by the presence of the other objects in the weighted sum. Even if this is the case, such relation is exploited in the very last stages of the network, *i.e.*, in the recurrent connection and, thus, it might be difficult to leverage.

In contrast, we believe that extracting object connections in the early stages of the architecture is crucial, as it allows the network to extract features that consider the evolution over time of a single object, *e.g.*, an object getting bigger and bigger or an object moving from left to right. For this reason, after building the matrix X and the feature vectors $\mathbf{x}_{t,i}$, we can extract features $\mathbf{y}_{\tilde{t},i}$ related to the evolution of the objects in the scene over time. This is done by applying the same set of convolutional operations to each object. In order to accomplish this in an efficient way, as reported in Table 4.1, we used 2D convolutional operations of size 3×1, thus insisting on three frames and on a single object. We stacked four convolutions with a growing number of filters $f = 64, 128, 256, 512$, always followed by 2D batch normalization, ReLU activation and 2D max-pooling operations of size 2×1, in order to reduce the temporal dimension while increasing the number of filters. The result is a matrix Y of shape $\widetilde{T} \times N_{objs}$ where each element $\mathbf{y}_{\tilde{t},i}$ represents the evolution of an object i in a temporal segment $\tilde{t}$, with $\tilde{t} \in 1, \ldots, \widetilde{T}$ indices of the reduced temporal dimension.

It is worth noticing, however, that the DSA module, compared to the proposed approach, does not require explicit associations, *i.e.*, tracking, between objects $o_{t,i}$ and $o_{t+1,i}$, as it considers all the objects of a single frame in isolation and the output of the attention layer is independent from the ordering of the inputs. Furthermore, it is not necessary to cap the maximum number of object, as the attention weights could be computed on a different number of object detected at each frame.

Object selection module

We would like to select the most relevant feature vectors $\mathbf{y}_{\tilde{t},i}$, to perform the classification. To this end, we propose to use a multi-head attention layer, introduced in Vaswani et al. (2017). Such layer, starting from three set of vectors, namely keys, queries and values, performs a weighted sum of the values, where the weight assigned to each value is computed by a similarity function between the query and the corresponding key.

The multi-head attention generalize the soft-attention mechanism presented in the previous section. First, it proposes to use a linear projection followed by a dot-product operation instead of a feed-forward operation to be learn to achieve similarity between key and query vectors. Second, it proposes to learn multiple attention operation in parallel (called *heads*) that gets aggregated in the output via concatenation. Formally, let $Q \in \mathbb{R}^{N_q \times d_{model}}$, $K \in \mathbb{R}^{N_k \times d_{model}}$ and $V \in \mathbb{R}^{N_k \times d_{model}}$ be respectively the query, key and value matrices, with N_k total number of key and value vectors, N_q number of query vectors considered, d_{model} size of the vectors forming the key,

Table 4.1: Summarization of object feature extraction module. The tensor dimensions in the output size column are respectively features, objects and frames, with $|\mathbf{x}^a_{t,i}| = 128$, $|\mathbf{x}^g_{t,i}| = 7$, $|\mathbf{x}^p_{t,i}| = 20$, $T = 135$ and with $N_{objs} = 25$.

Layer	**Output Size**	**Structure**
Input ($\mathbf{x}_{t,i}$)	$155 \times 26 \times 135$	$-$
Conv2d	$64 \times 26 \times 135$	1×3, 64, pad 0×1
BatchNorm2d	$64 \times 26 \times 135$	$-$
MaxPool2d	$64 \times 26 \times 67$	1×2
Conv2d	$128 \times 26 \times 67$	1×3, 128, pad 0×1
BatchNorm2d	$128 \times 26 \times 67$	$-$
MaxPool2d	$128 \times 26 \times 33$	1×2
Conv2d	$256 \times 26 \times 33$	1×3, 256, pad 0×1
BatchNorm2d	$256 \times 26 \times 33$	$-$
MaxPool2d	$256 \times 26 \times 16$	1×2
Conv2d	$512 \times 26 \times 16$	1×3, 512, pad 0×1
BatchNorm2d	$512 \times 26 \times 16$	$-$
MaxPool2d ($\mathbf{y}_{t,i}$)	$512 \times 26 \times 8$	1×2
MultiHeadAttention	512	heads $= 8$, dropout $= 0.1$
Fully-connected	10	$-$

query and value matrices. Then, the multi-head dot-product attention is defined as

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O \tag{4.7}$$

$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{4.8}$$

with h number of heads, $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W_i^O \in \mathbb{R}^{hd_v \times d_{model}}$ linear projections, and where

$$Attention(Q, K, V) = softmax\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \tag{4.9}$$

For each head h, thus, the attention layer first projects the input vectors a reduced embedding space of size d_k, then computes a set of attention weights $\{\alpha^h_{\tilde{t},i}\}$, *i.e.*, a similarity measure based on dot product between each key and the query, and use them to perform a weighted combination over the values.

We propose to consider the set of vectors $\{\mathbf{y}_{\tilde{t},i}\}$ as keys and values and the 2D global max-pooling of such vectors as query. The rationale behind this choice is that, ideally, the network will generate features that have higher activations in correspondence to relevant objects. Then, by pooling along the channel axis we would obtain a vector that is representative of the relevant object activations: this has been shown to be effective in highlighting informative regions in Woo et al. (2018) and to

be good for the classification task in Chapter 3. By using such vector as a query, we are training the network so that the attention weights are higher in correspondence to the object features that are the most similar to the pooled vector, thus, where the object activations are higher and to the relevant objects. Formally, we define the key K, query Q and value V matrices as

$$Q = \left[MaxPool(\mathbf{y}_{0,0}, \dots, \mathbf{y}_{\widetilde{T},N_{objs}}) \right]$$
$$K = \begin{bmatrix} \mathbf{y}_{0,0} \\ \cdots \\ \mathbf{y}_{\widetilde{T},N_{objs}} \end{bmatrix} \quad V = \begin{bmatrix} \mathbf{y}_{0,0} \\ \cdots \\ \mathbf{y}_{\widetilde{T},N_{objs}} \end{bmatrix} \tag{4.10}$$

With $d_{model} = |\mathbf{y}_{t,i}|$, $N_q = 1$, $N_k = \widetilde{T} \cdot N_{objs}$, and with the projection size $d_k = 64$ and the number of heads $h = 8$.

The usage of the max pooling as query vector has been exploited in other field and application such as Woo et al. (2018); Hu et al. (2018); de Andrade et al. (2018), but never to perform the selection of relevant objects as we propose, thus, we are naming this layer Spatio-Temporal Attention Selector (STAS).

Finally, feature vector in output of the STAS module is fed to a fully-connected layer of size of the number of unsafe maneuver classes to perform the classification. As an alternative, we propose to use such max-pooled vector directly for the classification, as shown effective in Chapter 3. Note that, by doing so, the features used for the classification are pooled from any feature vector of Y instead of being forced, by the attention layer, to belong mostly to a single feature vector. Relaxing such constraint might improve performance at the expense of explainability.

4.3 Experimental results

We run all the experiments in the *clip around the event* setup presented in Seciton 3.3. All the experiments were conducted using Adam optimizer Kingma and Ba (2014), minimizing standard Cross Entropy Loss with class weights and with weight decay of 10^{-4}. We used an initial learning rate of 10^{-3}, decreased by a factor of 0.5 after 50, 70, 90 and 110 epochs. This applies to all the experiments, but the one with the LSTM, which has a longer convergence time, where we decreased the learning rate after 100, 150, 200 and 250 epochs. We also clipped the norm of the gradients to 1.0 to prevent exploding gradients.

In this Section, we first present the comparison between the state-of-the-art and the proposed approaches with the best setup and parameters, which choices will be then discussed in the following ablation study. As in Chapter 3, we used the mean average precision (mAP) for evaluation, that is equivalent to computing the mean

area under the precision-recall curve for each class. This metric accounts for both precision and recall and is robust to class imbalance.

Comparison with state-of-the-art

We consider two variants of the proposed approach.

- *Ours + STAS* is the proposed architecture with the extracted features $\mathbf{y}_{\tilde{t},i}$ fed to the STAS module, described in Section 4.2.
- *Ours + MaxPool* is the same as above, but with the extracted features $\mathbf{y}_{\tilde{t},i}$ fed to a global max pooling layer.

We compare the proposed approaches with

- *Two-Stream*, *i.e.,* the two-stream architecture as proposed in Chapter 3, using both the features extracted from the full frame and the GPS/IMU data.
- *DSA + LSTM*, *i.e.,* the porting to the unsafe maneuver classification problems of DSA-based architectures proposed for accident anticipation (Chan et al. (2016); Suzuki et al. (2018)). In particular, the structure of the network is the same proposed in Section 4.2 up to the extraction of the object features $\mathbf{x}_{t,i}$. Then, we compute the attention map on each object feature per frame and fed the result to a LSTM. In contrast with their approach, we use the multi-head dot product attention instead of soft attention. This is both for a fair comparison with our proposed approach and as it has shown superior results compared to soft attention in Vaswani et al. (2017) on the Neural Machine Translation task. Furthermore, we used $\mathbf{x}_{t,i}$ as object features instead of the one proposed in the original papers, which were describing similar aspects but in a slightly different way, *e.g.,* using IDT to describe object motion, in order to be able to assess the impact of the different attention modules under similar conditions. Finally, we used standard cross entropy loss instead of the accident anticipation losses proposed in the papers.

The comparison with the state-of-the-art is reported in Table 4.2, along with the inference time and per-class average precision (AP). In the top row, we report several results not considering the appearance features, *i.e.,* having $\mathbf{x}_{t,i} = \left[\mathbf{x}^{p}_{t,i} \mid \mathbf{x}^{g}_{t,i}\right]$: while this is a valuable set-up to test as processing the video is computationally expensive, it can also be also be a lightweight component to add to a pipeline already performing object detection.

First, we can observe how the proposed approaches outperform the baselines we evaluate and, in particular, the DSA-based approaches from Chan et al. (2016); Suzuki et al. (2018). It is worth highlighting, however, that the sensor only version

of the Two Stream architecture is not using the object positional features. In the bottom row of Table 4.2, instead, we report the results considering also the appearance features. Again, we can observe that the proposed approaches outperforms the Two Stream baseline, either considering the original backbone pre-trained on Places-365 or the fine-tuned version proposed in Section 3.4.

Ablation study

The majority of the experiments were made without using the appearance features and are reported in Table 4.3. First, we can see how the usage of the depthwise separable convolutions (DW), without mixing sensor information in the early stages, significantly improves performance over the standard 1D convolutions (C1D): mAP is 0.576 when using C1D (first row) and it raises to 0.633 (+0.057) by switching to DW (third row). This might suggest that mixing up sensors in the early stages would promote high informative sensors (*e.g.*, the flow-direction accelerometer) and penalize low-informative ones, that could, however, be crucial in distinguishing some corner cases. Second, we evaluated the effect of the possible values of N_{objs} on the final prediction metric. We found the optimal value to be $N_{objs} = 25$, however, the usage of a higher number of objects is not significantly detrimental. As reference, we reported also the results with $N_{obj} = 0$, *i.e.*, considering only the dummy object, to observe the effect of adding the objects information. It is worth to highlight that the setup with $N_{objs} = 25$ is outperforming $N_{objs} = 100$ which is outperforming $N_{objs} = 250$: it shows how the heuristic we used to limit the number of object is not discarding the relevant ones. Third, we observe how the usage of the max-pooling layer instead of the STAS module, *i.e.*, not forcing the network to pick a single object, is slightly beneficial for the performance, at the expense of the explainability. It is worth to highlight, however, that the two results are not too distant from each other, as the difference in terms of mAP is small (0.013). Finally, we tested the performance of the model when removing the GPS/IMU features completely, *i.e.*, considering $\mathbf{x}_{t,i} = \mathbf{x}^p_{t,i}$, as such data might not be available in some vision-only application scenarios. We observed a severe drop of roughly 0.18 of in the overall *mAP*, showing the importance of such type of data, if available, to tackle the unsafe maneuver classification and confirming the results of Section 3.3.

The experiments in Table 4.4 show that, also when considering the appearance features, the architecture with max-pooling layer outperforms the STAS module by a relatively small margin in terms of mAP (0.012), again at the expense of explainability. Furthermore, we can see that the usage of the backbone fine-tuning proposed in Section 3.4 provides a significant improvement in mAP also in the proposed methodology, showing the effectiveness of the method. Moreover, again, the experiments without the GPS/IMU data, *i.e.*, having $\mathbf{x}_{t,i} = \left[\mathbf{x}^p_{t,i} \mid \mathbf{x}^a_{t,i}\right]$ show that such type

Table 4.2: Per-class results and inference time of the proposed models, without appearance (top row) and with appearance (bottom row). † the inference time of the object detection algorithm is not included.

Model	**Inference**	**Average Precision (AP)**										**mAP**
		SL	ST	NSL	NST	SB	SOE	SLC	SO	NSO	O	
Two Stream (Sensors only)	11 ms	0.35	0.55	0.57	0.30	0.80	**0.94**	0.59	0.65	0.59	0.23	0.556
DSA + LSTM	121 ms †	0.15	0.52	0.62	0.43	0.84	0.90	0.60	0.46	0.37	0.67	0.555
Ours + STAS (w/o appearance)	20 ms †	0.33	0.54	0.75	0.59	0.90	0.89	0.48	0.48	**0.68**	0.68	0.633
Ours + MaxPool (w/o appearance)	20 ms †	**0.38**	0.49	0.81	0.66	0.92	0.91	0.64	0.42	0.60	0.61	0.646
Two-Stream	917 ms	0.28	0.54	0.61	0.60	0.91	0.92	0.60	**0.68**	0.62	0.59	0.635
Two-Stream (fine-tuned)	917 ms	**0.38**	0.49	0.81	0.66	0.92	0.91	0.64	0.42	0.60	0.61	0.690
Ours + STAS (fine-tuned)	952 ms †	0.29	0.55	0.78	0.64	**0.96**	0.91	**0.82**	0.61	0.64	0.68	0.700
Ours + MaxPool (fine-tuned)	952 ms †	0.34	**0.72**	**0.87**	**0.78**	0.95	0.89	0.73	0.45	0.66	**0.72**	**0.712**

Table 4.3: Tests without using appearance features.

Model	**Sensors**		N_{objs}	**Object selection**			**mAP**
	DW	C1D		DSA	STAS	MaxPool	
Ours + STAS (no appearance, C1D)		✓	25		✓		0.576
Ours + STAS (no appearance)	✓		0		✓		0.525
Ours + STAS (no appearance)	✓		25		✓		**0.633**
Ours + STAS (no appearance)	✓		100		✓		0.614
Ours + STAS (no appearance)	✓		250		✓		0.602
Ours + MaxPool (no appearance)	✓		25			✓	**0.646**
Ours + STAS (no appearance, no sensors)			25		✓		0.458
Ours + MaxPool (no appearance, no sensors)			25			✓	0.459

Table 4.4: Tests using appearance features

Model	**Sensors**	N_{objs}	**Appearance**			**Object selection**		**mAP**
	DW		Places-365	ImageNet	Fine-tuned	STAS	MaxPool	
Ours + STAS (Places-365)	✓	25	✓			✓		0.656
Ours + STAS (ImageNet)	✓	25		✓		✓		0.657
Ours + STAS (fine-tuned)	✓	25			✓	✓		0.700
Ours + MaxPool (fine-tuned)	✓	25			✓		✓	**0.712**
Ours + STAS (fine-tuned, no sensors)		25			✓	✓		0.476
Ours + MaxPool (fine-tuned, no sensors)		25			✓		✓	0.506

of data are crucial to solve this task, as by removing them the performance drops of more than 0.2 in *mAP*. Finally, we can observe how each experiment outperforms the appearance-less counterpart of Table 4.3, clearly showing the effectiveness of this type of features.

In Table 4.2, we report the per-class AP. Although there is no clear winner, again in the setup without the appearance features the proposed approaches are in general outperforming the baselines in the various classes and the approaches with the appearance are outperforming their counterparts. `SB` and `SOE` are the classes with the highest AP, probably because such maneuvers are fairly well distinguishable from the rest and are the majority classes in the dataset. On the other hand, most of the models show a relatively low AP on the minority classes (*i.e.*, `SL`, `ST`, `SO` and `NSO`).

Table 4.2 also reports the inference times of the various models. We can observe how the inference time of the proposed methods is close to the Two Stream baseline one, thanks to the ROI pooling layer that allows the computation of the objects' appearance features in a single forward pass. It is worth to highlight, however, that the reported times does not include the object detection algorithm inference time. We used Faster-RCNN with ResNet-101 with an inference time of 72ms per image, thus requiring 9.7s for a full 135 frames video, as we prioritized accuracy over speed in the detection generation process. Nevertheless, requiring object detection as a prerequisite does not add any extra overhead if the proposed architecture is deployed in an application already performing object detection. Finally, please note how the object detection algorithm used at inference time could be, in principle, different, *e.g.*, faster, from the training one, although we did no experimentation in this regard.

4.4 Results visualization

In this Section, we provide qualitative results on the object being selected by the proposed method, since labels on the relevant object in each scene were not available in our dataset.

We consider the *Ours + STAS* model with appearance features and fine-tuned backbone, with the goal of visualizing the object being selected for the classification. This can be accomplished by just looking at the attention weights $\alpha_{t,i}$, since the output is a weighted combination of the object features. The results are the heatmaps in Figures 4.4, 4.5, 4.6, 4.7, which have been independently normalized into $[0, 1]$ for visualization purposes. Note that each heatmap has the objects on the vertical axis and the reduced temporal dimension on the horizontal axis and, thus, has shape $N_{objs} \times \widetilde{T}$. Also, the top row is always relative to the dummy object that, for visualization reasons, has been reported also in the top left corner of each frame.

Figure 4.4 shows a Subject Turn (`ST`) event in which the subject enters an intersection and starts to perform a turning maneuver ignoring the incoming vehicle in the

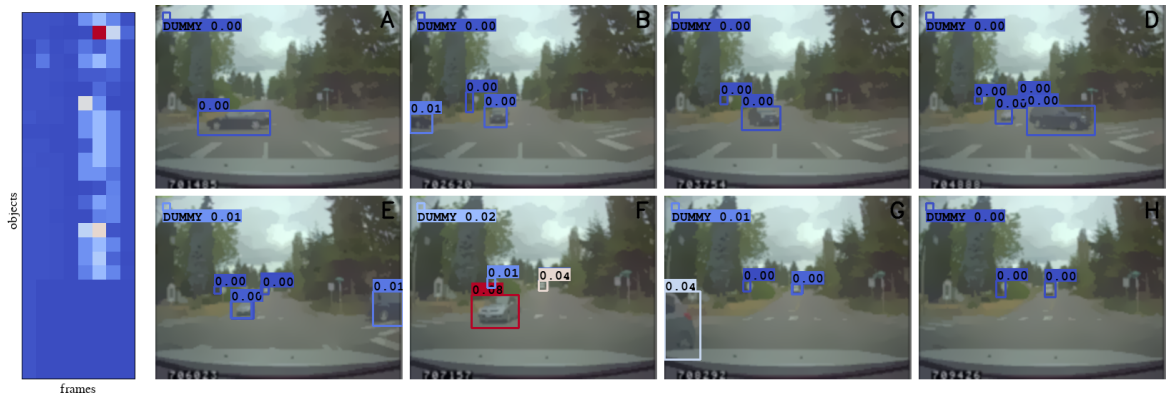


Figure 4.4: Subject turn (ST) event

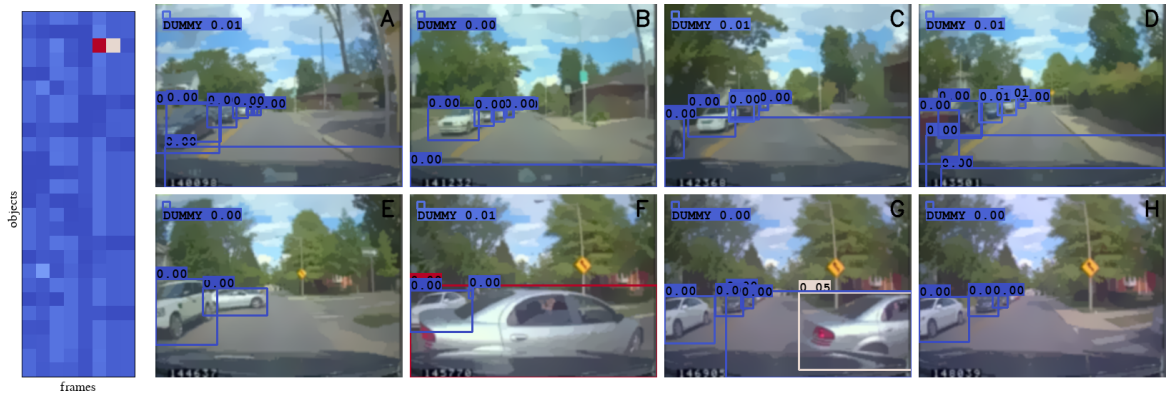


Figure 4.5: Non-subject turn (NST) event

opposite direction, being forced to interrupt the maneuver to not hit the other vehicle. The model is mostly looking at only the incoming vehicle features to perform the prediction in the moment in which the subject is forced to interrupt the maneuver, correctly identifying the relevant object, *i.e.*, the one in danger, at the right moment.

Figure 4.5 shows a Non-subject Turn (NST) event in which the subject is travelling on a single lane road approaching an intersection, when another vehicle enters and go through such intersection ignoring the subject, that is forced to hard brake. Again, the model is looking mostly at the right object, *i.e.*, the car cutting into the subject vehicle path, at the relevant moment.

The above example considered scenarios in which it is present a relevant object, either dangerous or in danger. Some of the maneuver, though, are relative just to the ego-vehicle. For instance, Figure 4.6 shows a Subject Over Edge (SOE) event in which the subject is going over the edge of the road. There is no object in the scene, but a detection of the subject vehicle hood detected as vehicle by the object detection algorithm. In this particular scenario, the sensors features are used to perform the prediction, extracted mainly from the dummy object. In contrast, Figure 4.7 shows

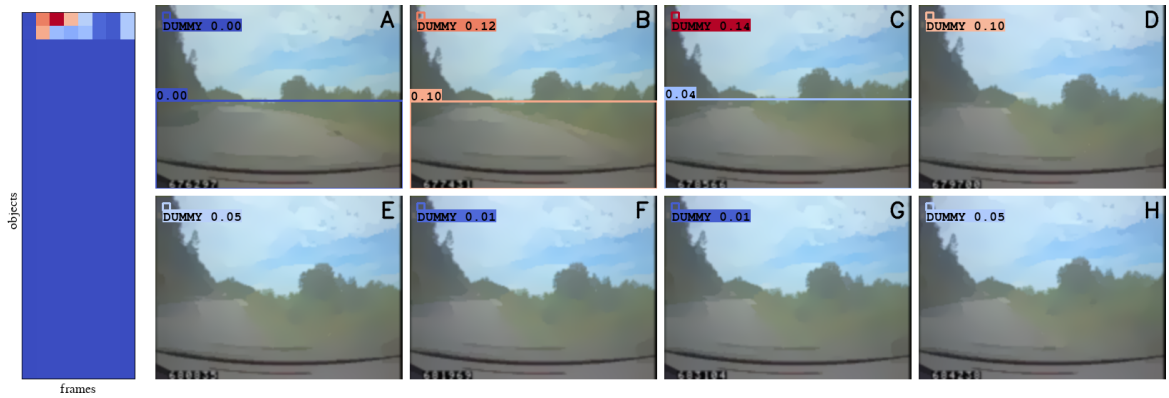


Figure 4.6: Subject over edge (SOE) event with no objects

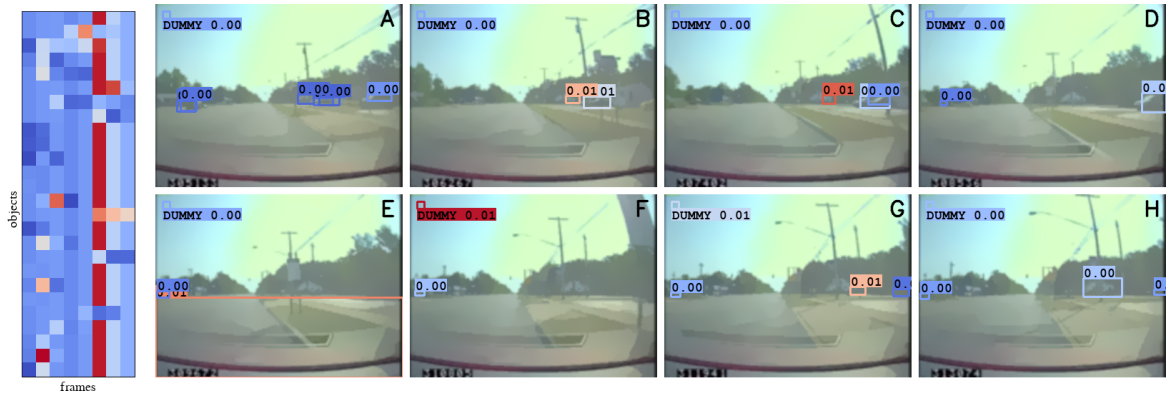


Figure 4.7: Subject over edge (SOE) event with not relevant objects

a Subject Over Edge (SOE) event with multiple detected vehicles, which are not relevant to perform the classification. The network is assigning an equal weight to most of them, that gets normalized close to 1 in the visualization of the heatmap. We believe that this outcome is due to the fact that the feature vector of each object $\mathbf{x}_{t,i}$ is composed also by the sensors features $\mathbf{x}^{\mathcal{S}}_{t,i}$, which are crucial for the prediction of this kind of event. Thus, likely, the resulting $\mathbf{y}_{\tilde{t},i}$ features will be mostly a combination of the sensor feature and be pretty much the same for all the object, *i.e.*, they will be independent from the object appearance and position. As a result, the attention mechanism, which is based on a similarity measure, will find all the features to have an equal contribution.

One can extend this concept to a context where the same object is detected multiple times. Then, the appearance and position feature, and thus the resulting $\mathbf{y}_{\tilde{t},i}$ would likely look similar and the STAS module would assign similar weights to each detection, again detecting multiple relevant objects. We found this to be a limitation of the proposed approach, which can however probably easily overcome by assigning low relevance to all the objects in the scenes when the largest attention weights

are spread all over the matrix, *i.e.,* when there is no single clear peak in the weights.

4.5 Conclusions

We presented an architecture that combines object appearance and position, the video stream and the sensors stream to tackle the unsafe maneuver classification problem. We discussed two variants of this architecture: the first one uses the max-pooled vector of the extracted object features directly for the classification, while the second one leverage the newly introduced STAS module to also identify the relevant object in the video, offering network explainability as an interesting by product. We observe that while the first architecture is better in terms of mAP on the unsafe maneuver classification task, the gap between the two is small, suggesting that explainability comes with a cost on overall performance, but that such cost can be acceptable, if explainability is desired. Furthermore, we presented several methodological and practical novelties over the relevant works in the literature: first, the usage of depthwise convolutions to process sensors data, in order to maintain the semantic of each sensor in deeper stages of the network; second, the usage of an RoI pooling layer to extract object appearance features in an efficient way both during training and inference; third, we introduced the concept of extracting features describing the evolution of a single object over time, by using a tracking algorithm, and using such features for classification; fourth, we presented a new backbone fine-tuning strategy tailored to the unsafe maneuver classification task that could however potentially be extended to other similar domains. Our experiments showed that the proposed approach outperforms the *state-of-the-art* on the unsafe maneuvers classification task and empirically showed the superiority of the STAS module compared to other attention-based methods in the literature on this particular task. Finally, we presented qualitative results on the capability of the network to select the relevant object, highlighting its advantages and limitations.

Chapter 5

Video captioning of safety-critical events from dashcam data †

In this Chapter, we address the problem of generating captions of safety-critical events on the road, considering as input the video acquired from a dashcam and the GPS/IMU data. We propose an encoder-decoder architecture, where the encoder is the architecture presented in Chapter 4, pre-trained on the unsafe maneuver classification task. The encoder leverages the output of an object detection network to compute features on the evolution of the objects in the scene over time, at the same time performing sensor fusion with the GPS and IMU data. We test different decoder models inspired by the video captioning literature, specifically a hierarchical decoder compared to a standard one and the usage of attention compared to simple pooling. We tested our methods on the novel SHRP-X dataset, consisting of the SHRP2 naturalistic driving dataset, re-annotated by us for the present task and released to the scientific community as additional contribution.

Video captioning, the task of automatically generating natural language descriptions of videos, has recently drawn the attention of the scientific literature, as it lies at the intersection between computer vision and language processing (Chen et al. (2019)) and, thus, inherits the challenges of the two fields of research. Practical applications include leveraging descriptions for video retrieval and indexing, and helping people with visual impairments. In parallel, scientific papers and appli-

†Part of the content of this chapter has been submitted in:

- M. Simoncini, N. Bellaccini, D. Coimbra de Andrade, L. Kubin, S. Salti, L. Taccari, L. Sarti, F. Schoen, F. Sambo, "Video captioning of safety-critical events from dashcam data", *IEEE Transactions on Intelligent Transportation Systems*, 2022.

cations concerning road safety and safety-critical events analysis are of significant importance, as highlighted in Chapter 1.

In this Chapter, we propose to join these two fields, addressing the novel problem of video captioning of safety-critical events. This combination opens to other automotive-specific applications, like automatic emergency call leveraging text-to-speech and, in the autonomous driving field, to provide human-understandable feedback to the driver on the motivations behind a given action, as in Kim et al. (2018).

To our knowledge, no other work in the literature tried to address the captioning of safety-critical driving events and no dataset on the topic is available. For this reason, as part of our contributions, we decided to annotate a subset of the videos of the SHRP2 NDS dataset (Hankey et al. (2016b)) with human-understandable descriptions of the events in the videos. We collected such annotations into the SHRP-eXplained (SHRP-X) dataset. Many captioning datasets and papers focus on a single sentence to describe the image or video at hand, as in Venugopalan et al. (2014); Pan et al. (2016); Venugopalan et al. (2015); Yao et al. (2015). In our case, the events we aim to describe are complex and are often based on a cause-effect chain. Thus, multi-sentence descriptions seems a natural fit to avoid trivial annotations and give the necessary level of detail, as shown in Figure 5.1, and we annotated the dataset with a multi-sentence paragraph for each example, as in Yu et al. (2016).

Ground truth SV approaches an uncontrolled intersection. V2 is on the intersecting road on the right. V2 is obscured by another vehicle. V2 starts to turn left across the path of SV. SV must brake hard to avoid a collision with V2.
Prediction SV approaches an uncontrolled intersection. V2 is at the intersection on the right. V2 turns left across SV's path. SV brakes hard to avoid a collision with V2. V2 brakes. SV continues on.

Figure 5.1: An example of the input video and detected objects with the corresponding annotations. In green, the object detected as traffic light, in yellow the objects detected as vehicles. Each sentence of the annotation describe a single action in the video that ultimately lead to the dangerous situation.

We tackle the problem with an encoder-decoder architecture, as it is common in the video captioning literature. Following a recent trend, we ground the feature extraction on the output of an object detector (Zhou et al. (2019); Anderson et al.

(2018b); Herdade et al. (2019b); Lei et al. (2019); Lu et al. (2018)), as it allow to generate captions that are more adherent to the salient part of the image and not bounded to a equally-spaced grid (Anderson et al. (2018b)). Also, we incorporate in the encoder the features extracted from the GPS/IMU module, as such sensors data have shown to be crucial in safety-critical events classification tasks, as shown in Chapter 3 and Chapter 4. To meet both needs, we use as an encoder the STAS architecture proposed in Chapter 4.

The encoder, thus, first extract features relative to the appearance and position of the objects in each frame, then combines the features of the same real object (*e.g.*, a vehicle) over several frames by using a tracking algorithm. The output is a tensor of shape *number of objects* $\times$ *number of frames* $\times$ *number of features*. The features for each object and for each frame gets enhanced by appending an aligned feature vector extracted from the GPS/IMU input, performing sensors fusion. Finally, a set of further-processed features are extracted using convolutional operations over solely the temporal dimension, thus, preserving the objects identities, which represents the evolution of an object in a given time span (*e.g.*, the vehicle performing a turn). We use a pre-trained encoder on the unsafe maneuver classification task.

As for the decoder, we tests multiple configurations that were shown to be effective in the literature, and, in particular, the usage of dot-product attention over the object features or a simple pooling and the usage of a hierarchical decoder compared to a single-loop one.

Finally, as it has been shown to be effective in Yao et al. (2017); Pan et al. (2017), we propose to boost the overall caption quality with extra semantical information on the safety-critical event and, in particular, with the presence or absence of a crash and the unsafe maneuver type leading to the dangerous situation.

5.1 Related works

To the best of our knowledge this is the first work attempting to perform video captioning of safety-critical events. In this Section, we provide a review of the literature on video captioning and highlight the aspects that could be applied to the problem we aim to address.

5.2 Video Captioning

Most of the state-of-the-art results on video captioning are inspired by the image captioning literature and obtained with an encoder-decoder architecture, where the encoder extracts a feature vector representation of the input video, also referred to as *context vector*, and the decoder produces the output sentence starting from such a representation. Generally, the encoder is based on convolutions while the decoder

is a recurrent network. Venugopalan et al. (2014) was one of the first works going in this direction: the authors applied a 2D convolutional network, pre-trained on an image classification task, to each frame on the video, then average-pooled the output into a single feature vector, which was eventually fed into the decoder. Such architecture is extended in Pan et al. (2016), where the authors used both a 2D and a 3D CNN to compute the context vector, and in Venugopalan et al. (2015), where the authors included the optical flow stream by using a pre-trained CNN on an activity recognition task. Yao et al. (2015) propose to generate the context vector dynamically at each step of the decoding process, leveraging soft-attention.

These approaches encode the video into an arbitrary temporal or spatio-temporal uniform grid, from which the context vector is extracted. However, the entities in the scene are not forced to follow a grid-like structure: for instance, a vehicle lying on the grid boundary would be treated differently from one fully contained in a grid cell. To overcome such limitation, recent works in the image captioning literature propose, instead, to ground the generated captions on objects and salient regions of the inputs, by considering the output of an object detector as input, as in Anderson et al. (2018b); Herdade et al. (2019b); Lei et al. (2019), or by incorporating the detection process in an end-to-end manner, as proposed in Lu et al. (2018); Zhou et al. (2019).

While the initial attempts mainly focused on simple and short sentences, some researchers addressed also the generation of entire paragraphs, for which a simple decoder might not be enough. In this regard, Yu et al. (2016) propose to use two nested recurrent modules, namely a paragraph generator and a sentence generator, that they named hierarchical decoder. The sentence generator is built upon a recurrent module for language modeling and on the attention mechanism to parse the input video. The paragraph generator is another recurrent module that produces the initial state of the sentence generator for the next sentence starting from a compressed version of the previously produced sentence. Wang et al. (2020) extended this approach to dense video captioning, introduced in Krishna et al. (2017), where the goal is jointly to produce the caption and temporally localize the event in the video. They also introduced a Cross Modal Gating (CMG) module to weight the contribution of the input and of the previously produced words.

Yao et al. (2017) propose to boost image captioning by adding attributes to the context vector, *i.e.*, high-level semantic information obtained in a Multi Instance Learning setup. Pan et al. (2017) extend this idea to video captioning, by extracting attributes both relative to the image, *i.e.*, objects and scenes, and to the video, *i.e.*, actions and attributes with a temporal dynamic.

5.3 The SHRP-X dataset

The SHRP2 NDS dataset (Hankey et al. (2016b)) is a collection of more than 8800 safety-critical events, gathered by more than 3300 drivers between 2010 and 2013. To address the captioning task, we consider the video acquired from the dashcam installed inside the vehicle and the sensor data from the GPS/IMU sensors.

We annotated such events with a set of temporally ordered sentences. Such sentences has a subject-predicate structure, were the subject is generally the subject vehicle (SV), other vehicles in the scenes (indicated with V2, V3, V4) or other actors (pedestrian, bicycles, animal, objects). The predicate, instead, is composed by a single verb (*i.e.,* action) at present simple tense, followed by its dependents. Generally, the set includes:

- one or more sentences describing the environment, *e.g.,* the presence of an intersection or a stop sign, the presence and position of other relevant vehicles or actors in the scenes;
- a set of sentences describing the events or the maneuvers computed by the various subjects, *e.g.,* change of lanes, going through an intersection, traffic light change, loss of control of a vehicle;
- a set of sentences describing the event itself or the reactions that the actors involved had with respect to it, *e.g.,* braking, steering in the adjacent lane;
- and finally a set of sentences describing what happened after the event, *e.g.,* the actors continuing driving or remaining stopped.

Table 5.1 shows the total number of annotation, sentences and distinct words. As the number of verbs and noun describing the event on the roadway is limited, the total number of distinct words is fairly small, 576. On the other hand, due to the complexity of safety-critical events, a fair amount of sentences is required in order to have a complete description. We used 17,647 sentences for 2,982 annotations, with an average of roughly 6 sentences per annotation.

To give a glimpse of the content of the dataset, we are reporting the results of a clustering over the sentences composing the annotations. As we are interested in highlighting what happens in a given sentence, we first replaced the instance-specific part of a sentence with placeholder, *e.g.,* replacing the actors (SV, V2, V3, V4) with `SBJ`, the direction (left, right) with `DIRECTION`. Then, we run an Agglomerative Clustering algorithm (Pedregosa et al. (2011)), considering as distance d the inverse of the METEOR (Banerjee and Lavie (2005)) score. As the METEOR score is not symmetric, we considered the average of the score between the first and the second sentence and between the second and the first to achieve symmetry. Thus, given

Table 5.1: SHRP-X dataset statistics

SHRP-X dataset	
Total number of annotations	2,982
- Train	2,361
- Validation	310
- Test	311
Total number of sentences	17,647
Total number of words	113,160
- Total number of distinct words	576

Table 5.2: SHRP-X sentence clustering results

Sentence centroid	**Cluster size**
`SBJ` brakes.	776
`SBJ` brakes hard to avoid a rear-end collision with `SBJ`	427
`SBJ` brakes to avoid a collision with `SBJ`.	417
`SBJ` steers `DIRECTION` to avoid a collision with `SBJ`.	321
`SBJ` is the leading vehicle.	300
`SBJ` approaches an uncontrolled intersection.	291
`SBJ` continues on.	278
`SBJ` begins to change lanes to the `DIRECTION`.	271
`SBJ` approaches a signalized intersection.	244
`SBJ` change lanes to the `DIRECTION`.	241
`SBJ` turns `DIRECTION`.	240
`SBJ` steers `DIRECTION`.	235
`SBJ` is ahead in the adjacent `DIRECTION` lane.	230
`SBJ` accelerates.	202

two sentences a and b, the distance we considered is the following

$$d(a,b) = -\frac{1}{2}\left(\text{METEOR}(a,b) + \text{METEOR}(b,a)\right) \tag{5.1}$$

.

We tested different thresholds on the number of clusters to select and found 1500 to give the best silhouette score. In Table 5.2, we are reporting the centroids of the clusters and the cluster sizes. The most frequent sentences are those describing the event itself, in the form like *SBJ brakes* or *SBJ brakes to avoid a collision with SBJ*. Within the most common sentences there are also the ones describing the environment, *e.g., SBJ is the leading vehicle* or *SBJ approaches an intersection*, and the ones that describe a driving action, like *SBJ turns DIRECTION* and *SBJ change lanes to the DIRECTION*. Note that such descriptions could be relative to a dangerous maneuver or to a not-dangerous one, depending on the context. For instance, in the annotation [...] *V2*

change lanes to the right. SV has to brake to avoid a collision. [...] the lane change sentence is relative to an unsafe maneuver and is likely to be the cause of the safety-critical event; in the annotation [...] *V2 change lanes to the right. SV change lanes to the right behind V2.* [...] the same sentence is used in a safe situation.
The annotations are available at `https://github.com/mattsim/SHRP2-X` *.

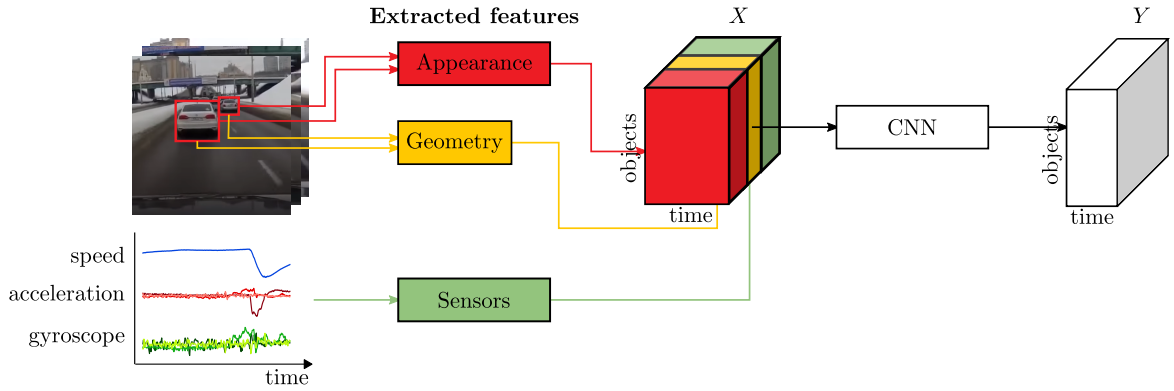
5.4 Encoder-decoder model for safety-critical events captioning

To address the captioning problem, we propose an encoder-decoder architecture, which was shown to be very effective in the vast majority of the works in the video captioning literature, as reported in Chen et al. (2019). The encoder compresses the input into a single vector representation that is then fed to the decoder, which is usually a recurrent neural network trained to predict the next word from the encoder output and its internal state. Figure 5.2 reports a schematization of the proposed model.

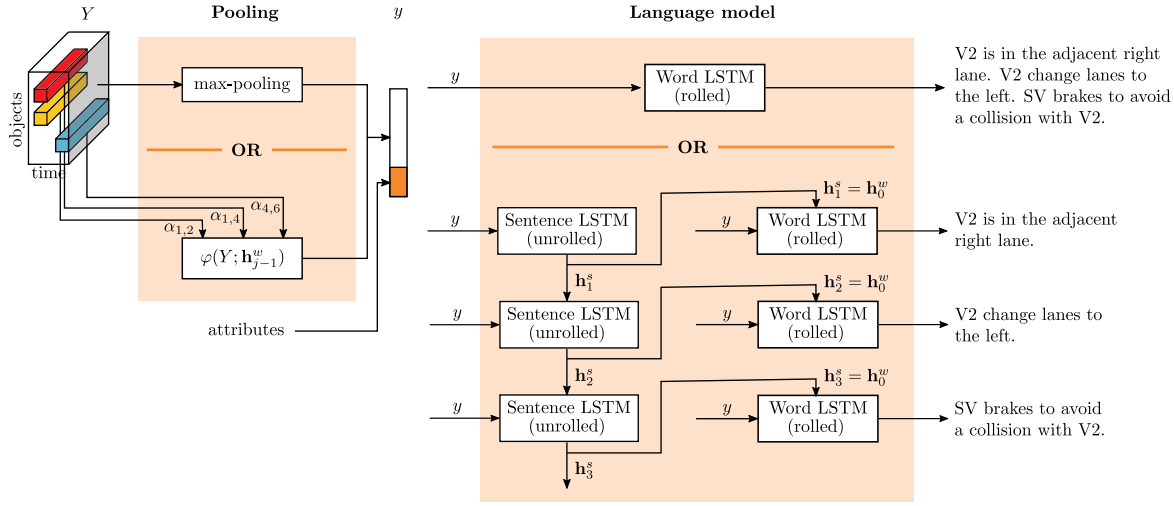
Encoder structure

For the encoder, we used the architecture proposed in Section 4.2. Such architecture takes as input the raw video, the output of an object detection algorithm for each input frame, and the raw sensors data (GPS/IMU), to produce features relative to the evolution of each object in the scene over several consecutive frames. This choice is motivated by several reasons. First, features extracted from object detection outputs, in contrast with using a fixed grid, allow the decoder to model its output explicitly on the entities in the scene, and it is shown to be effective in the literature (Zhou et al. (2019); Anderson et al. (2018b); Herdade et al. (2019b); Lei et al. (2019); Lu et al. (2018)). Second, in contrast with other works that extracts object feature vector for each frame and incorporates the temporal information in the final layers Zhou et al. (2019); Lei et al. (2019), such architecture leverages a tracking algorithm to extract feature of the same real object (*e.g.*, a car) over time. This early feature aggregation over time strategy has shown to be effective in comparison with a late one on the unsafe maneuver classification task in Chapter 4. Third, it addresses sensor fusion of two heterogeneous inputs, video and sensor data, in an effective way. Finally, we believe that the feature extraction layers pre-trained on a safety-critical event classification task would be efficient for a closely-related captioning task.

*The annotations will be available once the paper we submitted is accepted, currently under review. Thus, they might not yet be available at the specified URL.



(a) The proposed encoder, described in detail in Chapter 4, pre-trained on the unsafe maneuver classification task.



(b) The proposed decoders. On the left, the two strategies to generate the **y** vector, either simple pooling or by using attention. On the right, the recurrent model used to generate the captions, either single-loop or hierarchical.

Figure 5.2: A schematization of the proposed architecture.

The feature extraction of the encoder is performed exactly as reported in Section 4.2 up to the generation of the feature matrix Y. Thus, a matrix of shape is $T' \times N_{objs} \times c'$, with T' newly reduced temporal dimension, c' feature dimension and N_{objs} maximum number of objects considered, where each element $y_{t,i}$ represent the evolution of the object i over the temporal segment t.

Decoder structure

The role of the decoder is to translate the representation produced by the encoder into a human readable text. Generally, at the core of the decoder is a RNN trained to predict the next word given the output of the encoder and the previous internal state of the network. For paragraph captioning, though, hierarchical RNNs, *i.e.*, a

decoder composed by two asynchronous RNNs, namely a sentence RNN and a word RNN, have proven effective in Yu et al. (2016); Wang et al. (2020). The sentence RNN stores the information of the produced sentences and is triggered at the start of every sentence, producing the word RNN initial state. The word RNN, in contrast, is trained to produce the next word, as it would do in a single-loop decoder.

Moreover, the output of the encoder is a feature tensor including all the objects over different segments, that has to be reduced to a single vector to be handled by the decoder. Common strategies adopted consist either in the use of a simple pooling layer, as in Venugopalan et al. (2014); Pan et al. (2016); Venugopalan et al. (2015), or in the use of attention to dynamically generate the vector, as in Yao et al. (2015); Yu et al. (2016); Krishna et al. (2017); Wang et al. (2020). While attention is in general outperforming the simple pooling solution in various works in the literature, it requires a good amount of data to be properly trained.

However, as the captioning task we are addressing is slightly different, *e.g.*, the sentences are well structured and the dictionary size is fairly small, we are not doing any prior assumption and testing all four possible combinations, as schematized in Figure 5.2b.

Before explaining the proposed decoders in depth, let us introduce some notation. All the decoders are based on Long Short Term Memory (LSTM) cells. In the following Sections, as proposed in Anderson et al. (2018b), we will refer to the LSTM operation with the following notation

$$\mathbf{h}_t = LSTM\left(\mathbf{x}_t, \mathbf{h}_{t-1}\right) \tag{5.2}$$

where $\mathbf{x}_t$ and $\mathbf{h}_t$ are respectively the LSTM input and output vector at time t. We omit the variables associated to the memory cells for notational convenience. Let us define the ground truth captions for each annotation W as

$$\mathbf{W} = \{\mathbf{W}_0, \mathbf{W}_1, \ldots, \mathbf{W}_{N_p}\} \tag{5.3}$$

$$\mathbf{W}_i = \{w_0^i, w_1^i, \ldots, w_{N_s^i}^i\} \tag{5.4}$$

with $\mathbf{W}_i$ the i-th sentence of annotation $\mathbf{W}$ and with w_j^i the j-th word of sentence $\mathbf{W}_i$. Let N_p denote the number of sentences in the annotation W and N_s^i the number of words in the i-th sentence of the annotation W. Finally, we introduce $\overline{\mathbf{W}}$ as the concatenations of all the words w_j^i for each sentence $\mathbf{W}_i$ of the annotation $\mathbf{W}$

$$\begin{aligned} \overline{\mathbf{W}} &= \{\mathbf{W}_0 \mid \mathbf{W}_1 \mid \ldots \mid \mathbf{W}_{N_p}\} = \\ &= \{\overline{w}_0, \overline{w}_1, \ldots, \overline{w}_{\overline{N}}\}. \end{aligned} \tag{5.5}$$

with $\overline{N} = \sum_i N_s^i$.

Single-loop with pooling

The single-loop with pooling decoder is similar to the one proposed in Venugopalan et al. (2014). Starting from the matrix Y, we apply a 2D max-pooling operation to reduce both the temporal and the object dimension, compressing the information into a single context vector. The choice of the max-pooling operation, in contrast with Venugopalan et al. (2014), that was using an average-pooling one, is motivated by the results of Chapter 4, where the max-pooling is shown to be effective at tackling the unsafe maneuver task.

Specifically, starting from the matrix Y we apply a max-pooling operation over the first two dimensions. Then, we apply a fully-connected layer of size d^e followed by a ReLU activation and obtain a feature vector $\mathbf{y}$. We found the usage of this compression to be useful in our experiment. Then, iteratively, for each word $\overline{w}_j$ in the ground truth sentence $\overline{\mathbf{W}}$, we first perform a world embedding on the word $\overline{w}_j$ of size d^w, we concatenate such embedding to the context vector $\mathbf{y}$ and feed the results to the word LSTM, a single LSTM layer of size d^d

$$\mathbf{h}_j = LSTM^w\left(\mathbf{y} \mid \text{embedding}(\overline{w}_j), \mathbf{h}_{j-1}\right). \tag{5.6}$$

We finally feed h_j to a linear layer for the size of the vocabulary and train to predict the following $\overline{w}_{j+1}$ word with a standard cross entropy loss.

Single-loop with attention

This setup is similar to the one proposed in Yao et al. (2015), where the authors propose to leverage dot-product attention (Vaswani et al. (2017)) to dynamically build the context vector $\mathbf{y}$. This decoder architecture is identical to the single-loop with pooling one with the exception of what follows.

Starting from the tensor Y and the previous hidden state of the decoder LSTM h_{j-1} we compute the context vector $\mathbf{y}$ as the weighted sum

$$y = W_o\left(\sum_{t,i} \alpha_{t,i} v\left(\mathbf{y}_{t,i}\right)\right) \tag{5.7}$$

where

$$\alpha_{t,i} = softmax\left(k\left(\mathbf{y}_{t,i}\right)^\top q\left(\mathbf{h}_{j-1}\right)\right) \tag{5.8}$$

$$k(\mathbf{x}) = W_k\,\mathbf{x}, \quad q(\mathbf{x}) = W_q\,\mathbf{x}, \quad v(\mathbf{x}) = W_v\,\mathbf{x} \tag{5.9}$$

with $\sum_{i,t} \alpha_{t,i} = 1$, and with W_k, W_q, W_v, W_o linear projections of size d^a relative respectively to the key, query, value and output vectors.

Hierarchical with pooling

The hierarchical decoder is composed of two nested and asynchronous LSTM operations, the first one triggered at the beginning of each sentence of the annotation, the second one for each word, as in the single-loop decoders. The role of the sentence LSTM is to overlook the whole generation process by easily keeping track of which sentence has been predicted and to postpone the generation of the next sentence to the word LSTM, by generating its initial internal state $\mathbf{h}_0^w$.

Formally, starting from the matrix Y, the context vector $\mathbf{y}$ is generated as for the single-loop with pooling decoder. Then, for each sentence $\overline{\mathbf{W}}_i$ of $\mathbf{W}$, such vector is fed to the sentence LSTM

$$\mathbf{h}_t^s = LSTM^s\left(\mathbf{y}, \mathbf{h}_{t-1}^s\right) \tag{5.10}$$

and, for each word w_j^i of the sentence $\mathbf{W}_i$, we perform the same operation of Equation 5.6, with $\mathbf{h}_0^w = \mathbf{h}_t^s$.

Hierarchical with attention

The hierarchical decoder with attention decoder is similar to the one with pooling, but computes the context vector $\mathbf{y}$ for the word LSTM using dot-product attention, as described in the single-loop with attention decoder. As for the context vector of the sentence LSTM, we maintained the max-pooling operation over the Y tensor, as in the pooling decoders, as, intuitively, we want such part of the decoder to have a look over the whole event, *i.e.*, all the object at all timestamp, and not, potentially, to focus only on a portion of that, as well as try to keep the architecture simple.

Attributes for Safety-critical event captioning

Providing extra-attributes relative to the caption domain, if available, improves the overall result Yao et al. (2017); Pan et al. (2017). Such works used a set of attributes automatically learned from the annotations, in a Multiple Instance Learning (MIL) configuration. In particular, they consider the set of relevant words of size D_a in the annotation, for instance discarding articles and conjunctions, and the set of attributes $\{a_0, \ldots, a_{D_a}\}$ where each attribute a_j is positive for a give image I if the j-th relevant word is present in the image ground truth, negative otherwise. In this way, for instance, all the images acquired at night, that will likely have *night* in their ground truth caption, will have the night attribute as positive, while the daylights one will have it as negative.

While this approach has shown to be effective, we argue that in our context attributes with a deeper semantical meaning could be more effective. For instance, let us consider the attribute for *turn*. The presence of such attribute in a video would

indicate that a vehicle in the scene is turning, but does not indicate if it is the subject vehicle, another vehicle involved in the unsafe situation or a vehicle totally unrelated with the safety critical event. For this reason, we are boosting our captioning models with information that we know to be relevant in the specific domain.

In particular, we propose to use the knowledge of the safety-critical event being captioned to be a *crash* or *near-crash* event, and with the type of unsafe maneuver that is at the core of the event. We believe this to be beneficial in two ways: first, the model would learn to adjust the *a priori* probability of the words according to this information. For instance, the knowledge that a particular safety-critical event is a crash would increase the probability of words like *collides* and *hits* while lowering the one of *avoid* and *resume*; second, it should help to prevent severe errors, for instance, predicting a caption describing a near-crash for a video with a severe crash.

In practice, following the results of Yao et al. (2017), we are appending to the context vector of the word LSTM and of the sentence LSTM, when present, a flag indicating whether the safety-critical event is a crash or not, and a one-hot encoded vector of the unsafe maneuver in the event, among the 10 classes proposed in Chapter 2. We experimented both the usage of the ground truth of these attributes, to show their effectiveness, and the usage of the predictions from pre-trained models. In particular, we considered the probability scores of the best *MaxPool* model from Chapter 4 for the unsafe maneuver detection and the model from Kubin et al. (2021) as crash detector.

5.5 Experimental results

In this Section, we present our experimental results using the dataset introduced in Section 5.3, providing a broad ablation on the results we obtained. Also, we discuss in depth the experimental setup and the implementation choices under which such results are obtained.

Implementation details

All the experiments where conducted using the annotations of the newly introduced SHRP2-X dataset. For the encoder, we used the pre-trained weights from Chapter 4 on the unsafe maneuvers classification task and, thus, we are using the same data pre-processing of the video stream and sensor data, and the same way of extracting the detected objects. To limit the input size, we consider only the top $N_{objs} = 25$ objects with maximum volume, as shown effective in Section 4.3. Moreover, we performed experiments with two setups, specifically with and without the appearance features, *i.e.*, having $\mathbf{x}_{t,i} = \left[\mathbf{x}^g_{t,i} \mid \mathbf{x}^s_{t,i} \right]$. This second setup has shown to be valu-

able in terms of train and inference time while retaining good performance on the classification task.

In all the experiments, we are training the decoder with Adam optimizer with standard momentum parameters, starting from a learning rate of $5e-5$ and a weight decay of $1e-4$, and reducing the learning rate by a factor of 0.8 after 8 epochs of non-improvement on the validation loss. We used a batch size of $bs = 32$ for the *single-loop* decoders, while we used $bs = 1$ for the hierarchical decoders, following Wang et al. (2020), to cope with the asynchronous triggers of the sentence RNN that would occur, with a $bs > 1$, at different moment for different samples in the batch. As for the size of the dimension of the various components of the decoder, we performed multiple tests in the *single loop with pooling* setup and found the encoder bottleneck dimension $d^e = 64$, the LSTM dimension $d^d = 128$, the embedding dimension $d^w = 128$ and the attention dimension $d^a = 32$ to be the setup that yields the best results. Finally, to reduce the gap between the training and the inference, we are using Scheduled Sampling, presented in Bengio et al. (2015). Specifically, to train using the ground truth annotations and using the model own predictions at test time leads to a drop in performance due to the domain shift. The Scheduled Sampling strategy proposes to feed in the model own prediction during training with a probability p. Such probability is zero in the early epochs and then gradually increase up to a maximum probability $\overline{p}$. We are using $\overline{p} = 0.25$ in our experiments, starting the process with $p = 0.05$ after 50 epochs and increasing such values of 0.05 every 5 epochs for the *single-loop* setups, while starting with $p = 0.05$ after 25 epochs and increasing such values of 0.05 every 3 epochs for the hierarchical setups, as we observed a faster convergence rate.

For the quantitative evaluation of the results, we are using common metrics used in image/video captioning tasks: BLEU (Papineni et al. (2002)), ROUGE (Lin (2004)) and METEOR (Banerjee and Lavie (2005)). We are not reporting results using other popular metrics as they require multiple annotations per sample, while we have just a single one available.

Ablation study

All the results are reported in Table 5.3. We are using *RNN-pool*, *RNN-attention*, *HRNN-pool* and *HRNN-attention* to indicate respectively single-loop with pooling, single-loop with attention, hierarchical with pooling and hierarchical with attention decoders. In the first row, we are reporting the results of the single-loop with pooling architecture trained with and without the proposed attributes. We can observe an increase of roughly 0.03 points in both BLEU-4 and METEOR by adding the ground truth attributes of the crash and unsafe maneuvers (C^* and M^* respectively in the Table). An increase could be observed also by adding the attributes singularly: by adding the unsafe maneuver ground truth, we observe a boost of 0.03

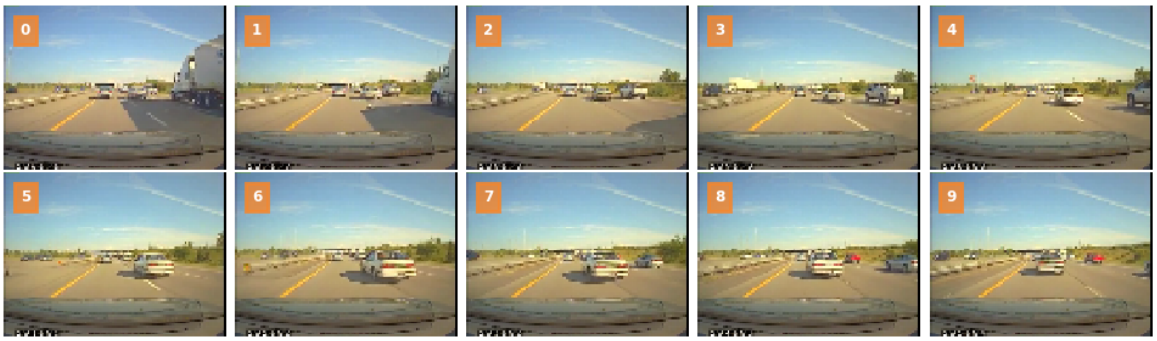
Table 5.3: Results of the proposed approaches. The features column indicates the type of features used, A (Appearance), G (Geometry), S (Sensors). The attributes column indicates the attributes fed to the context vector, M (unsafe maneuver predictions), M^* (unsafe maneuvers ground truth), C (crash predictions), C^* (crash ground truth). The SS column indicates the presence or absence of Scheduled Sampling (Bengio et al. (2015))

Model	Features	Attributes	SS	B@1	B@2	B@3	B@4	R@1	R@2	R@L	METEOR
RNN-pool	G, S			0.6363	0.5168	0.4266	0.3569	0.5970	0.3587	0.6089	0.4290
RNN-pool	G, S	M		0.6336	0.5175	0.4286	0.3589	0.5972	0.3627	0.6063	0.4366
RNN-pool	G, S	C		0.6333	0.5169	0.4272	0.3565	0.5966	0.3589	0.6044	0.4318
RNN-pool	G, S	M, C		0.6438	0.5245	0.4332	0.3619	0.6112	0.3695	0.6167	0.4410
RNN-pool	G, S	M^*		0.6469	0.5386	0.4528	0.3850	0.6162	0.3924	0.6244	0.4586
RNN-pool	G, S	C^*		0.6394	0.5209	0.4303	0.3598	0.5994	0.3639	0.6084	0.4380
RNN-pool	G, S	M^*, C^*		0.6469	0.5390	0.4536	0.3848	0.6157	0.3918	0.6300	0.4588
RNN-pool	G, S	M^*, C^*	✓	**0.6570**	**0.5475**	**0.4604**	**0.3906**	**0.6283**	**0.4018**	**0.6353**	0.4646
RNN-attention	G, S	M^*, C^*	✓	0.6413	0.5364	0.4521	0.3850	0.6123	0.3930	0.6321	0.4615
HRNN-pool	G, S	M^*, C^*	✓	0.6287	0.5267	0.4447	0.3786	0.6066	0.3931	0.6248	0.4621
HRNN-attention	G, S	M^*, C^*	✓	0.6546	0.5459	0.4593	0.3905	0.6229	0.3993	0.6320	**0.4670**
RNN-pool	G	M^*, C^*	✓	0.6273	0.5232	0.4408	0.3761	0.6271	0.4001	0.6376	0.4595
RNN-pool	A, G, S	M^*, C^*	✓	0.6507	0.5425	0.4555	0.3857	0.6174	0.3905	0.6286	0.4608
RNN-attention	A, G, S	M^*, C^*	✓	0.6560	0.5461	0.4593	0.3898	0.6252	0.3955	**0.6295**	0.4674
HRNN-pool	A, G, S	M^*, C^*	✓	0.6425	0.5381	0.4545	0.3877	0.6162	0.3975	0.6143	0.4641
HRNN-attention	A, G, S	M^*, C^*	✓	**0.6621**	**0.5512**	**0.4631**	**0.3928**	**0.6283**	**0.4007**	0.6273	**0.4685**

in both BLEU-4 and METEOR, while by adding the crash ground truth, we observe a boost of 0.003 in BLEU-4 and 0.009 on the METEOR score. The gap in the contribution of the two attributes has to be weighted on their occurrence in the data, in fact, the crashes are just a small portion of the totality of the events. If we provide, instead, the prediction of two classifiers trained to detect crashes and unsafe maneuvers types (C and M respectively in the Table), we observe a similar behaviour but with a smaller increase, as by adding both attributes, we have a boost of 0.005 in BLEU-4 and 0.012 in METEOR. Moreover, by comparing the last entry of the first row and the first entry of the second row, we can observe the impact of the Scheduled Sampling strategy. When activating the Scheduled Sampling, we observed a drop in the training and validation loss, due to the uncertainty introduced by the model own predictions, that translates, however, in a boost of the performance on the test set of 0.005 in BLEU-4 and 0.006 in METEOR. Although, for simplicity, we reported this results only on the RNN-pool setup, we empirically observed a similar effect on all the proposed decoders. In the second row, the results of the different proposed decoders are reported. The setup with the best performance is RNN-pool, that held the best results in terms of BLEU and ROGUE. It is worth to highlight, however, that the results of the HRNN-attention model are extremely close, *e.g.*, the difference in BLEU-4 score is just 0.0001, and this model yield the best results in METEOR score. In the third row, we highlighted the contribution of the sensor features to the final caption, and run an experiment with just the geometric features. We can observe a substantial drop of 0.015 in BLEU-4 and 0.005 in METEOR, suggesting that such features are useful to the generation process, likely for the events in which the ego-vehicle hits some other entities (*e.g.*, objects, road bumps) or for the loss of control events. In the fourth row, instead, we can observe the effect of adding the appearance features. First, we can observe how HRNN-attention is yielding the best results overall. However, the gap with the appearance-less counterpart is not massive, it is just 0.002 in terms of BLEU-4 and 0.0015 in METEOR. This results somehow suggests that the appearance of the objects are not crucial to the task, and that the positions and dynamic of the objects in the scene in conjunction with the features from the GPS/IMU sensors are sufficient for the generation of this type of captions. Also, it is interesting how the attention-based model are outperforming the pooling one in this setup. We believe that, since the features generated with the appearance are more complex, attention can actually play a role and improve the results over a simple, yet effective, feature pooling.

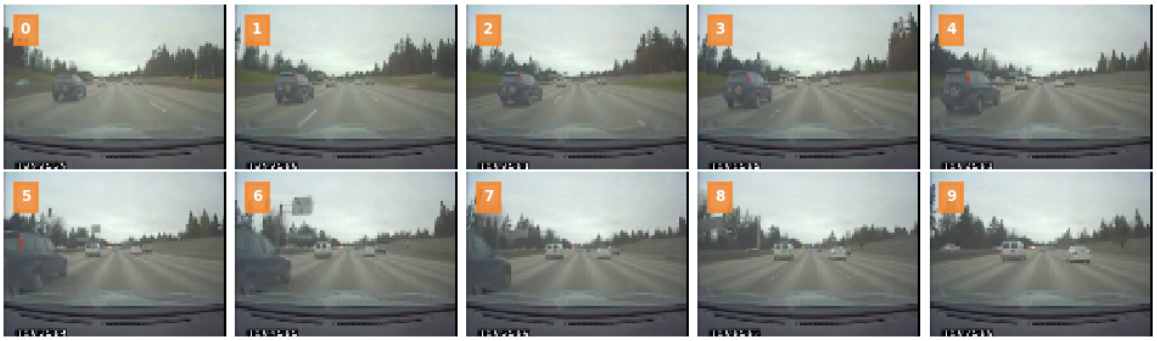
Results visualization

In this Section, we report some of the obtained captions along with the original ground truth, for a qualitative evaluation of the results. The images have been



Method	Caption
Ground truth	V2 is ahead in the right adjacent lane. V2 begins to change lanes to the left. SV has to brakes hard to avoid a collision with V2. Both vehicle continue traveling in now shared lane.
RNN-pool	V2 is ahead in the adjacent right lane. V2 begins to change lanes to the left into SV's lane. SV brakes hard to avoid a rear-end collision with V2. V2 steers right back into its original lane. SV continues on.
RNN-attention	V2 is ahead in the adjacent right lane. V2 begins to change lanes to the left into SV's lane. SV brakes hard to avoid a rear-end collision with V2. V2 continues on.
HRNN-pool	V2 is ahead in the adjacent right lane. V2 change lanes to the left into SV's lane. SV brakes hard to avoid a rear-end collision with V2. V2 completes the lane change. SV continues on.
HRNN-attention	V2 is ahead in the adjacent right lane. V2 begins to change lanes to the left into SV's lane. SV brakes hard to avoid a rear-end collision with V2. Both vehicles continue on.

Figure 5.3: Visualization of the results for an event with an unsafe lane change.



Method	Caption
Ground truth	V2 is in the left adjacent lane. V2 begins to change lanes right ahead of SV. SV must brake. SV steers right to avoid sideswiping V2. V2 steers left back into its original lane. SV continues on.
RNN-pool	V2 is ahead in the adjacent left lane. V2 begins to change lanes to the left into SV's lane. SV brakes hard to avoid a collision with V2. V2 steers right back into its original lane. SV continues on.
RNN-attention	V2 is ahead in the adjacent left lane. V2 begins to change lanes to the left into SV's lane. SV brakes to avoid a collision with v2. V2 steers right back into its original lane. SV continues on.
HRNN-pool	V2 is ahead in the adjacent right lane. V2 begins to change lanes to the left into SV's lane. SV brakes hard to avoid a collision with V2. V2 steers right back into its original lane. Both vehicles continue on.
HRNN-attention	V2 is ahead in the adjacent right lane. V2 begins to change lanes to the left into SV's lane. SV brakes to avoid a collision with V2. V2 steers right back into its original lane. Both vehicles continue on.

Figure 5.4: Visualization of the results for an event with an unsafe lane change.

Method	Caption
Ground truth	SV is stopped at an intersection. SV begins to make a left turn. V2 approaches the intersection from the right. V2 turns left across SV's path. SV brakes hard to avoid hitting V2. SV completes the left turn. SV continues down the roadway.
RNN-pool	SV is at an intersection controlled by a stop sign. SV begins to turn left. V2 approaches the intersection from the right. V2 begins to turn left across SV's path. SV brakes hard to avoid a collision with V2. V2 brakes. V2 steers left. SV continues on.
RNN-attention	SV is at an intersection. SV begins to turn left. V2 approaches the intersection from the right. SV brakes hard to avoid a collision with V2. V2 brakes. V2 steers left.
HRNN-pool	SV is approaching an intersection controlled by a stop sign. SV begins to turn left. V2 approaches the intersection from the right. V2 begins to turn left. SV brakes hard to avoid a collision with V2. V2 brakes. SV completes the turn. SV continues on.
HRNN-attention	SV is at an intersection controlled by a stop sign. SV is stopped. V2 approaches the intersection from the right. SV begins to turn left. SV brakes to avoid a collision with V2. V2 brakes. SV completes the turn.

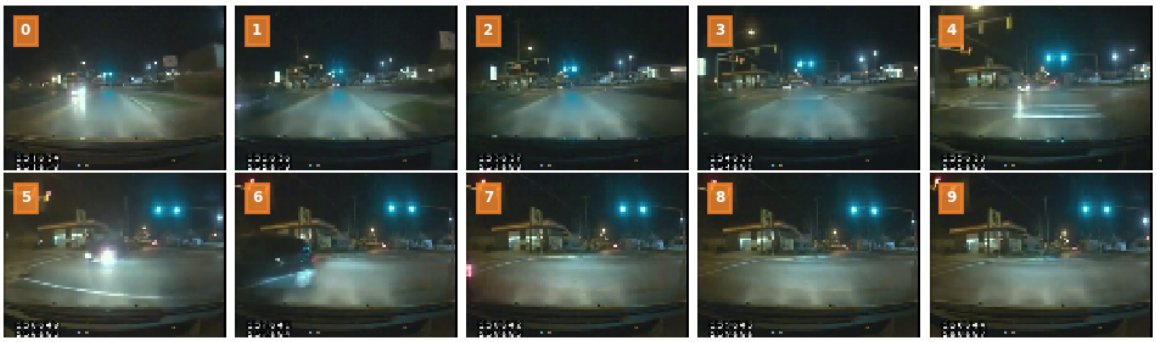
Figure 5.5: Visualization of the results for an event with an unsafe turn.

anonymized. In all the Figures, we are highlighting in yellow the errors that we believe to be mild, while in red the errors that we believe to be severe.

In Figure 5.3 we are reporting the captions of a non-subject lane change event. All the proposed decoders are correctly generating the caption, first describing the vehicle on the right, then identifying the lane change event and recognizing the need of the subject vehicle to brake. All the decoders correctly learned the language used in the training set, generating sentences coherent with the output domain.

In Figure 5.4, instead, we are reporting a similar event in which the generation fails. While, again, the output is coherent with the domain, the position of the vehicle, the direction of the lane change and the direction of the evasive maneuver are swapped, *i.e.*, the word *left* and *right* are confused. We observed this behaviour quite often (roughly, 30-40% of the narratives) and it is one of the major source of errors performed by the model.

Figure 5.5 shows the generated captions of a turn event performed by the subject. The subject vehicle is invading the intersection, risking a collision with another vehicle turning. While the core description of the event of all the four decoders are correct, some of them hallucinate things that are not in the scene, specifically, the presence of a stop sign or the action of the subject vehicle to continue on the roadway. This is likely due to the presence in the training set of such sentence in similar



Method	Caption
Ground truth	SV is approaching a signaled intersection. SV prepares to turn left. V2 approaches the intersection from the opposite direction. V2 goes straight at the intersection. SV begins to turn left. SV has to brake hard. SV steers to the right to avoid a collision with V2. V2 brakes. V2 steers right. V2 continues driving. SV completes the turn.
RNN-pool	SV approaches an intersection. SV is stopped. SV begins to turn left. V2 approaches the intersection from the right. V2 begins to turn left across SV's path. SV brakes hard to avoid a collision with V2. V2 brakes. SV continues straight. SV continues on.
RNN-attention	SV is approaching a signalized intersection. SV is stopped. SV begins to turn left. V2 approaches the intersection from the right. SV brakes hard to avoid a collision with V2. V2 brakes. SV continues on.
HRNN-pool	SV approaches a signalized intersection. SV is stopped. V2 is the the oncoming lane. SV begins to turn left. SV brakes hard to avoid a collision with V2. V2 completes the turn. SV continues on.
HRNN-attention	SV approaches a signalized intersection. SV is stopped at the intersection. SV begins to turn left. V2 is the intersection from the right. SV brakes hard to avoid a collision with V2.

Figure 5.6: Visualization of the results for an event with an unsafe turn.

Method	Caption
Ground truth	SV approaches an intersection controlled by a stop sign. SV loses control of the vehicle. SV begins to slides forward. SV brakes. SV regains control of the vehicle. SV stops.
RNN-pool	SV approaches an intersection. SV begins to turn left. SV loses control of the vehicle. SV begins to slide to the right. SV steers right. SV steers right. SV continues on.
RNN-attention	SV approaches a signalized intersection. SV begins to turn left. SV loses control of the vehicle. SV begins to slide to the right. SV hits the curb. SV continues on.
HRNN-pool	SV approaches a signalized intersection. SV begins to turn left. SV loses control of the vehicle. SV begins to slide to the left. SV hits the curb with the right.
HRNN-attention	SV approaches a signalized intersection. SV begins to turn left. SV loses control of the vehicle. SV begins to slide forward. SV hits the curb. SV hits an object.

Figure 5.7: Visualization of the results for an event with a loss of control.

condition of such sentences, *e.g.*, at night, that induces the decoders to produce such sentences.

In Figure 5.6, we observe a vehicle going straight at an intersection an provoking a near-collision. Due to the structure of the road, the vehicle goes from the central part of the image to the left, and induce the decoder, with the exception of HRNN-pool, to think that the vehicle is approaching from the right, instead of the oncoming lane. Finally, in Figure 5.7, we report the results of a loss of control event. All the decoder types are correctly generating a sentence in which the driver loses control of the vehicle, however, they mistake the sliding direction of the vehicle, with the exception of HRNN-attention.

To summarize, in all the presented samples, the generated captions of all the proposed decoders are similar, and all the decoders makes similar mistakes. However, as highlighted by the experimental results, on average some decoders make fewer mistakes than others.

5.6 Conclusions

We tackled the novel problem of safety-critical event captioning, starting from the data acquired from a dashcam installed inside the vehicle and sensors data from a GPS/IMU module. To do so, we fist generated, and released to the scientific community, the SHRP2-X dataset, a collection or roughly 3000 annotations of the SHRP dataset videos. We proposed to tackle the problem by using an encoder-decoder architecture. We used an encoder pre-trained on the unsafe maneuver classification task, that incorporates the output of an object detector on each frame of the video and a tracking algorithm, to extract features on the evolution of the objects of the scene over time, as well as performing the fusion of the sensor features. We tested four decoders from the video captioning literature, using a hierarchical or single-loop architecture and using attention or a simple pooling operation over the features from the encoder. Furthermore, we proposed the usage of attributes from the safety-critical domain, specifically, the presence or absence of a crash and the unsafe maneuver type, to boost the captioning performance. Our experiments showed that the usage of such attributes can give a significant boost in the overall generated captions. Moreover, we showed that the sensors data, if available, significantly improves the results of the models that performs learning on tasks involving safety-critical event data, confirming the findings of Chapter 3 and Chapter 4. In contrast, the appearance feature seems less important for the captioning generation: we believe they provide the same information of features on the detected object in the scene and their position and motion. Finally, of the four proposed architecture, we showed that the hierarchical decoder with attention outperforms the remaining three, confirming the findings of Yu et al. (2016); Wang et al. (2020), but with

by a significantly smaller margin.

Chapter 6

Conclusions and future work

The goal of this thesis was to tackle the analysis of safety-critical driving events recorded from a dashcam mounted inside a vehicle. This was done by introducing novel deep architectures tailored to address the classification and the detection of unsafety in the driving scene and to provide human-understandable explanations. In this Chapter, we summarize the key findings of this thesis and we propose future research directions.

6.1 Conclusions

In this thesis, we studied a novel approach for the analysis of safety-critical events, with the goal of pushing forward the research on road scene understanding and ultimately of increasing road safety. To this end, in Chapter 2 we introduced the novel unsafe maneuver classification task. In contrast with other tasks in the literature, focussing just on the result of the unsafe situation, as *crash detection*, or not taking into consideration the unsafety at all, as *maneuver classification*, this problem aims at classifying both crashes and near-crashes based on the maneuver that lead to the dangerous situation.

In Chapter 3 we introduced a novel two-stream architecture to tackle the unsafe maneuver classification problem, starting from the dashcam and GPS/IMU sensors data. The proposed architecture both addressed the alignment between the two heterogeneous streams, with different sampling rates and sampling timestamps, as well as addressing the classification task. Our experiments showed the importance of using both streams, if available, to tackle the task, as by removing either of the two the performance drops significantly. Also, they highlighted how this architecture was capable of identifying the temporal interval in which the unsafe event occurred, correctly predicting the unsafe maneuver class even if a long portion of safe driving was included in the samples. Finally, we proposed a backbone fine-tuning strategy tailored to the unsafe maneuver classification task, using the same labels on short

snippets of 32 frames at a lower frame rate, that greatly contributed to improve the overall results.

In Chapter 4 we extended the two-stream architecture in three ways. First, we included the output of an object detector to provide the network with explicit information on the entities on the road. Second, we proposed an alternative way of extracting sensor features, leveraging depth-wise separable convolutions. Third, we designed the new STAS module, that extracts features representing the evolution of a single real-world object over a given time span by using a tracking algorithm, and then uses a multi-head dot-product attention module to select only the relevant objects in the scene. Moreover, we empirically showed that by looking at the attention weights on each object, we can tell which one among the objects in the scene was addressed by the network as relevant, achieving a form of explainability. As an alternative and in contrast, we proposed a simple pooling operation over the extracted object features. This *opaque* version of the network showed superior results, suggesting that explainability comes with a cost. All the proposed novelties, combined, contributed to a new *state-of-the-art* on the unsafe maneuver classification task. Finally, our experiment showed how removing the features extracted from the sensors or the video streams results in a drop on the overall performance. While the first finding confirms the results on the importance of this type of feature for this task, the second one opens to new applicative scenarios in which the proposed architecture is a lightweight component plugged in a pipeline that already involves object detection.

In Chapter 5 we further explored explainability by addressing captioning of safety-critical driving events. We used an encoder-decoder architecture, with as encoder the same STAS architecture pre-trained on the unsafe maneuver classification task. As decoder, we tested multiple decoders from the literature. Our experiments highlighted how an hierarchical decoder leveraging dot-product attention showed superior results compared to the other approaches. The experimental phase also confirmed the importance of the GPS/IMU data to the generation of the caption while showed the appearance feature to be less crucial, suggesting the notion of the objects in the scene and the sensor data relative to the subject vehicle could be sufficient to tackle this type of captioning problem. Furthermore, we showed that providing the network with semantic attributes, specifically the unsafe maneuver type and the presence or absence of a crash, improves the overall result. Finally, as part of our contribution, we labelled and released to the scientific community the SHRP2-X dataset, a collection of multi-sentence captions on top of the SHRP2 NDS dataset.

6.2 Future works

Unsafe maneuver classification is far from being a solved problem. As future line of research, we plan to keep tackling such task by first trying to move to a bigger dataset, as we have the feeling that it would be hard to significantly improve the results on the minority classes without a larger sample size, that might help the network to cover the rare scenarios. To this end, we plan to use the Verizon Connect video base, a collection of more than 65 millions videos acquired across the US between 2018 and 2021, comprehensive of crashes, near-crashes, harsh-driving and safe events. Moreover, we plan to incorporate other information from the scene and, in particular, we believe that the information on the lanes and their separating line could help disambiguate corner case situations. Finally, we plan to extend the proposed architecture to other related tasks, that would allow for a growing understanding of what is happening on the road scene. In particular, the finding of this thesis could be used to address *unsafe maneuver detection*, *i.e.,* the task to not just classify but also localize in time and space the maneuver leading to the dangerous situation, and generalize that to *maneuver detection*, *i.e.,* detecting all the maneuver in the scene, but distinguishing the safe from the unsafe ones. To this end, it is necessary to design an architecture that exploits the relations between the detected vehicles and, possibly, the detected lanes in the scene, for instance using a Transformer-based architecture, as we believe such information is missing and could be useful to have a better understanding of the road scene.

We think that these improvements would benefit also explainability in the captioning task, as the maneuver performed by others and not strictly connected to the dangerous situation are part of the description. Moreover, we can try to reduce the gap between the two problems, addressing the actor identification and the captioning prediction in one shot, as done in the dense captioning and grounded captioning literature. To this end, though, the annotations of more data or the usage of a bigger dataset might be required.

Finally, we plan to verify the hypothesis that this type of models could be used to detect dangerous situations. To this end, we could think of equipping the vehicles with a dedicated hardware capable to run inference on a deep learning model and use them along with or in replacement of the kinematics triggers to detect a broader and more precise set of unsafe situation on the road, again used to coach the drivers and hopefully ultimately leading to a reduction of deaths and injuries on the roads.

Appendix A

Publications

Journal papers

1. L. Bravi, L. Kubin, S. Caprasecca, D. Coimbra de Andrade, **Matteo Simoncini**, L. Taccari, L. Sarti, F. Sambo, "Detection of Stop Sign Violations From Dashcam Data", *IEEE Transactions on Intelligent Transportation Systems*, 2021. **Candidate's contributions**: contributed to the definition of stop violation, contributed to design the algorithm, contributed to the final version of the manuscript

2. L. Kubin, T. Bianconcini, D. Coimbra de Andrade, **Matteo Simoncini**, L. Taccari, F. Sambo, "Deep Crash Detection from Vehicular Sensor Data with Multimodal Self-Supervision", *IEEE Transactions on Intelligent Transportation Systems*, 2021. **Candidate's contributions**: contributed to design the algorithm, contributed to the final version of the manuscript

3. **Matteo Simoncini**, D. Coimbra de Andrade, L. Taccari, S. Salti, L. Kubin, F. Schoen, F. Sambo, "Unsafe maneuver classification from dashcam video and GPS/IMU sensors using Spatio-Temporal Attention Selector", *IEEE Transactions on Intelligent Transportation Systems*, 2022. **Candidate's contributions**: designed the algorithm, designed and carried out the experiments, wrote the manuscript

Peer reviewed conference papers

1. L. Bravi, **Matteo Simoncini**, L. Taccari, L. Sarti, S. Caprasecca, A. Benericetti, A. Lori, S. Salti, F. Sambo, "Fault detection in non-reporting Vehicle Tracking Units", *2019 IEEE 22nd Intelligent Transportation Systems Conference (ITSC)*, pages:2791–2796, 2019. **Candidate's contributions**: designed the algorithm, contributed to the design of the experiments, contributed to the final version of the manuscript

2. A. Pjetri, **Matteo Simoncini**, F. Sambo, A. Lori, F. Schoen, "Light Footprint Driving Behaviour Classification", *2019 IEEE 22nd Intelligent Transportation Systems Conference* (*ITSC*), pages:1186–1191, 2019. **Candidate's contributions**: contributed to supervise the project, contributed to the final version of the manuscript

3. **Matteo Simoncini**, D. Coimbra de Andrade, S. Salti, L. Taccari, F. Schoen, F. Sambo, "Two-stream neural architecture for unsafe maneuvers classification from dashcam videos and GPS/IMU sensors", *2020 IEEE 23rd International Conference on Intelligent Transportation Systems* (*ITSC*), pages:1–6, 2020. **Candidate's contributions**: designed the algorithm, designed and carried out the experiments, wrote the manuscript

4. S. Magistri, F. Sambo, F. Schoen, D. Coimbra de Andrade, **Matteo Simoncini**, S. Caprasecca, L. Kubin, L. Bravi, L. Taccari, "A Lightweight Deep Learning Model for Vehicle Viewpoint Estimation from Dashcam Images", *2020 IEEE 23rd International Conference on Intelligent Transportation Systems* (*ITSC*), pages:1–6, 2020. **Candidate's contributions**: contributed to design the algorithm, helped to supervise the project, contributed to the final version of the manuscript

Papers under review

1. **Matteo Simoncini**, N. Bellaccini, D. Coimbra de Andrade, L. Kubin, S. Salti, L. Taccari, L. Sarti, F. Schoen, F. Sambo, "Video captioning of safety-critical events from dashcam data", *IEEE Transactions on Intelligent Transportation Systems*, 2022. **Candidate's contributions**: designed the algorithm, contributed to the annotation of the data, designed and carried out the experiments, wrote the manuscript

2. S. Magistri, M. Boschi, F. Sambo, D. Coimbra de Andrade, **Matteo Simoncini**, L. Kubin, L. Taccari, L. De Luigi, S. Salti, "Lightweight and Effective Convolutional Neural Networks for Vehicle Viewpoint Estimation from Monocular Images", *IEEE Transactions on Intelligent Transportation Systems*, 2022. **Candidate's contributions**: contributed to the final version of the manuscript

Bibliography

Abdul, A., Vermeulen, J., Wang, D., Lim, B. Y., and Kankanhalli, M. (2018). Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–18.

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., and Zhang, L. (2018a). Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086.

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., and Zhang, L. (2018b). Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Banerjee, S. and Lavie, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. *arXiv preprint arXiv:1506.03099*.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13.

Bradbury, J., Merity, S., Xiong, C., and Socher, R. (2016). Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*.

Breuer, A., Kirschner, J., Homoceanu, S., and Fingscheidt, T. (2019). Towards tactical maneuver detection for autonomous driving based on vision only. In *Intelligent Vehicles Symposium*, pages 941–948. IEEE.

Carreira, J. and Zisserman, A. (2017). Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308.

Chan, F.-H., Chen, Y.-T., Xiang, Y., and Sun, M. (2016). Anticipating accidents in dashcam videos. In *Asian Conference on Computer Vision*, pages 136–153. Springer.

Chen, S., Yao, T., and Jiang, Y.-G. (2019). Deep learning for video captioning: A review.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.

Cultrera, L., Seidenari, L., Becattini, F., Pala, P., and Del Bimbo, A. (2020). Explaining autonomous driving by learning end-to-end visual attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 340–341.

de Andrade, D. C., Leo, S., Viana, M. L. D. S., and Bernkopf, C. (2018). A neural attention model for speech command recognition. *arXiv preprint arXiv:1808.08929*.

Deo, N., Rangesh, A., and Trivedi, M. M. (2018). How would surround vehicles move? A unified framework for maneuver classification and motion prediction. *IEEE Transactions on Intelligent Vehicles*, 3(2):129–140.

Dozza, M. and Gonzalez, N. P.-e. (2012). Recognizing safety-critical events from naturalistic driving data. *Procedia - Social and Behavioral Sciences*, 48:505–515.

Fang, J., Yan, D., Qiao, J., and Xue, J. (2019a). DADA: A large-scale benchmark and model for driver attention prediction in accidental scenarios. *arXiv preprint arXiv:1912.12148*.

Fang, J., Yan, D., Qiao, J., Xue, J., Wang, H., and Li, S. (2019b). DADA-2000: Can driving accident be predicted by driver attention? analyzed by a benchmark. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 4303–4309. IEEE.

Feichtenhofer, C., Fan, H., Malik, J., and He, K. (2019). Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211.

Feichtenhofer, C., Pinz, A., and Wildes, R. P. (2017). Spatiotemporal multiplier networks for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4768–4777.

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., and Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Gunning, D. (2017). Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2:2.

Hankey, J. M., Perez, M. A., and McClafferty, J. A. (2016a). Description of the SHRP 2 naturalistic database and the crash, near-crash, and baseline data sets. Technical report, Virginia Tech Transportation Institute.

Hankey, J. M., Perez, M. A., and McClafferty, J. A. (2016b). Description of the SHRP 2 naturalistic database and the crash, near-crash, and baseline data sets. Technical report, Virginia Tech Transportation Institute.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Herdade, S., Kappeler, A., Boakye, K., and Soares, J. (2019a). Image captioning: Transforming objects into words. In *Advances in Neural Information Processing Systems*, pages 11137–11147.

Herdade, S., Kappeler, A., Boakye, K., and Soares, J. (2019b). Image captioning: Transforming objects into words. *arXiv preprint arXiv:1906.05963*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

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

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

Kim, J., Rohrbach, A., Darrell, T., Canny, J., and Akata, Z. (2018). Textual explanations for self-driving vehicles. In *Proceedings of the European conference on computer vision (ECCV)*, pages 563–578.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krishna, R., Hata, K., Ren, F., Fei-Fei, L., and Carlos Niebles, J. (2017). Dense-captioning events in videos. In *Proceedings of the IEEE international conference on computer vision*, pages 706–715.

Kubin, L., Bianconcini, T., Coimbra de Andrade, D., Simoncini, M., Taccari, L., and Sambo, F. (2021). Deep crash detection from vehicular sensor data with multimodal self-supervision. *IEEE Transactions on Intelligent Transportation Systems*. To appear.

Lei, J., Yu, L., Berg, T. L., and Bansal, M. (2019). Tvqa+: Spatio-temporal grounding for video question answering. *arXiv preprint arXiv:1904.11574*.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Lu, J., Yang, J., Batra, D., and Parikh, D. (2018). Neural baby talk. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7219–7228.

Müller, M. (2007). Dynamic time warping. *Information retrieval for music and motion*, pages 69–84.

Palazzi, A., Abati, D., Solera, F., Cucchiara, R., et al. (2018). Predicting the Driver's Focus of Attention: the DR (eye) VE Project. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(7):1720–1733.

Pan, Y., Mei, T., Yao, T., Li, H., and Rui, Y. (2016). Jointly modeling embedding and translation to bridge video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4594–4602.

Pan, Y., Yao, T., Li, H., and Mei, T. (2017). Video captioning with transferred semantic attributes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6504–6512.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peng, X., Liu, R., Murphey, Y. L., Stent, S., and Li, Y. (2018). Driving maneuver detection via sequence learning from vehicle signals and video images. In *2018 International Conference on Pattern Recognition (ICPR)*, pages 1265–1270. IEEE.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Rosenfeld, A. and Richardson, A. (2019). Explainability in human–agent systems. *Autonomous Agents and Multi-Agent Systems*, 33(6):673–705.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Suzuki, T., Kataoka, H., Aoki, Y., and Satoh, Y. (2018). Anticipating traffic accidents with adaptive loss and large-scale incident db. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3521–3529.

Taccari, L., Sambo, F., Bravi, L., Salti, S., Sarti, L., Simoncini, M., and Lori, A. (2018). Classification of crash and near-crash events from dashcam videos and telematics. In *2018 21st International Conference on Intelligent Transportation Systems*, pages 2460–2465. IEEE.

Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497.

Tran, D., Wang, H., Torresani, L., and Feiszli, M. (2019). Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5552–5561.

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., and Paluri, M. (2018). A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., and Saenko, K. (2015). Sequence to sequence-video to text. In *Proceedings of the IEEE international conference on computer vision*, pages 4534–4542.

Venugopalan, S., Xu, H., Donahue, J., Rohrbach, M., Mooney, R., and Saenko, K. (2014). Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*.

Wang, T., Zheng, H., and Yu, M. (2020). Dense-captioning events in videos: Sysu submission to activitynet challenge 2020. *arXiv preprint arXiv:2006.11693*.

Woo, S., Park, J., Lee, J.-Y., and So Kweon, I. (2018). Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

Xia, Y., Zhang, D., Kim, J., Nakayama, K., Zipser, K., and Whitney, D. (2018). Predicting driver attention in critical situations. In *Asian Conference on Computer Vision*, pages 658–674. Springer.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Yao, L., Torabi, A., Cho, K., Ballas, N., Pal, C., Larochelle, H., and Courville, A. (2015). Describing videos by exploiting temporal structure. In *Proceedings of the IEEE international conference on computer vision*, pages 4507–4515.

Yao, T., Pan, Y., Li, Y., Qiu, Z., and Mei, T. (2017). Boosting image captioning with attributes. In *Proceedings of the IEEE international conference on computer vision*, pages 4894–4902.

Yao, Y., Xu, M., Wang, Y., Crandall, D. J., and Atkins, E. M. (2019). Unsupervised traffic accident detection in first-person videos. *arXiv preprint arXiv:1903.00618*.

Yu, H., Wang, J., Huang, Z., Yang, Y., and Xu, W. (2016). Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4584–4593.

Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., and Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702.

Zekany, S. A., Dreslinski, R. G., and Wenisch, T. F. (2019). Classifying ego-vehicle road maneuvers from dashcam video. In *2019 IEEE Intelligent Transportation Systems Conference*, pages 1204–1210. IEEE.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. (2021). Scaling vision transformers. *arXiv preprint arXiv:2106.04560*.

Zhang, Y., Suda, N., Lai, L., and Chandra, V. (2017). Hello edge: Keyword spotting on microcontrollers. *arXiv preprint arXiv:1711.07128*.

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. (2017). Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhou, L., Kalantidis, Y., Chen, X., Corso, J. J., and Rohrbach, M. (2019). Grounded video description. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6578–6587.

Zhu, R., Fang, J., Xu, H., and Xue, J. (2019). Progressive temporal-spatial-semantic analysis of driving anomaly detection and recounting. *Sensors*, 19(23):5098.