RAM ROBOTICS AND MECHATRONICS

STUDY OF SEMANTIC SEGMENTATION APPLICATIONS FOR AUTONOMOUS VEHICLES

S. (Santiago) Sanchez Escalonilla Plaza

MSC ASSIGNMENT

Committee:
dr. B. SirmaÃğek
N. Botteghi, MSc
dr. ir. M. Abayazid
dr. ir. F. Heijden
dr. M. Poel

September, 2019

Robotics and Mechatronics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Summary

Autonomous Vehicles are machines capable of navigating the environment without human intervention. Such an interaction with the environment requires of a system that provides the vehicle with accurate information about the surroundings, this is called scene understanding. Scene understanding includes obtention, processing and analysis of data. There are different ways of obtaining information from the environment although the most common one is through the use of cameras. Cameras can obtain visual information of the surroundings in the same way humans do. There are different techniques that allow the user to learn from its composition depending on the final goal and the required level of accuracy. Some of the applications of these techniques are: image classification, object detection, object tracking or semantic image segmentation. Semantic image segmentation provides insight about the composition of the image in the highest possible detail. It consists on the pixel-label classification of the image. Semantic image segmentation can be very useful for navigation scenarios, allowing to create an accurate representation of the environment.

Classical Computer Vision provides the necessary tools to achieve any of the previously mentioned applications, however these tools are very rigid and limited against variations of external and internal parameters (illumination, occlusion, depth, camera resolution, disturbances or noise). Using Deep Learning for computer vision applications has proved to be key for applications working in dynamic environment conditions. There are different Deep Learning models designed to achieve semantic image segmentation ([1], [2], [3]). Unfortunately, the segmentation obtained after applying these methods differ from the ideal expected case (figure 1.3). In the best of the cases obtaining a partial object segmentation but in the worst completely missing the target.

There are two different approaches valid for the semantic video segmentation. The first one consists on the direct application of semantic image segmentation models in a frame-by-frame basis. However, this approach often leads to inconsistent results and the appearance of a 'flickery' effect due to the frames rapidly changing conditions. The second approach is to design specific tools for the analysis of videos, extracting the temporal context and using it for the current frame segmentation. This thesis focus on the last one.

To do so, this thesis has defined three particular research questions. What is the current state-of-the-art for semantic image segmentation? How to extend semantic-image-segmentation models for the analysis of sequences? and What kind of mechanicsms can be applied to reduce the number of false classifications?

The result for the first research question selected DeepLabv3 [4] (pretrained on Cityscapes [5]) as the state-of-the-art and the baseline model for this study due to its high accuracy on urban-scenarios. The last two research questions are answered together on the design of an extension algorithm that can be applied to semantic image segmentation models for the analysis of sequences. Two different approaches were the outcome of this study, an approach that combines the baseline segmentation of neighbouring frames (Image Buffer) and a different approach that modifies the segmentation probabilities and afterwards establishes a relation between frames (Attention Module). Later, each method was evaluated using two metrics chosen to determine the temporal consistency of the segmentation.

As a result, both of the suggested extensions improve the consistency of the segmentation overtime (chapter 5), in some cases helping on the segmentation of objects difficult to detect for the baseline model. On the other hand, these combinations also reduce the accuracy of the segmentation due to the increase of false positive classifications.

Consistency in the results is necessary to guarantee safe conditions for the system and the user in the autonomous navigation domain. However, DeepLearning applications require of huge amounts of data in order to obtain accurate and consistent results. The Image Buffer and Attention Module approach use sequential information generated by the baseline segmentation model to construct consisten results. The proposed extensions can serve as an intermediate solution when large data-sets are not available for training.

Contents

1 Introduction

1.1 Context

In an era for automation, it is not far fetched to think of a scenario where transportation does not suppose a hustle for the driver anymore. Regarding to commuting statistics, it is interesting to take a look at the American panorama as the living patterns are more standarized than over the different countries in Europe. A recent study by Statista [6] states that in America in 2016, an estimated 85.4 percent of 150M workers drove to their workplace in an automobile, while only 5.1 percent used public transportation for this purpose. Out of the 85.4 percent, a total of 77 percent (115M people) drive alone to work everyday [7].

Although some people enjoy the act of driving, it is fair to generalize that driving during rush hour is considered to be one of the most stressful scenarios for the daily commuting. While passengers can just sit and relax, the driver has to be constantly conscious about his actions during the whole ride. Autonomous Cars (AC) aim to free the driver from this activity, allowing him to spend his time in more valuable tasks. AC can also transform the current traffic system scenario by making it safer and more efficient to navigate, extending the benefits of automation to non-AC users.

Autonomous navigation however, is not a recent invention. In 1912, Lawrence Sperry successfully demonstrated the implementation of an autopilot-system on aviation. In an aircraft exhibition celebrated in Paris in 1914, Sperry performed numerous in-flight tricks in front of an audience to test the autonomy of the navigation system under no pilot conditions.

However, solving Autonomous navigation problems for cars, drones, bus, trucks... is not a trivial problem for different reasons:

- From the structural point of view, to list some examples: non-standardize roads (undefined or different lane sizes), inconsistent driving conditions (changes in weather, driving surface might deteriorate), obstacles or debris, ambiguous drivable space, undefined traffic signs location.
- From the non-structural point of view other factors come into play, such as: human or animal interaction (unpredictable behavior).

The eruption of DeepLearning on the last decade has allowed to create safer and more intelligent pilot systems that enable autonomous vehicles to operate better than under previously unseen scenarios. Deeplearning together with the motivation of some companies to take autonomous navigation systems into mass production makes the present time to be the perfect one to solve the autonomous vehicles enigma.

1.1.1 Autonomous Navigation

Although any system that requires autonomous navigation (cars, drones or any other mobile robot) can be considered for this topic, this document will focus on autonomous cars. The reason for this focus is that the late outburst of autonomous navigation in the automobile industry is promoting the scientific interest towards autonomous cars resulting in new studies and datasets that cover this application.

When talking about autonomy, the National Highway Traffic Safety Administration (NHTSA) has defined the following levels of car automation:

- Level 0: No Automation. The driver performs all driving tasks.

- Level 1: Driver Assistance. The Vehicle is controlled by the driver, but some driving assists features may be included in the vehicle design (such as ESP, Airbags, Lane keeping,...)
- Level 2: Partial Automation. Driver-assist systems that control both steering and acceleration/deceleration, but the driver must remain engaged at all times (e.g. cruise control or parking assistance).
- Level 3: Conditional Automation. The driver is a necessity but not required at all times. He must the ready to take control of the vehicle at all times with notice.
- Level 4: High Automation. The vehicle is capable of performing all driving functions under certain conditions. The driver may have the option to control the vehicle.
- Level 5: Full Automation. The vehicle is capable of performing all driving functions under all conditions. The driver may have the option to control the vehicle.

There are different companies trying to adapt classic vehicles to the different levels of automation. Nowadays most of the cars available have at least a level 2 of automation making level 3 the next step of the challenge.

Level 3 is currently dominated by Tesla, since the release of autopilot in 2016, Tesla has been manufacturing new vehicles surpassing the 1 Billion miles mark driven autonomously (followed by Waymo with 10 Million miles). Despite this big improvement, further levels of automation require a deeper study of the current technology and gather big amounts of data from driving patterns and uncommon situations.

In order to grant cars with autonomy, former cars need to be upgraded both in the hardware as well as in the software side. A key piece for this upgrade is the choice of the car's equipment. Cameras are the most common sensor present in autonomous vehicles (figure 1.1 shows an example of some of the sensors that are present on the autonomous cars), which along with other type of sensors are able to recreate a virtual representation of the surroundings.

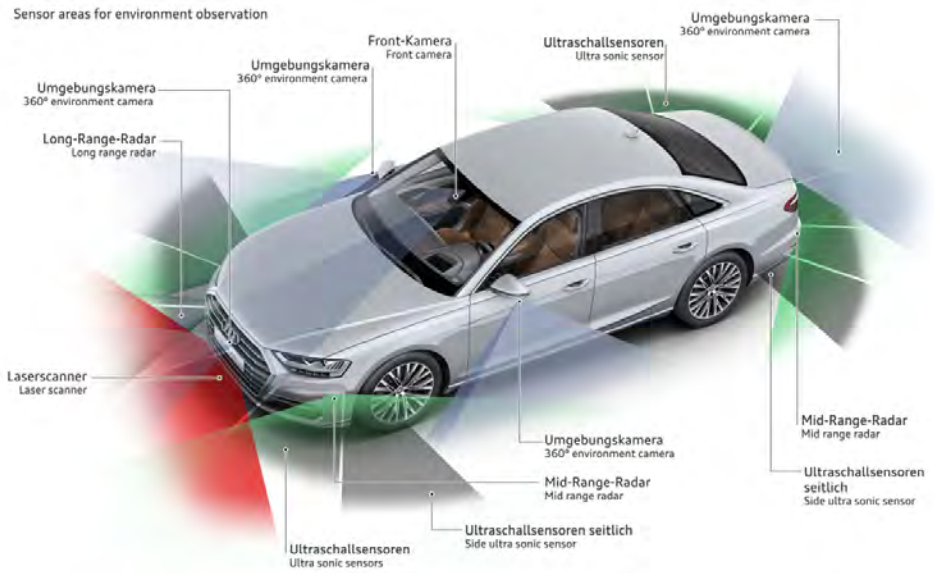

Figure 1.1: This figure depicts an example of the number of external sensors that are present for navigation in a hypothetical level 3 autonomous car. This specific set-up counts with 5 different cameras. Image source: [8].

The application domain of autonomous driving extends to anyplace with a drivable area (figure 1.2). Apart from the variety of roads, the difficulty of automation is enhanced by the bounds of

the problem: outdoors application. This loose definition of the domain specifications is what makes autonomous driving so challenging.

Figure 1.2: Different types of roads. From left to right: urban road, highway and rural road.

Granting a machine with the capacity of overtaking humans for tasks such as transportation is a non-trivial problem. Driving is a life-risk activity and therefore needs of a meticulous study, testing and evaluation of these new autonomous technologies.

1.1.2 Semantic image segmentation

Computer Vision is the field of engineering that focuses on the extraction of the information encoded on images for its use on different applications. Classical computer vision extracts this information through the calculation of different image descriptors. The calculation of the image descriptors is conditioned by the system characteristics: image resolution, object shape, light conditions, application domain. The process that defines the image descriptors derivation is called feature extraction.

Image descriptors are usually designed as hand-engineered filters, providing solutions that are rather rigid (application specific) and reliable only under very restricted conditions. Unfortunately autonomous navigation falls into a completely opposite scenario, requiring of applications that can perform robustly under very dynamic circumstances.

The main advantage of Deep Learning is its flexibility to generalize under previously unseen data. Since the application of Convolutional Neural Networks (CNNs) [9] for image processing, Deep Learning has been the protagonist on countless Computer Vision conferences and research papers. CNNs allow the extraction of features in a more efficient and meaningful way than classical image descriptors, based on image gradient calculations. Standing out due to their capacity of automation of image descriptors, CNNs allow the creation of Image Processing applications with a high level of abstraction and accuracy.

Semantic image segmentation (figure 1.3) is just one of the many Deep Neural Networks (DNN) applications. The goal Semantic Segmentation application is to detect and classify objects in the image frame by applying pixel-level classification of an input image into a set of predefined categories. Semantic image segmentation provides a level of scene understanding much richer than any other detection algorithms, it includes detailed information about the shape of the object and its orientation. Semantic segmentation can be used in autonomous navigation to precisely define the road (or drivable space) and its conditions (erosion, presence of obstacles or debris), it is also very useful for navigation in crowded areas being able to accurately calculate the gap between obstacles and even make predictions of the future position of the obstacles based on its shape and trajectory. Semantic image segmentation models can generally be divided into two parts: the feature extraction layers (hidden layers) and the output layers. The feature extraction layers use CNNs along with other techniques such as pooling or skip connections to obtain a low level representation of the image. And the output layers create the necessary relations to draw-out the pixel classification.

The scope of the project will be restricted to the analysis of a hypothetical video feed coming from the frontal camera of an autonomous car. The purpose of this camera is to elaborate a frontal representation of the environment that can be used for navigation. Flying objects, dirt

or sun glare are some of the external factors that can affect the correct performance of cameras. In order to guarantee the passengers' safety, the detection system of autonomous cars must stand out for the robustness of its results and all these situations need to be considered. An additional observation is that when applied to autonomous navigation applications, the segmentation should prevail the detection of obstacles over driving space to ensures the avoidance collisions.

Figure 1.3: Ideal result of a semantic image segmentation. In this figure, all the objects that conform the image are perfectly classified into the different colors that define each category: road, sidewalk, pedestrian, tree, building or traffic sign. This image is part of the Cityscapes groundtruth densely annotated dataset. Cityscapes is a large-scale dataset created as tool to evaluate the performance of vision algorithms intended for semantic urban scene understanding and help researchers to exploit large volumes of annotated data. Image source: [10].

1.2 Problem statement

Semantic segmentation allows autonomous cars to obtain an accurate representation of the outside world. This representation is used to define the available navigation space and the presence of obstacles necessary to calculate navigation trajectories.

Figure 1.3 shows an example of a perfect semantic image segmentation, however it is very difficult to obtain a segmentation in such a high level of detail. The deep learning model would require large amounts of high resolution finely annotated and varied data to allow the training optimization algorithm reach the desired accuracy while not overfitting. In contrast, figure 1.4 shows a real example of how an image that has been processed using an out-of-the-box state-of-the-art semantic image segmentation model (DeepLabv3 [4]) that was trained on the Cityscapes dataset [5] looks like.

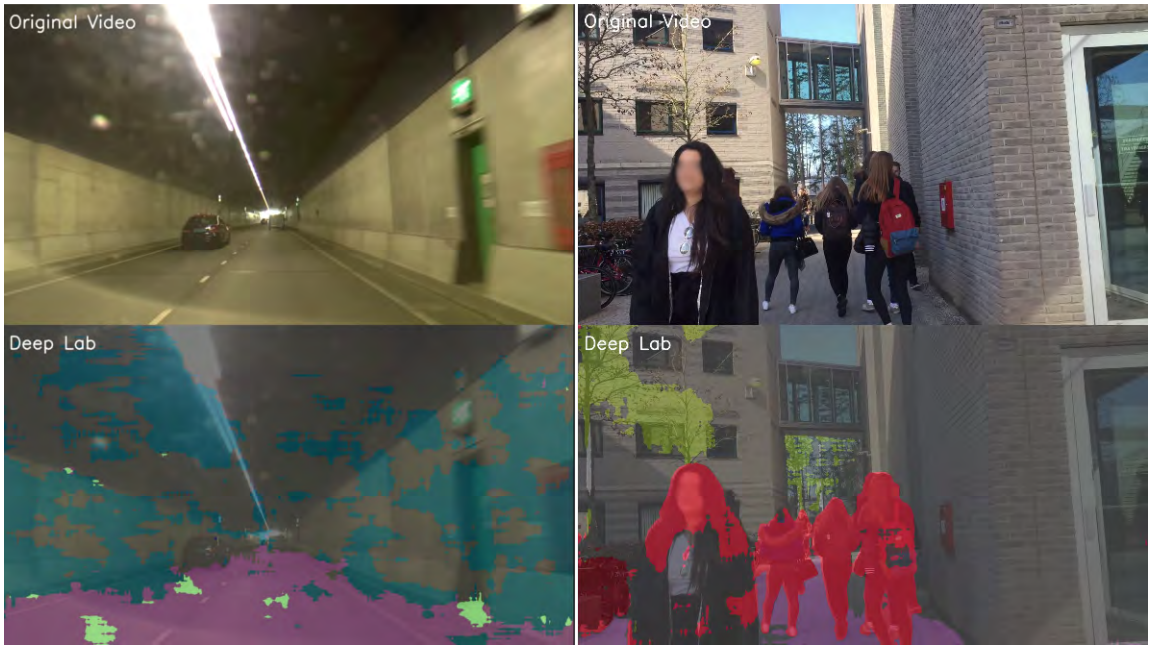

Figure 1.4: Figure illustration of the segmentation level obtained by DeepLabv3 [4], the current state-of-the-art semantic image segmentation model trained on the Cityscapes dataset [5]. This figure illustrates two different levels of segmentation imperfection. Left image: example of a totally missed classification of the cars in front of the camera, added to a noisy classification of the road and the walls. Right image: example of a partial segmentation of the pedestrians.

Figure 1.4 illustrates how the current level of a semantic image segmentation state-of-the-art model performance differ from the ground-truth example shown on figure 1.3. Although a partial classification might be good enough for obstacle avoidance, in some cases the semantic image segmentation model completely misses the classification of the obstacle and therefore can cause and accident. For this reason, autonomous cars have numerous sensors that allow the detection of obstacles at different distance ranges (figure 1.1) not relying on bare-semantic image segmentation models as the main source of information.

Another of the effects that can be observed after applying semantic image segmentation models for the analysis of videos is temporal inconsistency. Analyzing a video frame by frame causes a segmentation that is not consistent over time, small variations in the frame produce high variances in the segmentation (appendix A).

This master thesis examines ***how to reduce the incorrect classifications produced by semantic image segmentation models by combining the information of neighbouring frames***. In an attempt to improve the obstacle detection, this study can be broken down into the following research questions:

- *Analysis of the state-of-the-art:* ***what is the current state-of-the-art for semantic image segmentation?***
- *Temporal extension:* ***how to extend semantic-image-segmentation models for the analysis of sequences?***
- *Reducing missed classifications:* ***what kind of mechanisms can be applied to reduce the number of false classifications?***

1.2.1 Report Outline

This report is divided into a series of chapters that will guide the reader through the research examination. Chapter 2 is dedicated to the analysis of the necessary tools for this study, as well as an introduction of some of the previously implemented solutions. Chapter 3, defines the problem statement and the resources used during this study. Chapter 4, describes the design parameters of suggested approaches used to include temporal context to the segmentation and a series of experiments designed to evaluate the solution. Chapter 5, groups all the results for the different experiments. And Chapter 6 elaborates on conclusions and ideas that can be used as an extension of this research.

2 Background

As previously stated in section 1.2, one of the main goals of this master thesis is ***how to extend semantic image segmentation models for the analysis of sequences*** (videos are a sequence of images). The main difference between images and videos is that the latter consists of a group of images (frames) that are adjacent in time, indirectly encoding a temporal history. In order to exploit the sequential information present in videos, this chapter will study the available tools capable of modeling sequences. In chapter 4, some of these techniques will be used with a semantic image segmentation model in an attempt to add the video temporal information into the segmentation.

Given a causal system, sequence modeling consists on elaborating a model that is able to reproduce the dynamic behavior present in the observed data. From probabilistic methods to neural networks, this chapter summarizes different procedures used to capture temporal dynamics.

The methods reviewed in this chapter can be divided into two different groups depending on the tools used for sequence modeling: Conditional Probability and Deep Learning Architectures. The first one reviews causal modeling using probability relations. The second one introduces deep neural networks that have been specifically designed for modeling videos.

2.1 Sequence Modeling: Conditional probability

This section analyzes how to model the association between variables using probabilistic relations. Given two observed variables 'x_1' and 'x_2', the conditional probability of 'x_2' taking a value given that 'x_1' takes another value (two different events) is is defined as [11]:

$$P(x_2|x_1) = \frac{P(x_1, x_2)}{P(x_1)} \text{ , if } P(x_1) > 0 \tag{2.1}$$

Where the numerator of equation 2.1 is the joint probability of both events happening at the same time, and the numerator is the marginal probability of event 'x_1. It is possible to extend this notation to cover a bigger set of events (or a sequence of events). For a set of variables $X_t = x_1, x_2, ...x_t$ (for $t > 0$), the probability of the variable x_t conditioned to the rest of the variables in 'X_t' is:

$$P(x_t|\{X_\tau, \tau \neq t\}) = \frac{P(x_1, ...x_t)}{P(x_1, ...x_{(t-1)})} = \frac{P(X_t)}{P(\{X_\tau, \tau \neq t\})} \text{ , if } P(\{X_\tau, \tau \neq t\}) > 0 \tag{2.2}$$

Besides expressing the relation between variables using conditional probability notation, it is also possible to use graphical models or probabilistic graphical models. A Probabilistic Graphical Model (PGM) is a graph that expresses the conditional dependence relation between random variables. Conditional probability notation in combination with probabilistic graphical models are commonly used in fields such as probability theory, statistics and machine learning.

A possible graphical representation of equation 2.2 for $t = 4$ can be found in figure 2.1

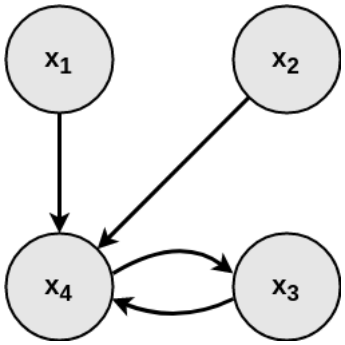

Figure 2.1: Possible probabilistic graphical model of equation 2.2 for $t = 4$. In this graph, 'x_4' depends on 'x_1, x_2 and x_3'; 'x_3' depends on 'x_4'; and 'x_1 and x_2' are each independent.

There are two main approaches that can be followed when defining the probability model for a dynamic system: a generative approach or a discriminative approach. Although the final purpose of both approaches is the same, to sample data from a probability distribution, each approach is different.

Generative models focus on modeling how the data is generated, in other words, modelling the joint probability $P(X_t)$, where 'X_t' is the set of variables involved in the process– e.g. in retrospect to the analysis of videos, each variable in 'X_t' can represent the value of a pixel over consecutive time steps, and 't' the frame index. A generative model is able to calculate the joint distribution of all the present variables '$P(X_t)$'. For the simple case of having 4 variables, modeling '$P(X_4)$' allows finding all the possible combinations for these variables: having observed the pixel at times 1, 2 and 3, it is possible to estimate x_4; or any other combination, such as computing x_2 given x_1, x_3 and x_4.

On the other hand discriminative models only focus on modeling the conditional relation between variables (equation 2.2), not paying attention on how the data is generated– e.g. having observed x_1, x_2 and x_3 it is possible to estimate x_4 but it does not allow to compute any other combination for the variables.

From the definition, generative models may appear to be more general and insightful about the present dynamic system than the discriminative ones. However, discriminative models are often preferred for classification tasks such as logistic regression [12]. The reason for this preference is that the generalization performance of generative models is often found to be poorer than the one proper of discriminative models due to differences between the model and the true distribution of data [13].

The most common procedures to create probability models will be reviewed in the following order: Naive Bayes Classifiers, Markov Chains and Hidden Markov Models (HMM).

Naive Bayes Classifier

The Naive Bayes classifier is a generative approach because it models the joint probability '$P(X_t)$' and afterwards calculates the conditional probability applying the Bayes Rule. Starting from the definition of conditional probability (equation 2.2), it is possible to apply the product rule of probability to the numerator, '$P(X_t)$' as:

$$P(X_t) = P(\{X_\tau, \tau \neq t\}, x_t) = P(\{X_\tau, \tau \neq t\}|x_t) \cdot P(x_t) \tag{2.3}$$

And the sum rule to the denominator to define $P(X_{(t-1)})$ as the marginal distribution of $P(X_t)$:

$$P(\{X_\tau, \tau \neq t\}) = \sum_T P(\{X_\tau, \tau \neq t\}, x_t) = \sum_T P(\{X_\tau, \tau \neq t\}|x_t = T) \cdot P(x_t = T) \tag{2.4}$$

where T comprehends all the possible states of x_t.

The Bayes Rule is the result of applying these two properties to the definition of conditional probability (equation 2.2):

$$P(x_t|\{X_\tau, \tau \neq t\}) = \frac{P(X_t)}{P(\{X_\tau, \tau \neq t\})} = \frac{P(x_t) \cdot P(\{X_\tau, \tau \neq t\}|x_t)}{\sum_T P(x_t = T) \cdot P(\{X_\tau, \tau \neq t\}|x_t = T)} \tag{2.5}$$

The general form of the Bayes Theorem says that *the posterior probability of an event is proportional to the prior knowledge of that event times the likelihood of the observation conditioned to that event.* In other words, if the probability of a given variable (or set of variables) $P(\{X_\tau, \tau \neq t\})$ is fixed, the posterior probability (equation 2.5) can be expressed as a proportional factor of the numerator.

Using the previous example that tracks the value of a pixel over 3 consecutive frames, the value of the pixel at time frame 4 will be given by:

$$P(x_4|X_3) \propto P(x_4) \cdot P(X_3|x_4) \tag{2.6}$$

In a more general form, the conditional probability of a state x_t given a set of previous observations from x_1 to $x_{(t-1)}$:

$$P(x_t|\{X_\tau, \tau \neq t\}) \propto P(x_t) \cdot P(\{X_\tau, \tau \neq t\}|x_t) \tag{2.7}$$

The Naive Bayes assumption states that the features (observations $X_{(t-1)}$) are conditionally independent given the class label (x_t)[11].

Applying the Naive Bayes assumption of independence allows to exploit the second term of equation 2.7 into '$t-1$' different terms:

$$P(\{X_\tau, \tau \neq t\}|x_t) = P(x_1|x_t) \cdot P(x_2|x_t), ..., P(x_{(t-1)}|x_t) \tag{2.8}$$

$$P(x_t|\{X_\tau, \tau \neq t\}) \propto P(x_t) \prod_{n=1}^{(t-1)} P(x_n|x_t) \tag{2.9}$$

Equation 2.9 defines a model that predicts the value for the state x_t for a set of observed states $X_{(t-1)} = (x_1, x_2, ..., x_{(t-1)})$. It is the final form of the Naive Bayes classifier, which as a consequence of the Naive Bayes assumption do not capture dependencies between each of the observed states in $X_{(t-1)}$ (figure 2.2). Even though this conditional independence assumption might sound unrealistic for real case scenarios, empirical results have shown a good performance in multiple domains with attribute dependencies [14]. These positive findings can be explained due to the loose relation between classification and probability estimation: 'correct classification can be achieved even when the probability estimates used contain large errors' [14].

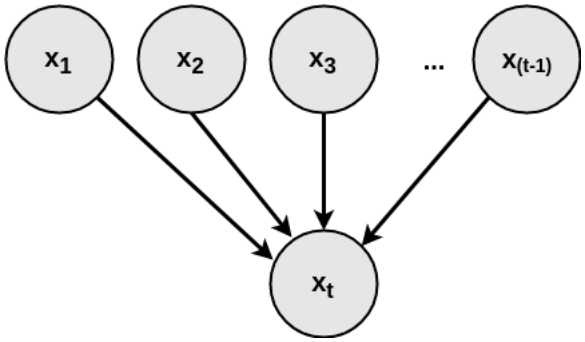

Figure 2.2: Graphical representation of the final form Naive Bayes classifier. It is based on the Naive Bayes assumption that states that: the observations $(x_1, ...x_{(t-1)})$ are conditionally independent given the value of x_t.

Markov Models, Hidden Markov Models and Conditional Random do not make any assumptions about the in-dependency of the variables and will be illustrated next.

Markov Chains

Markov chains or Markov Models (MM) are stochastic generative models for dynamic systems that follow the Markov property. Markov's property states that the future state of a variable depends only on the current observation (there is only dependence between adjacent periods). In the framework of video processing, Markov's property can be interpreted as: the value of a pixel in the present is only dependent on its immediate past (figure 2.3).

Figure 2.3: Graphical representation of a video as a Markov Chain. In this figure, each variable x_t represents the value of a pixel over time (t). The Markov property states that each variable depends only on the value of its immediate predecessor.

Using probabilistic notation, Markov's property can be applied as:

$$P(x_t|\{X_\tau, \tau \neq t\}) = P(x_t|x_{(t-1)}) \tag{2.10}$$

Where $\{X_\tau, \tau \neq t\}$ contains all the previous states from x_1 to $x_{(t-1)}$. The resulting joint probability of a Markov chain like the one present in figure 2.3, is defined as:

$$P(X_t) = P(x_1)P(x_2|x_1)P(x_3|x_2)... = p(x_1)\prod_{t=2}^{T} P(x_t|x_{(t-1)}) \tag{2.11}$$

In discrete-MM, the variables can only take certain values from a set of possible states that differ from each other. For a set of N possible states, there is a N by N transition matrix that contains the probabilities of transitioning between states. Figure 2.4 shows an example of a MM with two possible states x_1 and x_2, the transition probabilities that define this MM can be found in table 2.1.

	x_1	x_2
x_1	$P(x_1\|x_1)$	$P(x_2\|x_1)$
x_2	$P(x_1\|x_2)$	$P(x_2\|x_2)$

Table 2.1: Transition matrix of a Markov Model with two possible states (x_1 and x_2).

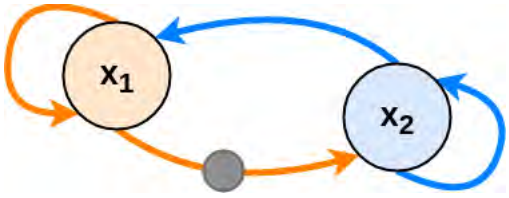

Figure 2.4: Graphical representation of a Markov Model with two possible states: x_1 and x_2. The connections between states represent the possible paths that the current state can follow. The values that condition each path are usually contained in a transition table (table 2.1). The dark circle is current state transitioning from x_1 to x_2. Image adaptation from: [15].

In a stochastic process, the rows of a transition probability matrix have to sum up to one, which means that a state has a finite amount of possible states. The values inside the transition matrix can be: given; calculated gathering samples from the process, doing a statistical analysis of the data and assuming that the process follows a certain distribution; or approximated using

probability distribution approximation methods such as the Metropolis-Hastings algorithm, that assumes an initial probability distribution and through several iterations it moves it closer to the real distribution [11].

The strong assumption made in equation 2.10 can be relaxed by adding dependence with more than one past states, transforming the MM into a k-order Markov chain. A second order Markov chain is illustrated in figure 2.5.

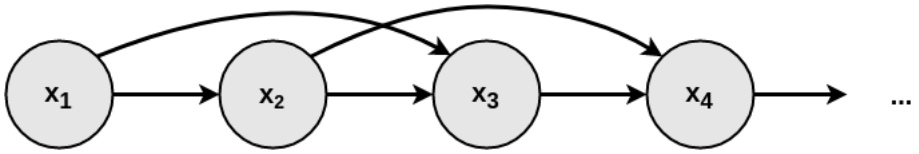

Figure 2.5: Graphical representation of a second order Markov chain. Image adaptation from: [11]

The corresponding joint probability of a second-order Markov chain follows the next equation:

$$P(X_t) = P(x_1, x_2)P(x_3|x_1, x_2)P(x_4|x_2, x_3)... = p(x_1, x_2)\prod_{t=3}^{T} P(x_t|x_{(t-1)}, x_{(t-2)}) \tag{2.12}$$

Equations 2.10 and 2.12 can be applied for processes where the state of the observed variables correspond with the state of the system. However, there are some processes with underlying hidden processes, these are called Hidden Markov Models and will be defined next.

Hidden Markov Model

Hidden Markov models (HMM) also belong to the stochastic generative models category. They differ from Markov chains due to having the observable variables related to each other through a series of underlying hidden processes '$Z_t = (z_1, z_2, ...z_t)$'.

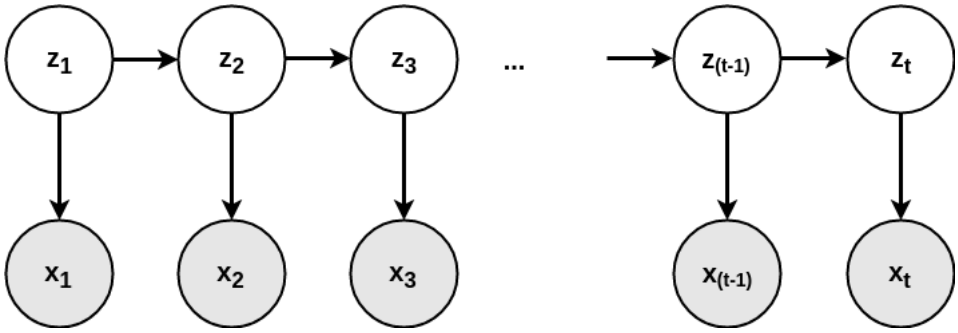

Figure 2.6: Graphical representation of a first order hidden Markov model. Image adaptation from: [11]

Figure 2.6 shows the representation of a first order HMM. There are two equations necessary to define this HMM. The relation between observable variables X_t and hidden processes Z_t; and the relation between hidden processes with each other:

$$P(x_t|\{X_\tau, \tau \neq t\}, Z_t) = P(x_t|z_t) \tag{2.13}$$

$$P(z_t|\{Z_\tau, \tau \neq t\}) = P(z_t|z_{(t-1)}) \tag{2.14}$$

Resulting in the following joint distribution:

$$P(Z_t, X_t|Z_t) = P(X_t|Z_t)P(Z_t|\{Z_\tau, t-1 \leq \tau < t\}) \tag{2.15}$$

$$= P(z_1, x_1|z_1)\prod_{t=2}^{T} P(x_t|z_t)P(z_t|z_{(t-1)}) \tag{2.16}$$

$$\tag{2.17}$$

The probabilities that relate hidden states 'z_t' (equation 2.14) are called transition probabilities, while the probabilities that associate hidden processes with observable variables 'x_t' (equation 2.13) are the emission probabilities. Both of them can be calculated in an analogous way to the transition probabilities for the Markov chains (table 2.1).

Markov Models and Hidden Markov Models, although more general than the Naive Bayes Classifier, are also limited by definition. Each state is defined only to be affected by a finite number of previous states 'k' and the effect of any other states happening before '$t-k$' is assumed to be encoded in this period, this limitation is often described as a short-term memory problem. Trying to find patterns in the sequence to determine the k-gram dependencies beforehand can help to alleviate this issue [16].

This section has introduced different methods that are used to model sequential information from the probabilistic theory point of view. Next section introduces different approaches used to overcome temporal context using Deep Learning architectures.

2.2 Video Modeling: Deep Learning Architectures

Considering videos as sequences of static images, this section can serve as an introduction to different approaches used to add temporal context to the analysis of videos.

2.2.1 Gated Recurrent Flow Propagation - GRFP

Seeking to solve the semantic segmentation inconsistency characteristic of evaluating video segmentation with individual image segmentation methods, [17] announced a method that combines nearby frames and gated operations for the estimation of a more precise present time segmentation.

The Spatio-Temporal Transformer GRU (STGRU) in [17], is a network architecture that adopts multi-purpose learning methods with the final purpose of video segmentation. Figure (2.7) shows a scheme of the STGRU architecture.

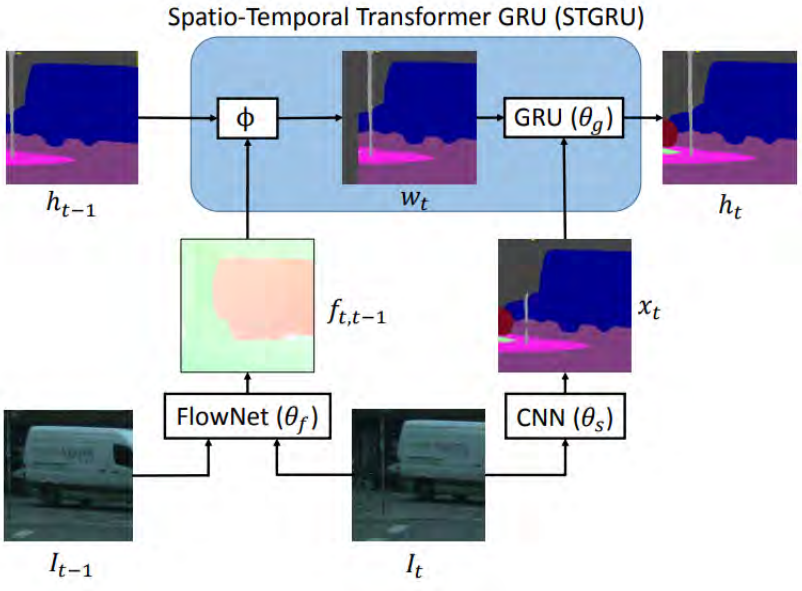

Figure 2.7: Overview of the Spatio-Temporal Transformer Gated Recurrent Unit. Pairs of raw input images are used to calculate the optical flow of the image (FlowNet). This optical flow is then combined with the semantic segmentation of the previous frame, obtaining a prediction of the present segmentation (blue box). A segmentation map of the present frame is then passed together with the prediction to a GRU unit that combines them based on the sequence. Image source: [17]

Inside of the STGRU, FlowNet is in charge of calculating the optical flow for N consecutive frames. A wrapping function (ϕ) uses this optical flow to create a prediction of the posterior frame. Later, a GRU compares the discrepancies between the estimated frame (w_t) and the current frame evaluated by a baseline semantic segmentation model (x_t), keeping the areas with higher confidence while reseting the rest of the image.

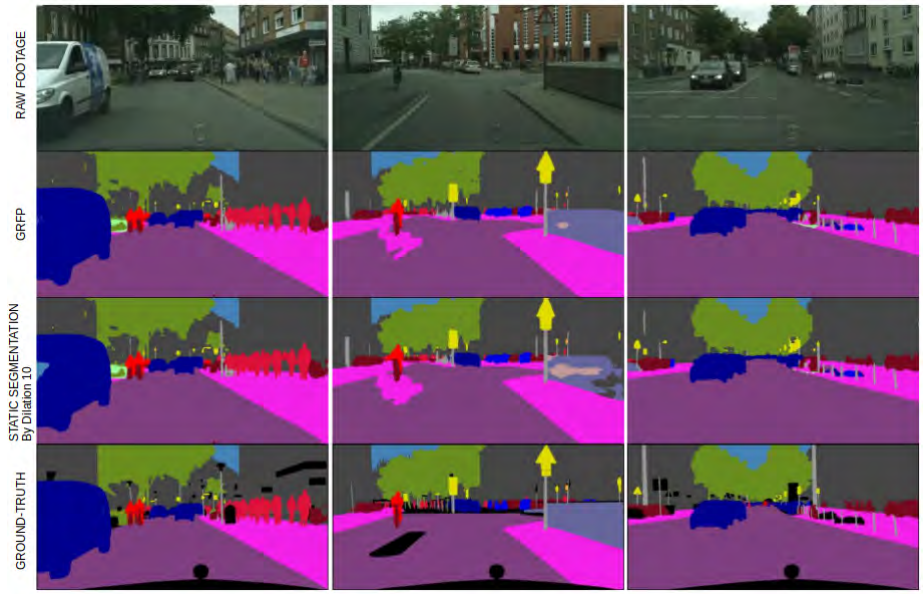

Figure 2.8: Image adaptation from Nelsson et al. [17]. In this image it can be seen a comparison between: GRFP method, Static Semantic Segmentation and the groundtruth segmentation. From left to right, GRFP achieves a segmentation improvement for the left-car, the right-wall and the left-pole.

The STGRU presented in [17] was evaluated both quantitatively and qualitatively (fig. 2.8), exhibiting a high performance compared with other segmentation methods.

2.2.2 Other architectures for video applications

The current trend for semantic video segmentation models consists on combining multipurpose neural networks (sequence modelling with feature extraction networks) into advanced models capable of efficiently performing this task.

Some other video segmentation architectures include:

- Feature Space Optimization for Semantic Video Segmentation [18].
- Multiclass semantic video segmentation with object-level active inference [19].
- Efficient temporal consistency for streaming video scene analysis [20].

Semantic image segmentation is not the only application in computer vision that can benefit from leveraging temporal context, tracking also use temporal analysis tools to achieve a better performance. The main reasons why temporal context is necessary for tracking are to guarantee the detection of the object even through occlusion and to reduce the number of identity switches (during multiple object tracking). These kind of applications are very common on surveillance or sport events. In 2017, [21] combined image appearance information with other tracking methods (Simple Online Real-time Tracker [22]) based on Kalman Filter and the Hungarian algorithm to obtain state-of-the-art detection at high rates (40Hz).

More information about any of the mentioned methods can be found on the cited references.

3 Problem Analysis

This chapter provides a detailed description of the problem and the materials that will be used for the study.

3.1 Domain Analysis

3.1.1 Semantic segmentation for Autonomous cars

Semantic segmentation is considered one of the hardest computer vision applications. It differs from image classification or object detection in how the classification is performed (figure 3.1). Image classification models classify the image globally, they assign a label to the whole image (e.g. cat or dog image classifier). Object detection models look for patterns in the image and assign a bounding box to the region of the image that is more likely to match with the target description (it provides classification and location within the image). And semantic segmentation produces pixel-level classification of an image; it describes each pixel of the image semantically, providing a more insightful description of how the image is composed than the other two methods.

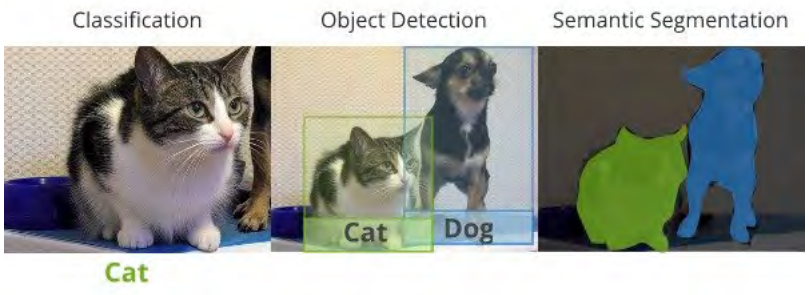

Figure 3.1: Figure comparison of the three computer vision classification applications. From left to right: image classification (global classification), object detection (local classification) and semantic segmentation (pixel-level classification). Image source: [23]

Humans are very good at segmentation of images, even without knowing what the objects are. This is the main reason why semantic image segmentation is necessary for autonomous navigation. Although other detection models are able to classify obstacles and locate them in the space, they can only find the obstacles they have previously seen. E.g. an obstacle detector used to avoid pedestrians in autonomous cars, it will only be able to alert the vehicle in the presence of pedestrians (it was just trained to learn how the pedestrian category is modeled). However, the type of obstacles that can be found in a undefinable driving scenario (it covers any object in any kind of shape and it is not feasible to create a dataset that covers for all), this is the reason why semantic image segmentation is present in autonomous navigation. An ideal semantic image segmentation model will be able to define the boundaries of any objects, even when these objects have not been previously 'seen' (figure 3.2). Apart from being a good obstacle detector, a perfect semantic image segmentation model has the ability to store these previously unseen objects, tag them and use them to re-train the network and improve the accuracy.

Figure 3.2: Adverse situations that can be solved using semantic image segmentation. Object detection models can only to detect objects that the model is familiar with. However, it is very difficult to create a dataset that includes all the possible types of obstacles, imperfections or debris that may appear in the road. Semantic image segmentation aims to achieve perfect definition of the image even when the objects are unknown.

There are some requirements that need to be present when applying semantic segmentation into autonomous navigation. The vehicle receives the data as a stream of images (it does not count with a video of the route beforehand) and it has to perform inference in real-time. The model should be very sensible on the detection of obstacles, e.g in an ambiguous situation where the segmentation of the road is not perfectly clear, due to imperfections or the presence of objects, the classification of obstacles must prevail.

3.1.2 Software analysis

A whole variety of programming languages, machine learning frameworks and semantic image segmentation models can be used for the purpose of this thesis.

The most common programming languages used for machine learning applications are Python and C++. The former is preferred on the research scope, while the latter is mainly used in commercial applications. Apart from the programming languages, there are different frameworks that provide the developer with the tools required to handle big amounts of data: Theano, PyTorch, TensorFlow or Keras are some of the frameworks compatible with both Python or C++.

Although the programming language and machine learning framework affect the performance of the application, the final result only depends on the implementation algorithm (semantic image segmentation model). As a matter of preference, this thesis study is developed using Python and Tensorflow.

3.1.3 Semantic image segmentation model selection

Depending on its inner structure, the different semantic image segmentation models are able to obtain different levels of segmentation accuracy and inference speed. Figure 3.3, although it is not up to date, shows some of the available possibilities arranged by accuracy and inference speed on Cityscapes dataset [5]. Cityscapes is a large-scale urban-scene dataset that contains high resolution fully annotated segmentation images for semantic segmentation applications (section 3.1.5).

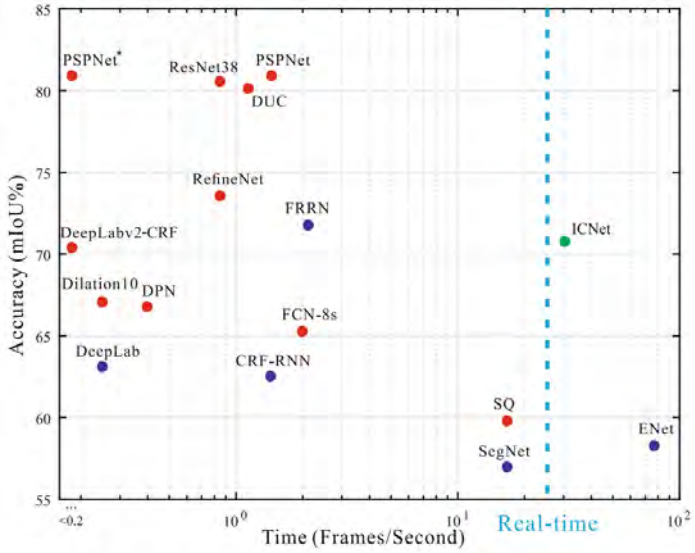

Figure 3.3: Classification of different semantic image segmentation models according to inference speed and accuracy (mIOU) on Cityscapes test set [5]. It can be observed how faster models (bottom-right corner) usually achieve a lower level of accuracy than slower ones (upper-left corner). Image source: [3].

In figure 3.3, the inference speed was measured by counting the amount of frames that the segmentation model is able to process each second. The mIoU (mean Intersection over Union, figures G.1 and G.2) measures the mean accuracy of all the frames processed at each speed. As a result, the upper-left corner contains the most accurate (but slower, $\sim 0-1$ frames per second) models while the less accurate (but faster, $\sim 10-100$ frames per second) are grouped on the bottom-right corner.

Later in the same year of the release of the study in figure 3.3, Chen et al. [4] in their paper *Rethinking Atrous Convolution for Semantic Image Segmentation*, introduced DeepLabv3, a new iteration of the DeepLab model series that became the state-of-the-art for semantic image segmentation models on Cityscapes test set (table 3.1).

In order to continue with state-of-the-art efficiency, DeepLabv3 [4] with pretrained-weights on Cityscapes dataset [5] is choosen as the baseline model for this thesis study.

3.1.4 DeepLabv3

DeepLabv3 [4], developed by Google, is the latest iteration of the DeepLab model series for semantic image segmentation– previous versions: DeepLabv1 [39] and DeepLabv2 [1].

DeepLab is based on a fully convolutional layer architecture (FCN) that employs atrous convolution with upsampled filters to extract dense feature maps and capture long range context [4]. [40] showed how powerful convolutional networks are at elaborating feature models and defined a FCN architecture that achieved state-of-the-art performance for semantic segmentation on the PASCAL VOC benchmark [41]. Another of the advantages of using FCN is that the architecture is independent of the input size, they can take input of arbitrary size and produce correspondingly-sized output [40]. In contrast, architectures that combine Convolutional Networks (for feature extraction) with Fully-Connected Conditional Random Fields (for classification)[39; 1] are designed for a fixed input size, as a result of the particular size necessary to pass through these classification layers.

One main limitation of solving semantic segmentation using Deep Convolutional Neural Networks (DCNNs) are the consecutive pooling operations or convolution striding that are often

Method	mIOU
DeepLabv2-CRF [1]	70.4
Deep Layer Cascade [24]	71.1
ML-CRNN [25]	71.2
Adelaide_context [26]	71.6
FRRN [27]	71.8
LRR-4x [28]	71.8
RefineNet [29]	73.6
FoveaNet [30]	74.1
Ladder DenseNet [31]	74.3
PEARL [32]	75.4
Global-Local-Refinement [33]	77.3
SAC_multiple [34]	78.1
SegModel [35]	79.2
TuSimple_Coarse [36]	80.1
Netwarp [37]	80.5
ResNet-38 [38]	80.6
PSPNet [2]	81.2
DeepLabv3 [4]	**81.3**

Table 3.1: Table comparison of the performance of different semantic image segmentation models on the Cityscapes dataset [5]. Table adaptation: [4].

applied into the DCNNs, consequently reducing the size of the feature map. These operations are necessary in order to increasingly learn new feature abstractions [42], but may impede dense prediction tasks, where detailed spatial information is desired. Chen et al. [4] suggest the use of 'atrous convolution' as a substitute of the operations that reduce the size of the input (figure 3.4).

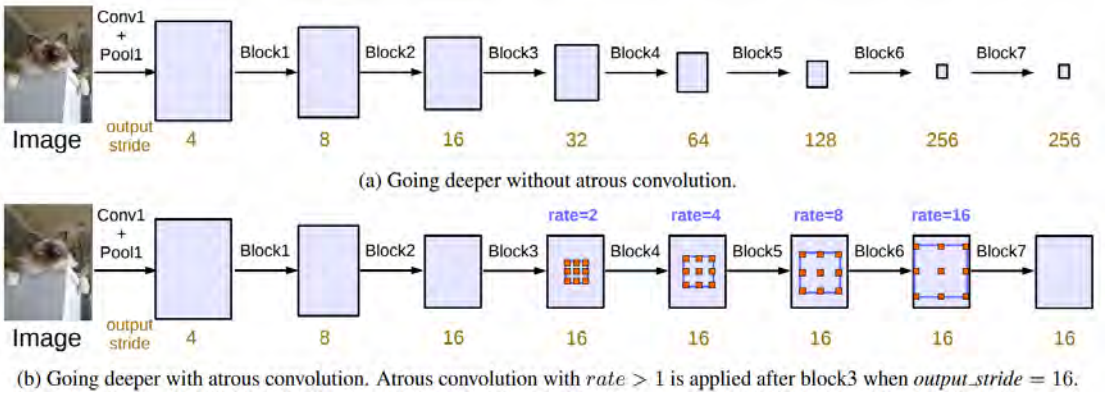

Figure 3.4: This figure compares the effect of consecutive pooling or striding to the feature map. (a) Shows an example where the feature map in the last layers is condensed to a size 256 times smaller than the input image, this is harmful for semantic segmentation since detail information is decimated [4]. (b) Applies atrous convolution to preserve the output stride obtaining equivalent levels of abstraction [4]. Image source: [4]

Atrous convolution, is also known as a dilated convolution. Apart from the kernel size, dilated convolutions are specified by the dilation rate, that establishes the gap between each of the kernel weights. A dilation rate equal to one corresponds to a standard convolution, while a dilation rate equal to two means that the filter takes every second element (leaving a gap of size 1), and so on (figure 3.5). The gaps between the values of the filter weights are filled by zeros,

the term 'trous' means holes in French. [39; 43; 1] show how effective the application of dilated convolution is in maintaining the context of the features.

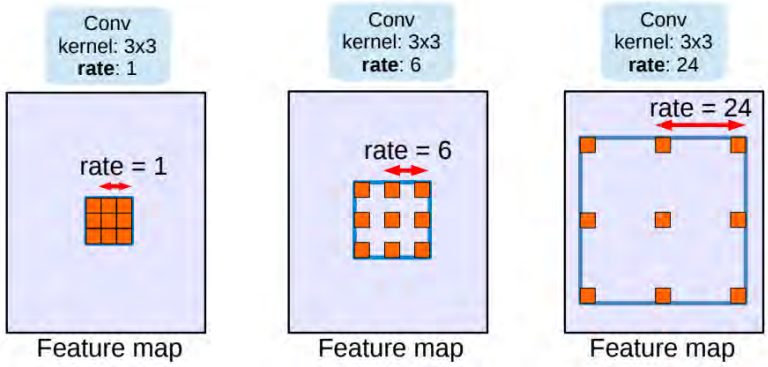

Figure 3.5: Atrous convolution of a filter with kernel size 3x3 at different stride rates. The dilation rate determines the space in between the different cells of the kernel. A rate=1 corresponds to the standard convolution, a rate=6 means that weights of the kernel are applied every sixth element (gap of 5 units), and so on. Image source: [4].

A second limitation faced by semantic image segmentation models is that they have to detect objects at multiple scales. This is a problem when using regular sized filters (normally 3x3) because they can only 'see' in regions of 9 pixels at a time, which makes it very difficult to capture the overall context of big objects. DeepLabv3 [4] employs Atrous Spatial Pyramid Pooling (ASPP) to overcome this issue, it consists on applying atrous convolution with different dilation rates over the same feature map and concatenate the results before passing it into the next layer figure 3.6. This approach helps capturing feature context at different ranges without the necessity of adding more parameters into the architecture (larger filters) [44; 45; 1].

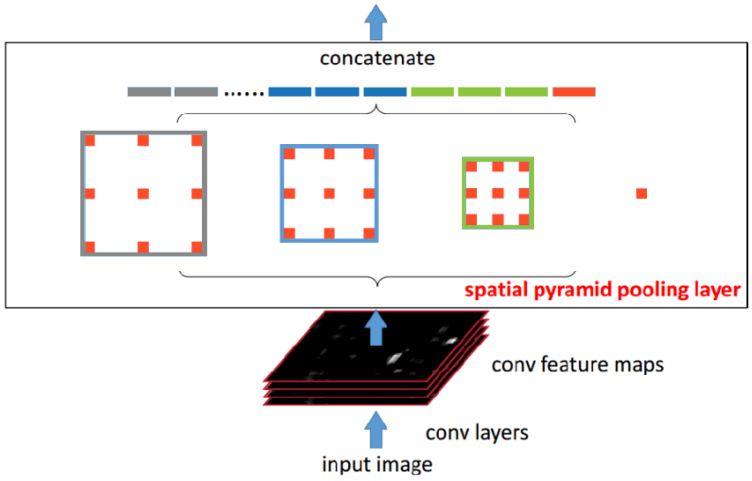

Figure 3.6: Graphical representation of ASPP. Atrous convolution with different dilation rates is applied on the same feature map, the result of each convolution is then concatenated and passed to the next layer. Spatial pyramid pooling is able to capture feature context at different ranges[44; 45; 1]. Image adaptation: [46]

Figure 3.7 shows the final architecture of DeepLabv3. Blocks 1 to 3 contain a copy of the original last block in ResNet [4]; which consists of six layers with 256 3x3 kernel convolution filters (stride=2), batch normalization right after each convolution and skip connections every 2 layers [47]. Block 4 is equivalent to the first 3 but it applies atrous convolution with dilation rate of 2 as a substitute of downsampling the image with convolutions of stride 2, maintaining the

output stride to 16. The next block applies ASPP at different rates and global average pooling of the last feature map, all the results of this block are then concatenated and passed forward. The resulting features from all the branches are then concatenated and passed through a 1x1 convolution before the final 1x1 convolution that generates the final logits [4].

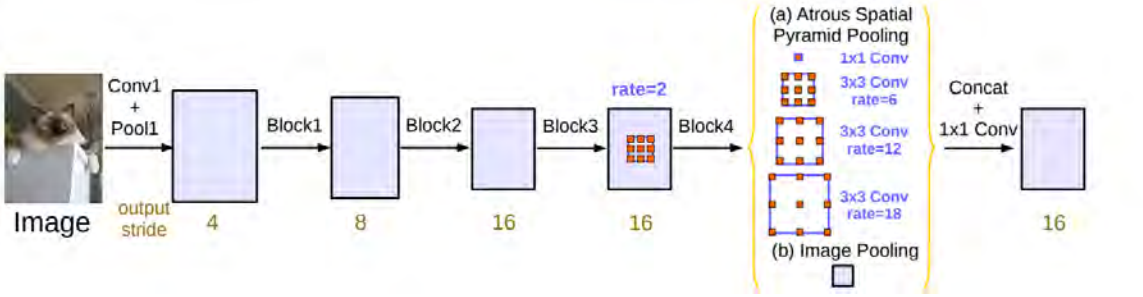

Figure 3.7: DeepLabv3 architecture. The first 3 blocks are a replica of the last block of the original residual neural network [47]. The following blocks incorporate the use of atrous convolution and ASPP, which stops the output stride reduction to 16, diminishing the negative effects of consecutive pooling. Image source: [4]

The output image consists on a HxWxC matrix where H and W correspond to the height and width of the output image and C is the number of categories in the dataset. Every pixel is assigned a real number for each category, that represents the likelihood (or logits) of that pixel belonging to each category, this is called the score map. The score map is then reduced by means of an argmax operation that determines the index of the category with the highest likelihood, obtaining the semantic segmentation map.

3.1.5 Cityscapes dataset

Cityscapes is the state-of-the-art dataset for urban scene understanding. It was created out of the lack of available datasets that adequately captured the complexity of real-world urban scenes. Despite the existence of generic datasets for visual scene understanding such as PASCAL VOC [41], the authors of Cityscapes claim that "*serious progress in urban scene understanding may not be achievable through such generic datasets*" [5], referring to the difficulty of creating a dataset that can cover any type of applications.

Nonetheless, Cityscapes is not the only dataset of its kind. Other datasets such as CamVid [48], DUS [49] or KITTI [50] also gather semantic pixel-wise annotations for the application in autonomous driving. Cityscapes is the largest and most diverse dataset of street scenes to date [5], it counts with 25000 images (figure 3.8) from which 5000 are densely annotated (pixel-level annotation) while the remaining 20000 are coarsely annotated (using bounding polygons, which offers a lower level of detail). Compared to the other datasets purposed for autonomous driving, Cityscapes has the largest range of traffic participants (up to 90 different labels may appear in the same frame)[5] and has the largest range for object distances, covering objects up to 249 meters away from the camera [5]. The 19 different classes that altogether form this dataset are listed in figure G.3.

Figure 3.8: Different types of annotations present in the Cityscapes dataset [5]. Upper figure shows an example of a densely annotated image (richer in detail). Bottom figure shows an example of a coarsely annotated image (lower level of detail). Image source: [10].

All these characteristics make the Cityscapes dataset the most challenging urban-scene understanding benchmark to date, "*algorithms need to take a larger range of scales and object sizes into account to score well in our benchmark*"[5]. Yet there are some limitations that need to be considered when evaluating the performance of a model that has been trained using this dataset. Cityscapes only captures urban areas (inner-city) of cities primarily from Germany or neighbouring countries [5], which may turn in a low efficiency when applied to highways, rural areas or other countries (due to differences in the architectures). The original images were taken during spring and summer seasons and do not cover situations with adverse weather conditions and or poor illumination. More information about the composition and a statistical analysis of this dataset can be found in [5].

3.2 Methodology

After several test runs of the baseline model on image sequences (appendices A and E), it was observed that the production of wrong classifications (completely or partially missed object's classification) is a transitory effect. When applied on a sequence of images, the baseline model is usually able to detect most of the objects producing segmentations of different qualities on each frame. Differences in the lighting conditions or noise in the image may be the cause of this variation from frame to frame. However, **this is an effect that can be exploited in order to achieve a better segmentation**.

The next conclusion came after analyzing a moving object over consecutive frames (appendix B). The moving object was recorded using a regular camera (at 30fps), it was noticed how the displacement of the subject from frame to frame was very small, depending on its relative speed with respect to the motion of the camera and its distance from the camera. This small displacement produced by objects moving at relatively low-medium speeds (walking person, moving bike or moving car) can be used as a motivation that the segmentation of neighbouring frames can be combined in order to obtain a more accurate segmentation of the present. In the next

chapters, this concept is regarded as the Image Buffer approach, figure B.2 shows an illustration of and Image Buffer of size 2: it holds 2 frames from the past and merges them to the segmentation of the present frame.

Apart from a straightforward combination of the pixel-classification output of neighbouring time frames (Image Buffer, section 4.1), a second approach that computes a weighted combination of the pixel-classification logits (section 3.1.4) produced by the baseline model will be introduced next.

As explained in section 3.1.4, the way in which DeepLab assigns the final classification labels to each pixel is by a previous calculation of a C-dimensional score array for each pixel, that together form a HxWxC scoremap for the input image (C is the number of possible categories, 19 in the case of Cityscapes; H and W correspond to the height and width of the input image respectively). The C-dimensional array contains the likelihood of each pixel to belong to each one of the possible categories of the dataset. The final pixel-labels are assigned by reducing the C-dimensional array of each pixel into one value that represents the index of the maximum value of the array (this index is the final classification label and it refers to one of the colors in figure G.3), this is done by means of an argmax operation.

The weighted combination of the classification scores as an approximated version of a conditional probability (section 2.1) is the second approach that will be tested in the next chapters. This method is referred as Attention Module (section 4.2).

3.2.1 Testing and evaluation

In order to stay truthful to the nominal conditions of baseline model, it would be necessary to test it using images with the same characteristics as the Cityscapes dataset [5], this is 2048x1024 pixel images. However, it was not possible to find video sources with the same resolution as the Cityscapes dataset and a 1920x1080 pixel resolution was adopted for the different tests. Although the difference on the number of total pixels between both formats is less than 2 percent, using lower resolution images than the ones used for training the weights of the baseline model might have an effect on the final segmentation. The study of this issue has not been covered and is added as one of the limitations in the discussions section (section 5.2).

The performance of both of the suggested approaches listed before, Image Buffer (4.1) and Attention Module (4.2) as well as the baseline performance (3.1.3) will be evaluated both quantitatively and qualitatively.

It is necessary to count with a groundtruth label annotation dataset to quantitatively measure the segmentation performance. However, groundtruth annotations for muli-label semantic video segmentation are very costly and no available datasets that covers this need were found. The Densely Annotated VIdeo Segmentation (DAVIS) 2016 benchmark [51] was chosen as an approximation for this requirement. It is formed by 50 densely annotated single-object short sequences, from which only 10 are suitable for the evaluation of this exercise (due to compatibility with the Cityscapes preset categories). The DAVIS categories that will be used for this study are:

- Breakdance-flare. A single person moving rapidly in the middle of the screen.
- Bus. A bus centered in the picture frame in a dynamic environment.
- Car-shadow. A car moving out from a shadow area.
- Car-turn. A car moving towards and outwards from the camera.
- Hike. A person moving slowly in the center of the frame.
- Lucia. A person moving slowly in the center of the frame.

- Rollerblade. A person moving fast from left to right of the frame.
- Swing. A person swinging back and forth and being temporarily occluded in the middle of the screen.
- Tennis. A person moving fast from right to left in the frame.

Since goal of this thesis is to improve the detection and the segmentation over time by reducing the number of missed classification and maintaining a consistent segmentation, the metrics used for the evaluation cover the temporal consistency and the accuracy.

A semantic image segmentation model that is consistent over time will produce a segmentation area with a smooth transition from frame to frame (depending on whether the tracked subject is moving or not). The segmentation area is calculated by counting the number of pixels classified with the target label at each time step. Afterwards, the frame-to-frame area variation is calculated as the difference of the area between consecutive frame pairs. A final computation of the standard deviation of these differences gives a global metric for the segmentation fluctuations (it is expected to obtain a lower number for more temporal consistent methods) that will be used for comparison of the different approaches.

The accuracy is calculated using the Intersection Over Union (described in section 3.1.3). And, in the same way as with the area, the frame-to-frame fluctuations of the accuracy are calculated as a comparison metric for all the approaches.

The qualitative evaluation consists on the observation and interpretation of the segmentation result of each one of the methods using different video sources.

The videos used for the qualitative evaluation are:

- Citadel - University of Twente
- Carré (modified) - University of Twente. One every ten frames was removed from the original clip to simulate a temporal occlusion or the faulty behavior of the camera sensor.
- Driving in a tunnel. Video source: [52].
- Driving under the rain. Video source: [53].
- Driving in the night. Video source: [54].
- Driving in low sun. Video source: [55]

3.3 Conclusions

Videos are a very powerful source of information. In contrast with the analysis of pictures, videos provide objects with a temporal context as a series of frames that can be exploited to benefit segmentation. In order to do so, two different approaches will be evaluated: Image Buffer and Attention Module (section 3.2).

These two approaches will build up on top of DeepLabv3 [4] pretrained on the Cityscapes dataset [5], which is chosen as the baseline for this thesis due to its performance as the state-of-the-art semantic image segmentation model (section 3.1.3).

The results will be evaluated both qualitatively and quantitatively in a series of videos chosen to cover a wide variety of scenarios (section 3.2.1). The metrics used for the comparison of the different approaches are chosen to cover both the accuracy and the temporal consistency of the predictions (section 3.2.1).

The machine learning framework and programming language are fixed together: Python and Tensorflow. Python allows for rapid script prototyping and debugging and counts with lots of

libraries that make working with images and arrays very natural. Tensorflow includes large amounts of documentation online, a very vivid community and it is being constantly updated with new utilities that offer new possibilities for the use of Deep Learning (section 3.1.2).

Next chapter describes in detail the design of each of the suggested approaches.

4 Design and Implementation

This chapter covers in detail both of the suggested approaches used to provide temporal context to the semantic image segmentation: Image Buffer (section 4.1) and Attention Module (section 4.2).

4.1 Approach I: Image Buffer

Image Buffer is the first of the approaches studied to solve for the semantic image segmentation models' temporal inconsistency (appendix A). It is the result of analyzing an object's location from frame-to-frame and the quality of the segmentation provided by DeepLabv3 (section 3.1.3) over consecutive frames (appendix B).

The first premise follows from figure B.1. *The translation of an object recorded at 30 fps over small batches of consecutive frames is negligible.*

The second premise follows from figure B.2. *The combination of the segmentation in neighbouring frames helps completing the segmentation and avoiding temporal miss classifications.* In figure B.2, 3 consecutive segmented frames were overlap in order to obtain an 'augmented segmentation'. The result of this experiment showed how the figure of a person was completed using the information of the 3 involved frames. Figure C.2 shows another example with inconsistent segmentation, in which a car is being partially segmented before it goes missing in the last evaluated frame. In this case, a segmentation combination of neighbouring frames would help to maintain some of the detection labels, avoiding the complete miss-classification of the last frame.

The origins of this fault in segmentation could be of different nature, from temporal physical occlusion such as flying objects covering the camera lens, to sensor saturation due to sun glare, or information loss due to system performance problems. Some of these cases are covered in chapter 5.

The approach works as follows:

1. The present time frame is extracted from the video feed and passed through DeepLabv3 that computes the segmentation map.

2. The segmentation map is concatenated along with the segmentation of the 3 previous time frames contained in the Segmentation Buffer (figure 4.1).

3. The segmentation of the 4 evaluated frames is combined obtaining the augmented segmentation for the present time (figure 4.2).

4. Lastly, the Segmentation Buffer is updated by dropping the oldest frame and storing the last frame drawn from the original video.

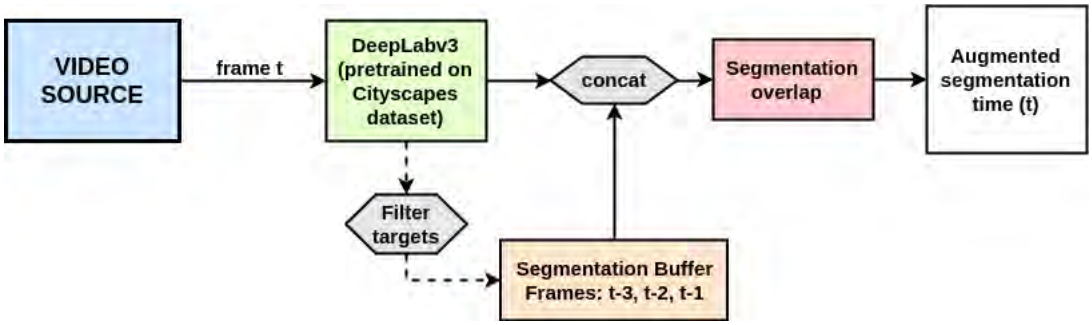

Figure 4.1: Block diagram of the Image Buffer approach (section 4.1). This block diagram depicts how the segmentation maps of 4 consecutive frames are combined into one (augmented segmentation).

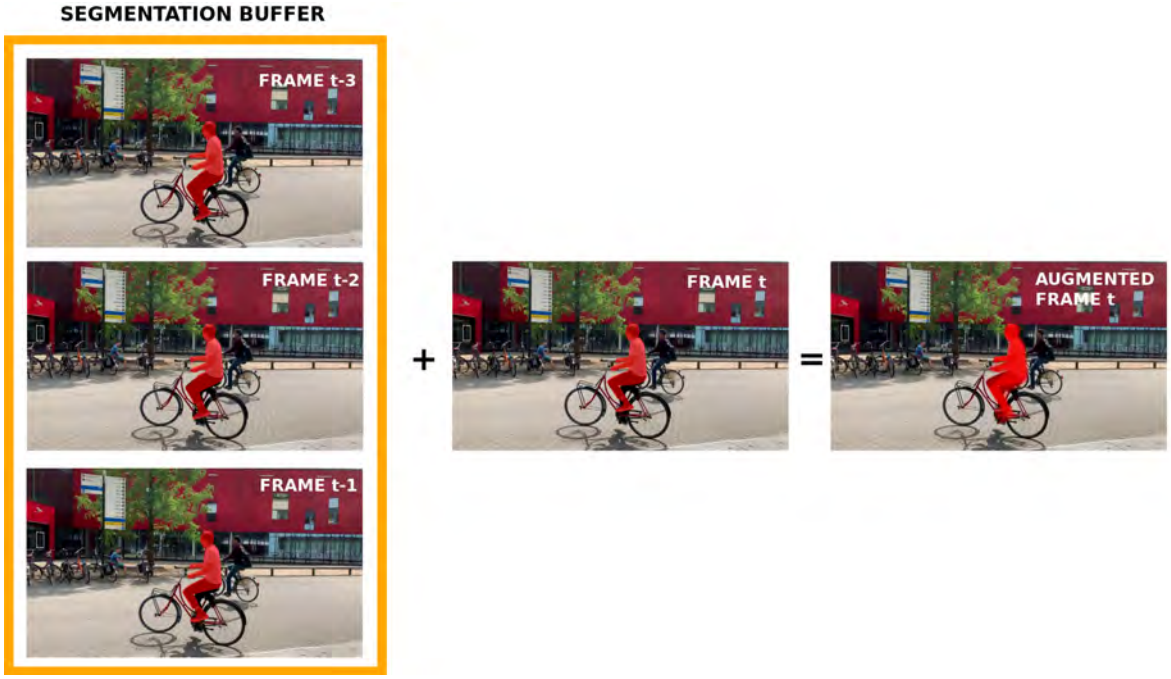

Figure 4.2: Conceptual representation of the Image-buffer approach. In this figure, the output is calculated as a combination of the segmentation in 3 consecutive past frames plus the segmentation of the present frame.

To delimit this approach it is necessary to define the size of the segmentation buffer, this follows from the observation made in figure B.1 that shows how the relative displacement on three consecutive frames is very small. For simplicity, the size of the image buffer is fixed to four: covers the present frame and three frames from the past.

This technique is only applied to a predefined subset of categories, the target categories, a step-by-step description of the algorithm can be found in appendix F. The reason for this discrimination is the concern towards the segmentation of the obstacles that can be found in the road. As a proof of concept, the target categories for the experiments are: 'person', 'rider', 'car' and 'bicycle'.

It is expected that this approach alleviates the defect of partially or completely missed classifications over the target categories. Nevertheless, the effect of this approach will only be apparent if the defect labels are missing for less than the size of the Image Buffer (if the labels are missing for longer than that, this approach is not able to recover them).

Overlapping past information for the inference of the present segmentation imposes a temporal lower bound to the object detection, at the same time, it reduces the time inconsistency

alleviating the 'flickery' effect. Static elements or elements that move slowly with respect to the image frame are expected to benefit from this approach. The results of this application will be presented on the next chapter.

4.2 Approach II: Probabilistic Approach

Apart from a bare combination of the segmentation maps obtained at different time frames (section 4.1), there is also the possibility of making a more educated combination by modifying the probability map of each frame prior to the combination, resulting in an augmented segmentation. This approach is inspired by the transition probabilities in Markov Models that model the probability of an event transitioning between different possible states (section 2.1).

The intuition for this approach comes from the following line of thought: 'given that machine learning models are able to classify by calculating a confidence score (the logits) over a set of predefined labels, can those values be extended overtime (video inference) and used to influence the classification of consecutive frames?'. In other words, is it possible to establish a causal relation between past and present frame pixel-classification?

There are two problems that need to be addressed in order to create a temporal relation on the semantic image segmentation classification. First, the logit-map of each frame needs to be obtained and analyzed. Second, a relation between consecutive frames has to be defined.

4.2.1 Semantic image segmentation - Scoremap

In semantic segmentation, the logit-map is a HxWxC matrix that contains, for each pixel in the image (HxW) a vector of real numbers that represents the likelihood of belonging to each one of the predefined categories (C), the logits. The logit-map is located one step before the final assignation of labels (figure 4.3) and is formed by real numbers without any apparent bounds.

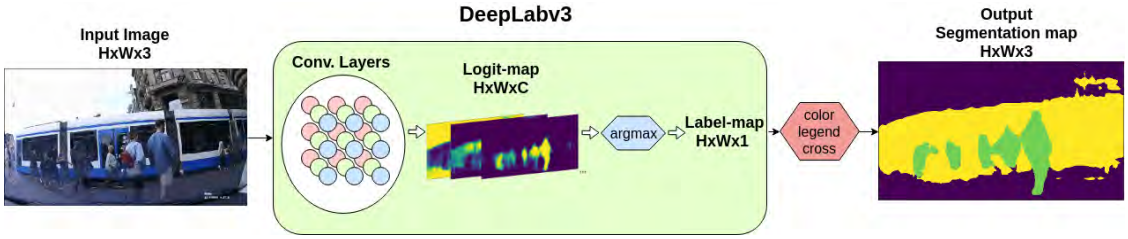

Figure 4.3: Graphical interpretation of the DeepLabv3 [4] semantic image segmentation tool-chain. The convolutional layers produce a HxWxC matrix (the logit-map) that is reduced by means of an argmax function into a HxWx1 matrix with the final labels of each pixel (the label-map). Finally an external function assigns a color value to each label.

In order to establish a relation between consecutive frames, it is necessary to transform the logits of each pixel to a common scale for comparison. This can be done by means of a softmax function (equation 4.1) that transfer the raw prediction values of the neural network (the logits) into probabilities (logit-map into a probability-map). Figure G.4 shows an example of the logit-map after the application of the softmax activation function.

$$S(C_i) = \frac{e^{C_i}}{\sum_j e^{C_j}} \tag{4.1}$$

where C_i refers to each element in the logits vector C.

Once that the probabilities have been determined, it is possible to establish a relation between consecutive frames.

4.2.2 Modeling videos as a weighted sum of frames

The Attention Module models the classification probability of the present as a weighted sum of consecutive frames (equation 4.2). The number of frames in consideration for the sum is four (three past frames plus the present frame), this is a conservative choice given the results obtained in appendix B.

$$\text{aug} = I_0 \cdot p(t) + I_1 \cdot p(t-1) + I_2 \cdot p(t-2) + I_3 \cdot p(t-3) \tag{4.2}$$

aug is the resultant augmented logits (figure 4.4), I_n are the weights of frame $(t-n)$ and the probabilities $p(t-n)$ are extracted from the baseline segmentation model (section 3.1.4) after applying the transformation of the softmax function.

Weights estimation

In theory, it should be possible to find the optimal weight values that shape equation 4.2 resulting in the probabilities with the smallest discrepancy with respect to the true segmentation. This is an optimization problem that requires the definition of a loss function and data-set with ground-truth annotations. In classification tasks the most common loss function is the cross-entropy function or the negative log likelihood[56] (equation 4.3)

$$L = -\sum_{c=1}^{M} y_{o,c} log(p_{o,c}) \tag{4.3}$$

where M is the number of classes; $y_{o,c}$ is a binary indicator that the class label c is the correct classification for the observation o; and $p_{o,c}$ is the predicted probability of observation o being of class c [57].

As opposed to the more common quadratic loss function, the cross-entropy loss function does not suffer slow learning due to the computation of small gradients [56]. The gradients computed on the cross-entropy loss function increase when the prediction moves further from the target value [56] (appendix G.4 for derivation).

Besides the definition of a loss function, it is also necessary to count with a ground-truth annotation data-set. However, as mentioned in section 3.2.1, no public multi-label semantic video segmentation data-set was found and developing one was not feasible due to the lack of resources. As an alternative, in order to find the 'best' weight values, the equation 4.2 was tested using different parameter combinations on some of the videos listed in section 3.2.1. The result of these tests can be found in appendix C. The value of the final weights can be found in table 4.1.

I_0	I_1	I_2	I_3	T
4	3	2	1	1

Table 4.1: Weight parameters for equation 4.2

As a result, using positive weights (bigger than 1) increases the sensitivity of the per-pixel classification, which added to the combination of successive frames result in a more consistent temporal segmentation (section 5.1.1). However, the increase in the sensitivity provokes a higher number of false positive classifications in the form of noise (appendix C). The amount of false positive labels can be alleviated by setting up a threshold that pushes the augmented logits to zero if a minimum value is not reached (figure 4.4). Similar to the Image Buffer, this approach is only applied to a certain category targets as a proof that it can be used to leverage the classification of any category labels. Algorithm 2 (appendix F) describes step by step the application of this approach as an extension of the baseline classifications.

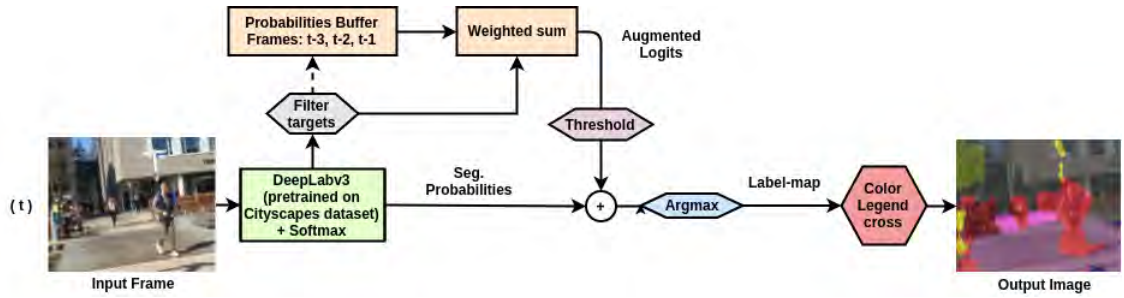

Figure 4.4: Block diagram of the Attention Module approach (section 4.2). This block diagram depicts how the classification probabilities of 4 consecutive frames are combined into one.

4.3 Experiments

The experiments are divided in two parts: a quantitative evaluation that measures the performance of each method using different metrics and a qualitative evaluation that is done by interpretation of the segmentation results.

The list of experiments and evaluation metrics can be found in section 3.2.1.

5 Results and Discussion

This chapter collects and analyzes the results for all the experiments listed in section 3.2.1. It also critically analyzes the followed methodology highlighting the best case scenarios and revealing the possible failure modes.

5.1 Results

5.1.1 Quantitative results - Davis data-set

This section evaluates the performance of both suggested methods (Image Buffer and Attention Module), comparing them with the baseline segmentation (using DeepLabv3). Since the suggested methods are designed to *reduce the number of false negative by leveraging the temporal information encoded in image sequences-videos*, this section looks at the segmentation performance measuring and plotting the *Area overtime* (AOT) and *IOU overtime* (IOU-OT). In addition, the frame to frame variation of both metrics will be calculated and plot side by side, expecting to find less variations on the approaches that handle past time information.

In order to quantify and compare the performance of each method, it is necessary to count with ground-truth annotations. However, it was not possible to find a data-set that covered this need and it was not feasible to create a custom one given the time-line. For this reason, the DAVIS data-set [51] is used as an approximation with the limitation it can only evaluate one label at a time. The list of categories covered for this objective can be found in section 3.2.1.

Figures 5.1 and 5.2 give an example of the segmentation evaluation for the 'tennis' category of the DAVIS data-set [51]. Figure 5.1 shows an obvious result on how leveraging neighbouring frames benefits the final segmentation output. In this first figure in can be observed how the baseline segmentation is surpassed by both of the suggested methods, achieving a smoother shape with the Attention module approach. The main difference between both of the suggested methods is that the Image buffer directly uses the segmentation produced by the baseline model, while the Attention module by modifying the probability map of the segmentation is able to produce new segmentation labels that might benefit the final segmentation. In figure 5.1 it can also be observed how both of the suggested methods increase the number of false positive classifications, although it was intended to limit this number the study of its reduction is left as future work.

Figure 5.2 contains the graphs that measure the AOT, the IOU-OT and its variations for the 'tennis' category of the DAVIS data-set. The AOT is calculated to compare the differences in the production of labels of each method. As expected, it can be observed how the area detection for both of the suggested methods is superior than the baseline detection and the ground-truth annotations. The reason for this is the temporal combination and the increase in the production of false positive classifications. The IOU-OT indicates the accuracy of each method over time, expecting values closer to 1 for the best case scenarios. The results show how both the Image Buffer and the Attention Module generally reproduce a lower accuracy than the baseline method, which can be explained due to the increase of false positive classifications. Lastly, the variation of each metric is computed to asses the reliability of each method (a reliable application is expected to give consistent results overtime). For a perfect segmentation, the AOT variation should be similar to the ground-truth annotations and the IOU-OT variation equal to zero.

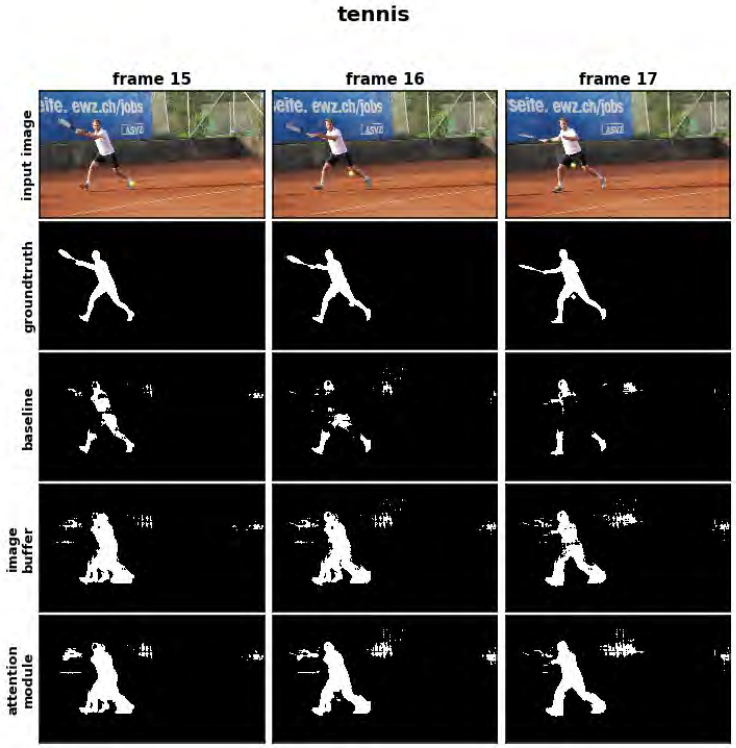

Figure 5.1: Figure comparison of the segmentation of 3 consecutive frames and its corresponding groundtruth annotation. It can be observed how both the Image Buffer and the Attention Module achieve a better segmentation of the 'person' category although there is also an increase in the amount of false positive classifications. Sequence source: DAVIS dataset [51] - 'tennis' category.

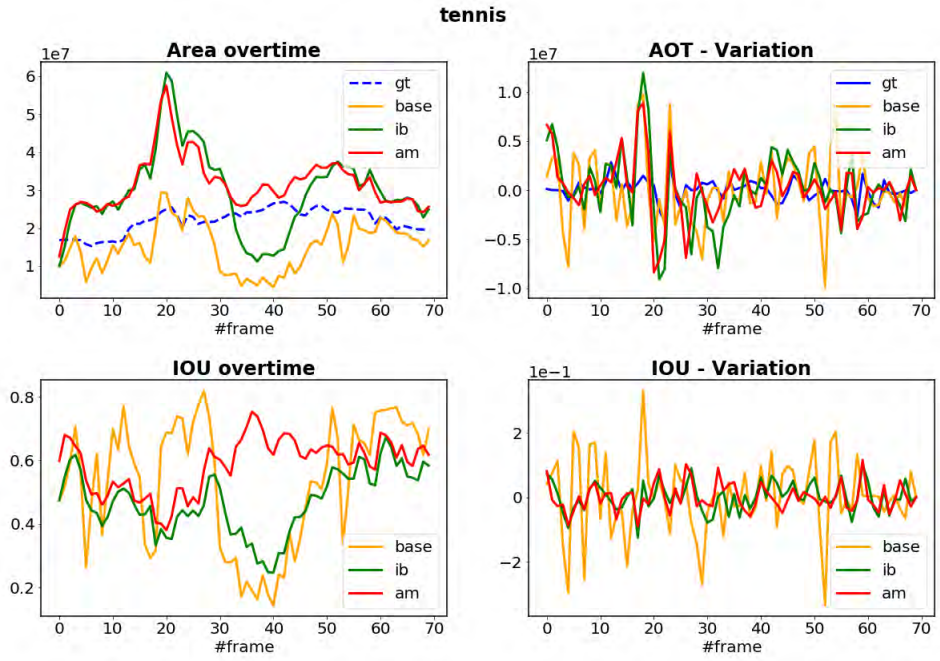

Figure 5.2: Graph evaluation of the 'tennis' category (DAVIS dataset [51]) using the metrics defined in section 3.2.1. The upper graphs shows how the AOT of both suggested approaches is higher than the baseline model, although they tend to have a lower variation. The lower graphs show how both of the suggested approaches tend to be less accurate than the baseline segmentation, this can be reasoned by the increased production of false positive classifications. In addition, the IOU variation for both the IB and AT is lower than for the baseline.

The figures corresponding to the rest of the tested categories can be found in appendix D. In addition to these graphs, the dispersion of the variation of each metric is calculated by computing the standard deviation of the sequence; tables 5.1 and 5.2 gather these results. The variation is computed to measure the frame-to-frame consistency of the prediction, a segmentation that fully captures the dynamics of the sequence (that is consistent overtime) would result in a AOT Variation standard deviation equal to the ground-truth's one and a IOU-OT Variation standard deviation equal to zero.

	AOT Variation STD			
	Groundtruth	**DeepLabv3 (Cityscapes)**	**Image Buffer**	**Attention Module**
Breakdance-flare	2297873	4771063	3649613	**3347300**
Bus	1441019	4600729	3042283	**2936702**
Car-shadow	285037	1388070	**1025686**	1219912
Car-turn	1128675	1767648	1221533	**1212808**
Hike	342290	1992053	**1044184**	1268832
Lucia	443833	1430598	**991134**	1453644
Rollerblade	666624	3548873	3240213	**2480139**
Swing	1663433	**4711279**	4975625	4908373
Tennis	1073167	3690312	3523527	**2904218**

Table 5.1: AOT Variation STD for some of the DAVIS data-set [51] categories evaluated using different semantic segmentation approaches (section 3.2.1). Although the three approaches under study are far from achieving a Variation close to the groundtruth, the consistency of both the Image Buffer and the Attention Module outperforms the bare DeepLabv3 implementation.

	IOU-OT Variation STD (%)		
	DeepLabv3 (Cityscapes)	**Image Buffer**	**Attention Module**
Breakdance-flare	8.65	4.64	**4.11**
Bus	4.58	**2.82**	2.92
Car-shadow	2.97	**1.75**	1.81
Car-turn	1.53	0.93	**0.89**
Hike	4.68	1.57	**1.49**
Lucia	2.85	**1.95**	2.13
Rollerblade	14.67	7.25	**5.89**
Swing	5.56	3.69	**3.78**
Tennis	12.12	4.43	**4.20**

Table 5.2: IOU-OT Variation STD for some of the DAVIS data-set [51] categories evaluated using different semantic segmentation approaches (section 3.2.1). A robust segmentation would be translated in having a low STD, which means that the accuracy does not fluctuate over the sequence. It can be observed how the lowest values are obtained by the two approaches that exploit temporal information, the Image Buffer and the Attention Module.

Although the results shown in this section are a simplification (single-label classification) of what it might be present in a real driving scenario, these results can serve as a guideline on how the combination of frames affects to the final segmentation. Next section presents the results for a series of multi-label classification experiments.

5.1.2 Qualitative results - Video evaluation

In this subsection, the testing is extended to cover the multi-label detection. Due to the lack of groundtruth annotations, these experiments will be evaluated by the visual-quality of its results.

Figure E.1 shows the segmentation performance of the baseline model (DeepLabv3) and its extension with the image buffer approach (detected area is contoured in red or blue) and the attention module approach. In particular, the extensions only leverage a set of predefined categories over the rest: **person, rider, bicycle, car and bus**. This is done as a proof of concept as well as due to hardware limitations, but it can be adapted to hold any other target categories. The color palette used for the classification can be found in figure G.3.

Citadel - University of Twente

This video was taken on a normal day between lectures at the University of Twente. It was chosen to evaluate the performance of these methods in an environment rich on features with different targets present at the same time. Figure shows how both of the extensions help completing the partial segmentation of some targets.

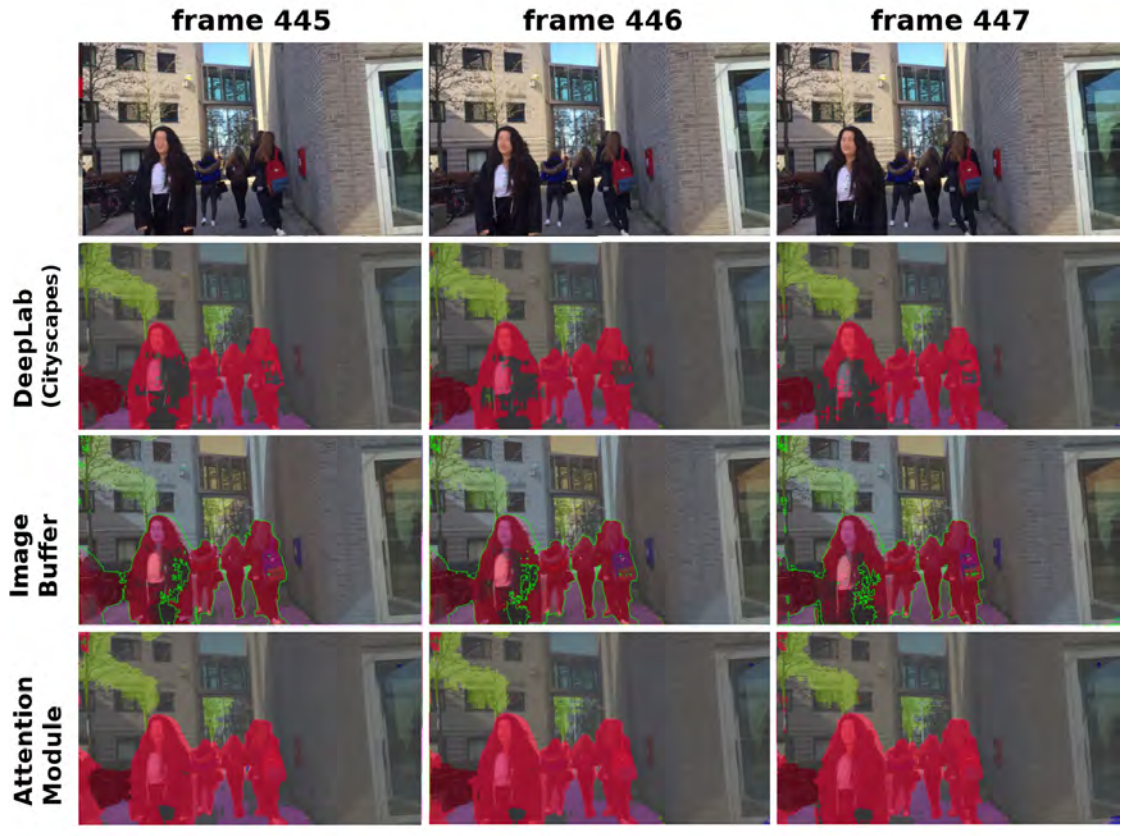

Figure 5.3: Figure comparison of the segmentation of three consecutive frames for a video sequence recorded outdoors in the presence of multiple target categories. It can be observed how the baseline segmentation (DeepLab) is able to detect most of the subjects and how the suggested extensions help completing the segmentation.

The rest of the qualitative experiments mentioned in section 3.2.1 can be found in appendix E. Next section elaborates on the limitations of each method and makes a conclusion based on the results.

5.2 Discussion

The results in appendix D and E show how the Image Buffer and Attention Module approaches may benefit the segmentation of a set of predefined target categories, this is due to the combination of information from neighbouring frames into the present segmentation. However, the definition of both of these extensions is limited to the set-up of this study: camera type, segmentation model (section 3.1.3) and training data-set (section 3.1.5).

Both of the extensions combine the segmentation of 3 past frames with the present frame. This is the result of the study performed in appendix B and can only be applied to study-cases where the target object location does not differ considerably for 4 consecutive frames. In other words, it only applies to cases with a high recording-speed/target-relative-movement (w.r.t the camera) ratio. A low ratio will result on a 'ghosting effect' on the segmentation and a diffuse boundary definition of the targets.

The segmentation model also affects the final performance of both of the suggested extensions. Both of the extensions base their results on an initial baseline segmentation that is lastly modified. In the case of the Image Buffer this limitation is not very extreme and the final accuracy directly depend on the efficiency of the baseline model. In the other hand, the Attention Module modifies the probability-map of each frame by its multiplication with different weights (positive, greater than 1), this helps increasing the sensitivity of the detection of the target categories but in turn may produce false positive classifications.

Another limitation of the Attention Module approach is that its parameters were calculated heuristically, and the segmentation result might be sub-optimal. This was inevitable due to the lack of a sequential ground-truth data-set that could be used to find the optimal values of the weights and the threshold value (table 4.1).

The chosen data-set for training the baseline model also affects the final performance of the segmentation. It determines the final accuracy of the segmentation and sets some limitations on the resolution of the images and the covered scenarios. Cityscapes data-set (section 3.1.5), covers driving scenarios in urban areas which might be the reason why the baseline segmentation offers a low performance in figures E.3 and D.3 (due to the angle of the camera). Besides that, Cityscapes gathered samples during spring and summer seasons, so it does not include adverse weather conditions nor poor illumination images.

Nevertheless, both of the extensions have proved how leveraging sequential information for the application of real-time predictions is beneficial for the consistency of the results. When applied to autonomous navigation, creating an image segmentation training data-set that covers all the possible scenarios is very difficult and holding the information of past frames adds a protection layer against failure modes, which can be very useful at the early stages of the project.

6 Conclusions and Recommendations

6.1 Conclusion

The evaluation of the baseline performance (appendix A), showed that the output segmentation over a sequence is irregular. This is a problem when the application relies on the frame segmentation as a main source of information. In order to attenuate these variations, both of the extension approaches (Image Buffer and Attention Module) combine the information of neighbouring frames to make a more educated segmentation of the present. Tables 5.1 and 5.2 and the graphs in appendix D show the effect of these extensions reducing the fluctuations on the segmentation overtime.

This thesis also evaluated some cases in which the baseline model was not able to detect a target in the field of view. These extensions proved to reduce the number of false negative classifications. The Image Buffer approach sets a time lower bound on the segmentation providing a safety margin over sudden faulty modes. The Attention Module, due to the modifications on the probability map, is able to increase the sensitivity of the detection of a set of predefined targets while at the same time providing the advantages of the Image Buffer approach. Which can be very useful for the detection of obstacles in the case of autonomous navigation (figures E.3 and E.2).

Despite the improvement in the segmentation by combining information from neighbouring frames, this solution might not be optimal for the application in real-time systems. Both of the extensions (Image Buffer and Attention Module) require to store an array of dimensions: $MxHxWxN$. Where M is the number of target categories, H and W the dimensions of the image source and N the number of frames to be combined. This can be computationally very costly to be evaluated in a real-time implementation.

In contrast, it is suggested to use this methodology during the training phase of the segmentation model. The final user can create artificial annotations for the categories that appear to be more challenging for the baseline model and use them as ground-truth annotations for further training. This would expect to improve the segmentation over those target categories and get rid of the computational constraint imposed by having to store an array for the segmentation augmentation.

6.2 Future research

Once that it has been proved how using sequential information can be favorable for semantic video segmentation, different ways using this information could be studied. Instead of overlapping the segmentation results of the baseline model or making a weight sum relation between consecutive frames, a more elaborated relation that uses probabilistic relations can be defined.

The combination of sequential information and enhancement of the probability-map results in an increase of false positive classifications in the form of segmentation noise. There are different ways that can help alleviate this problem, while at the same time raising the overall accuracy of the segmentation. Some of them are:

- Fine tunning the parameters that define the Attention Module. In such a way that, the values that do not reach a minimum after the probability augmentation are pushed to 0 and therefore will not be assigned the target label.
- Using different temporal modules with different sensitivity values. After the computation of the augmented probabilities, the results will be compared and only the classifications with the highest agreement will be output.

- Limiting the work-space of the temporal modules. The augmented segmentation can be applied only to a certain region of interest, such as just the driving-space. This can alleviate the production of false positive classifications and increase the performance of the algorithm.
- Combining this method with a 3D map estimation. After the calculation of the augmented segmentation map, it could be projected onto a 3D map of the environment, assigning labels to the 3D objects and discarding the labels that do not project into any plane (or very far away).

Another extension of this study could be to instead of basing the augmentation on only past frames, an estimated prediction of nearby future states (t+1, t+2 ...) could be calculated and added into the combination. At the same time, given a sequential fully annotated data-set could provide the optimal set of parameters that define the relations between adjacent frames.

On top of that, these extensions could be applied to create an artificial sequential data-set. Using the results of the baseline model with the extension on the temporal domain could be used to generate short annotated sequences. These sequences can then be used to train DeepLearning architectures that embed the temporal behavior on its structure. Deep Learning models such as convLSTM [58], GRFP [17] or the Multiclass semantic video segmentation [19] can be live-trained with these temporal extensions, creating a self-supervised segmentation pipeline.

It could be interesting to study the performance of semantic image segmentation models and its temporal extension from the inference-speed point of view. Although this study was performed on the state-of-the-art semantic image segmentation model (highest in accuracy), other models designed to perform at much higher speed rates can influence the type of solution needed for the temporal analysis.

Finally, finding new applications where this method can be applied. One of the highlights of this study is that slow environments benefit from sequential information, helping to smooth and to complete partial segmentations (figure E.1). It would be interesting to see the effect of these approaches in medical applications where the environment is less dynamic than the navigation one. Other than that, applications with different types of sequential data can also be analyzed under the same suggested approaches.

A Figure comparison: baseline performance

A.1 Vanishing biker

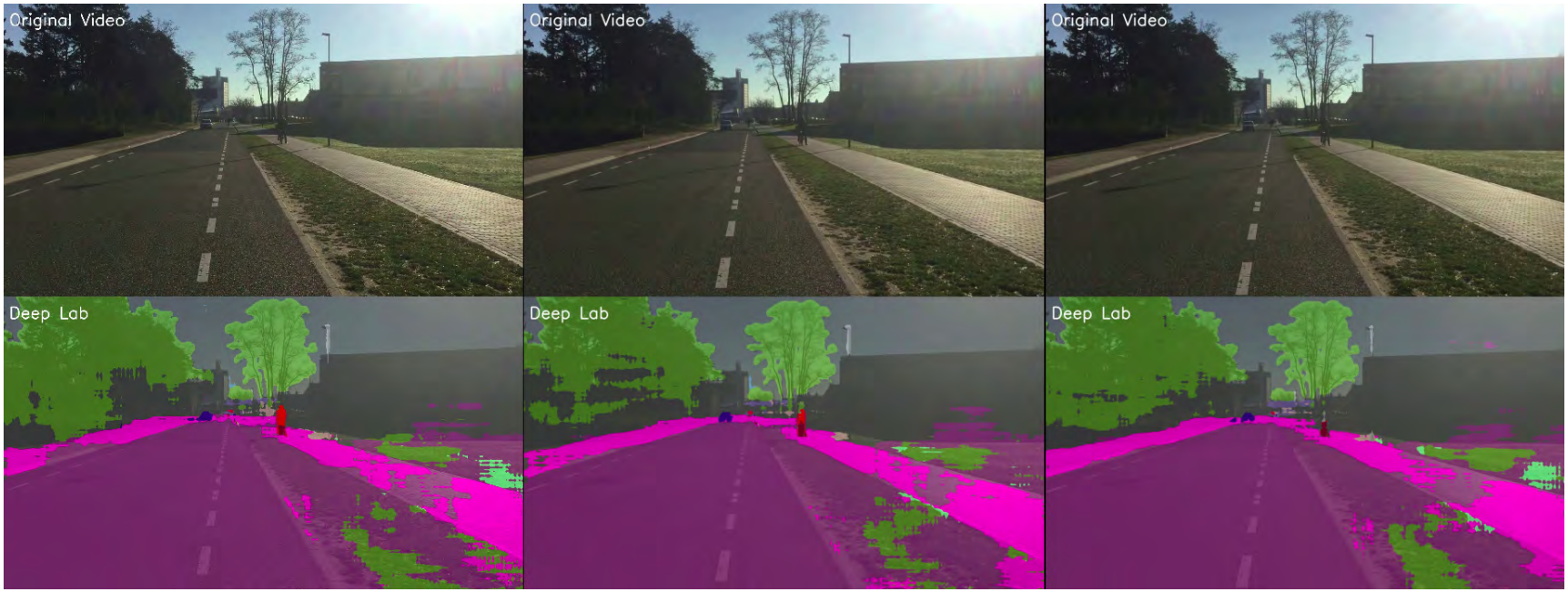

Figure A.1: This figure shows how an instance of DeepLabv3 (pretrained using the Cityscapes dataset) performs during 3 consecutive frames. In this series: the camera is moving towards a cyclist that remains static in the sidewalk, it is observed how the segmentation of each component (vegetation, road, sidewalk and rider) is not consistent over time, with the rider vanishing on the 3rd frame.

A.2 Inconsistent partial segmentation

Figure A.2: This figure shows how an instance of DeepLabv3 (pretrained using the Cityscapes dataset) performs during 3 consecutive frames. In this series: the camera is moving in between a moving crowd, showing in the same way in figure A.1 how although the input images are almost the same for the naked eye, the segmentation changes over time.

B Segmentation Analysis:

B.1 Relative displacement on a 30FPS video

Figure B.1: This figure compares the translation of a moving object during 3 consecutive frames. Each of the two series shown in this figure (bike and truck) correspond to 3 consecutive frames extracted from two videos recorded at 30 fps. The upper series (from left to right) show a subject moving in a close plane horizontal to the camera. The lower series (from left to right) show a truck moving in a far plane horizontal to the camera. The further to the right image for both series is a mask overlap of each target over the 3 consecutive frames (zoomed in), it represents the relative displacement over time of these objects for a video recorded at 30 fps. It is observed how the relative displacement in minimal for both examples. The bike series, introduces how the combination of the history of an object with larger relative displacements may introduce a 'ghosting effect' to the resulting image. Apart from being dependent on the recording speed (fps) and distance to the camera, the relative displacement is also dependent on the relative speed between camera and object (although a speed figure comparison is not elaborated, this was observed along the different tests for this thesis study).

B.2 Inconsistent segmentation overlap

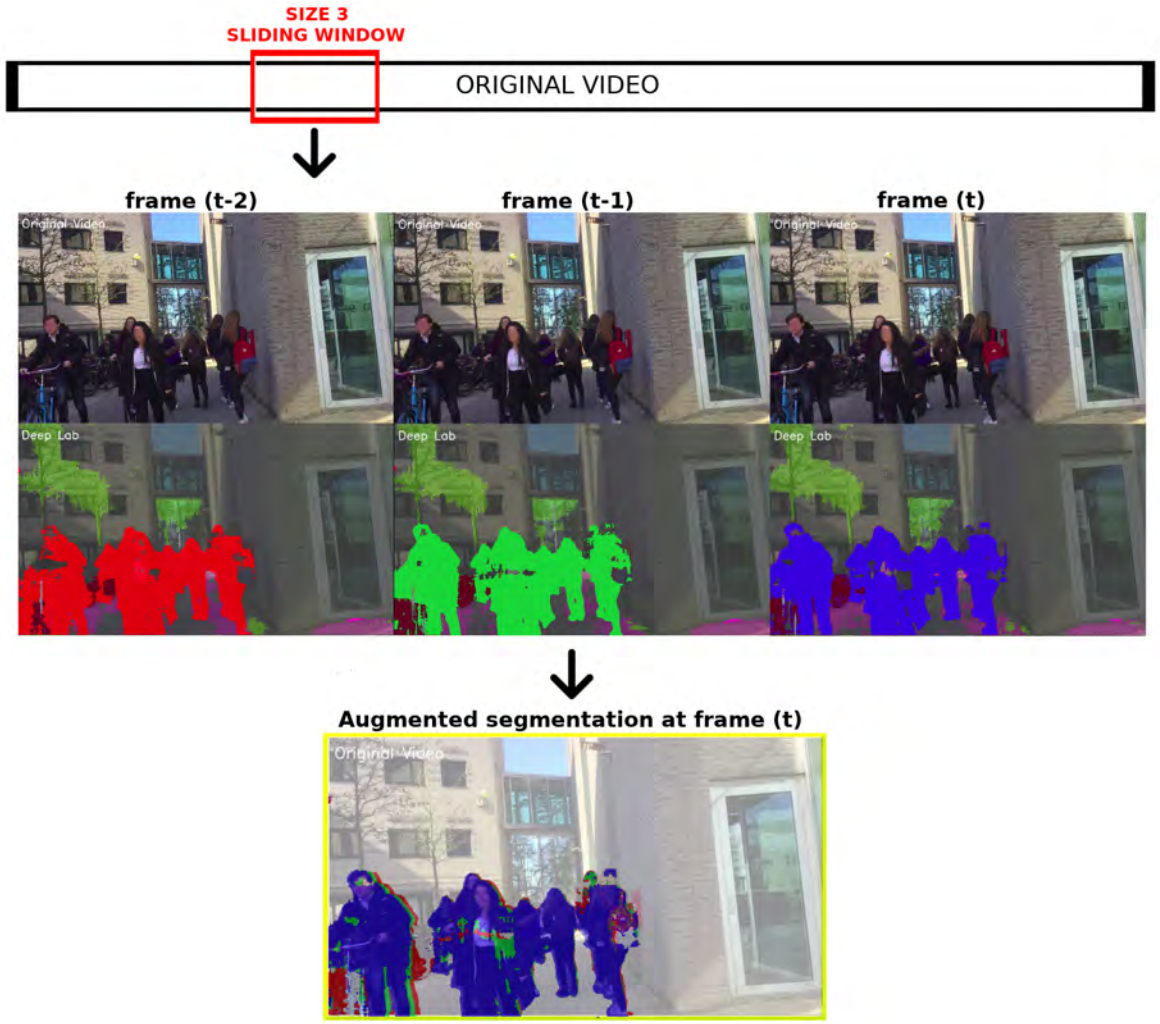

Figure B.2: Conceptual figure that shows how the combination of the segmentation of a particular target (person in this case) of neighbouring frames can be beneficial for the segmentation at the present time. From top to bottom: 3 consecutive frames are extracted from the original video using a sliding window that covers the present frame and 2 other consecutive frames in the past (the extraction can be done for more than 3 frames); the baseline segmentation model (DeepLabv3 on Cityscapes) is applied to each one of these frames, the person label is highlighted in a different color (red, green, blue) for each frame in order to improve the visualization; the hypothetical augmented segmentation output is constructed by overlapping the segmentation of the 3 consecutive frames in the sliding window (bottom figure). It can be observed how the Augmented segmentation at frame (t) contains parts of the segmentation of each of the examined frames. The results show how the wholes in the segmentation of the man in the left and the woman in the middle at time (t) are 'filled' by the segmentation of frames at times (t-2) and (t-1). This augmentation approach is regarded as Image Buffer in chapters 3 to 6.

C Attention Module parameter selection

This appendix contains different figures that support the assessment of the weight values in equation 4.2. In addition to the weights (I_0 to I_3) an extra parameter T is used as a threshold to push the augmented probabilities to zero when a minimum value is not reached, this limits the production of false positive classifications (appendix 2). These test were evaluated in 2 clips that appeared to be challenging for the baseline classification, and one that presents optimal conditions for the segmentation. The procedure followed was increasing the value of each weight, giving more emphasis to the recent frames, until a satisfactory result was achieved. The clips belong to the video [52].

C.1 First test:

I_0	I_1	I_2	I_3	T
1	1	1	1	0

Table C.1: First test's parameters for equation 4.2.

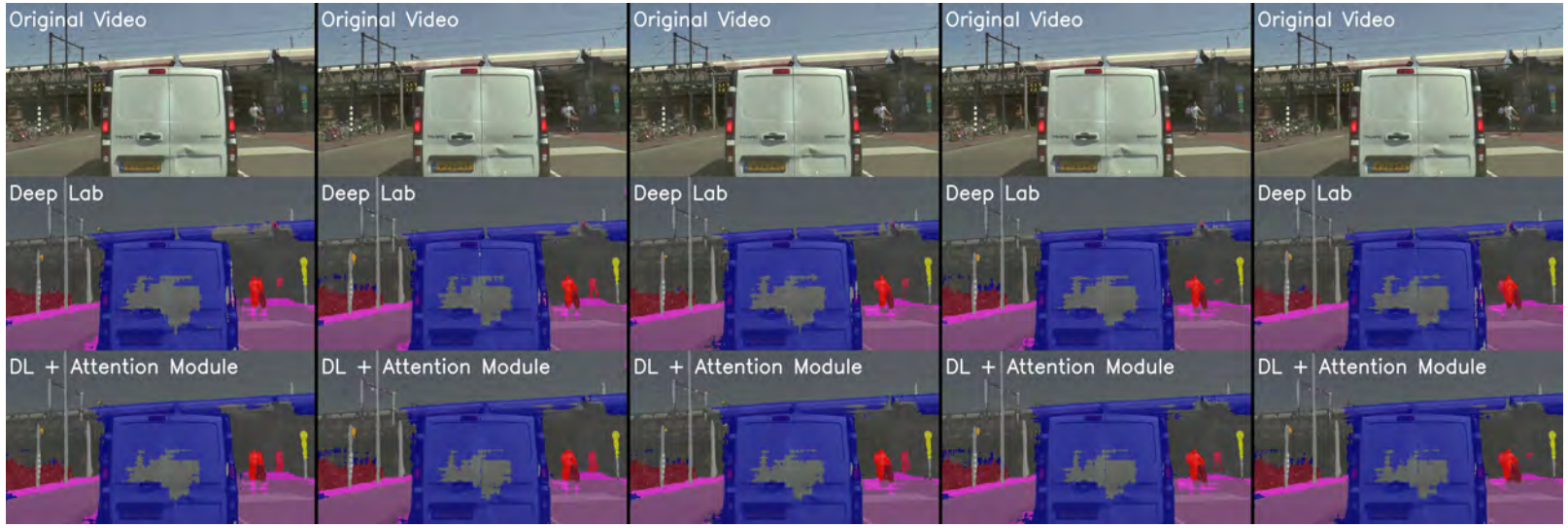

Figure C.1: Segmentation results for the AM parameters in table C.1. It can be observed how the hole in the baseline segmentation of the van is starting to get filled in the DL+AM approach. Likewise DL+AM obtains a better segmentation of the people in the background that are completely missed in the last frame of the series.

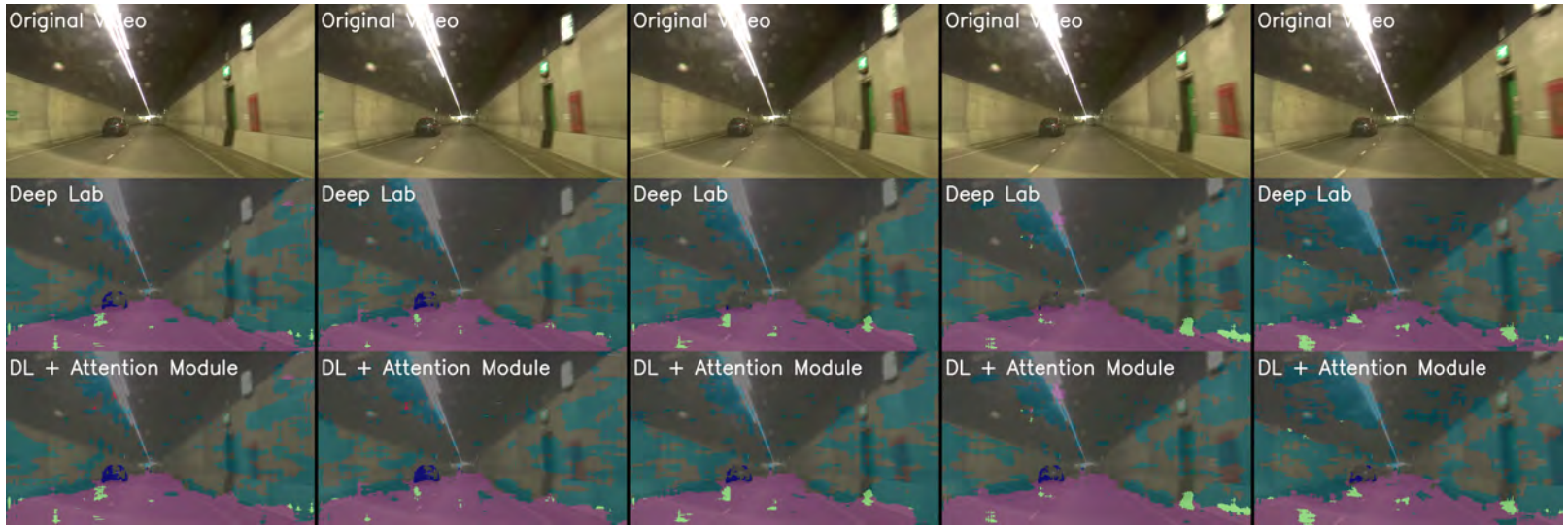

Figure C.2: Segmentation results for the AM parameters in table C.1. In this case the baseline segmentation model completely misses the car at the end of the serie. In the other hand, the DL+AM approach is able to detect, although not fully, the car in all the frames of the serie.

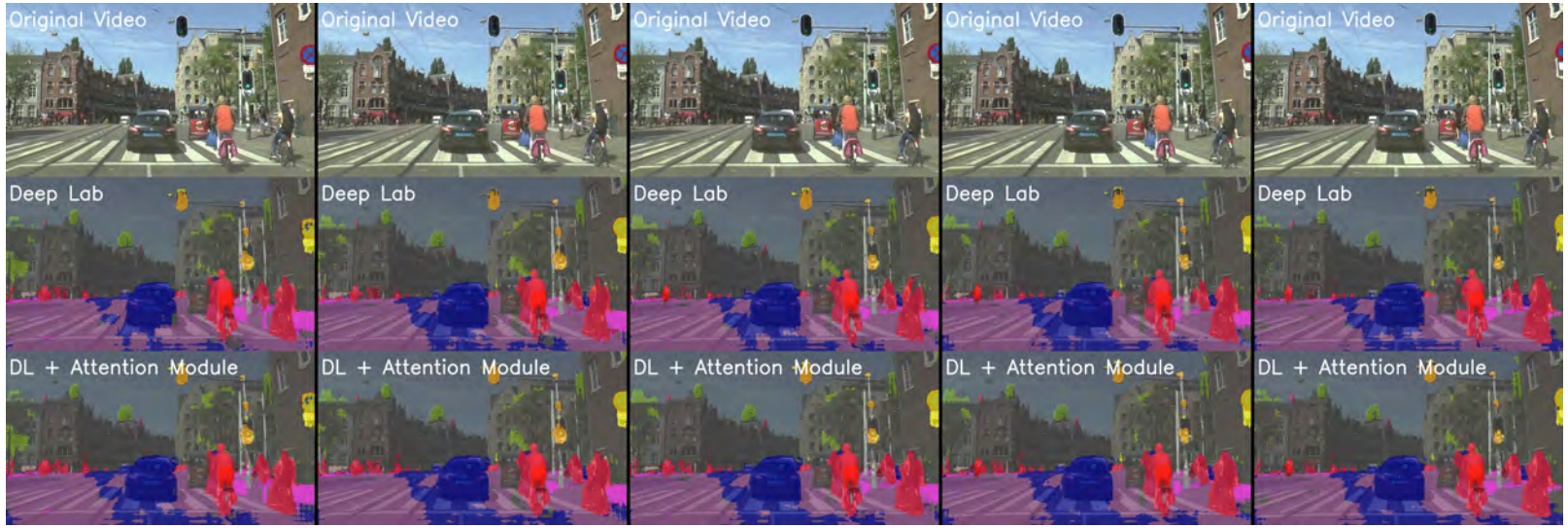

Figure C.3: Segmentation results for the AM parameters in table C.1. This figure shows how the weighted sum of the Attention module increases the number of false classifications in the road.

C.2 Second test:

I_0	I_1	I_2	I_3	T
3	2	1	1	0

Table C.2: Second test's parameters for equation 4.2.

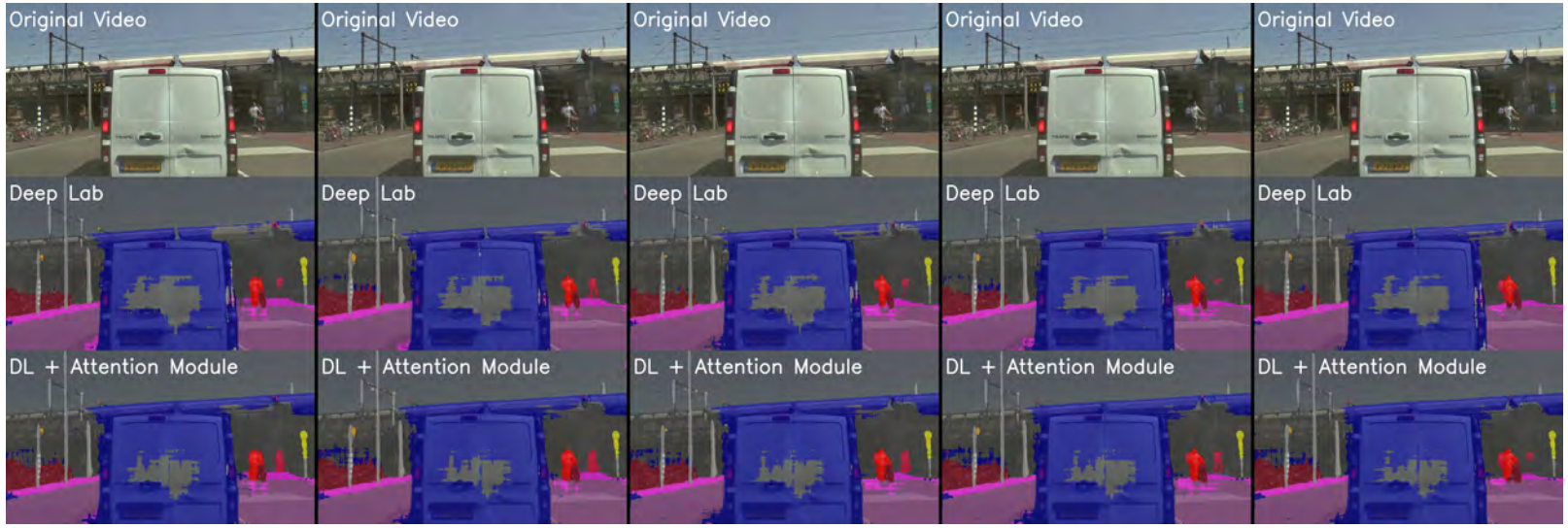

Figure C.4: Segmentation results for the AM parameters in table C.2. It can be observed how the hole in the DL+AM segmentation of the van is fuller than the one in figure C.1. There is also an increase in the area of classification of the people in the background compared to the obtained by using the parameters in table C.1.

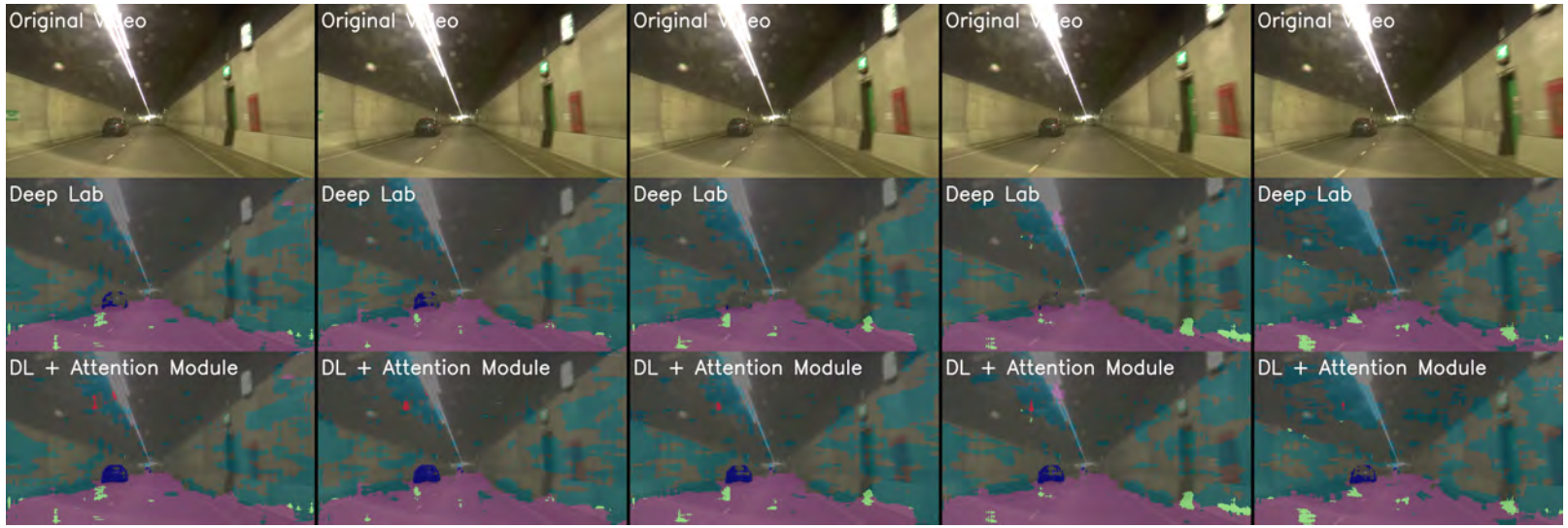

Figure C.5: Segmentation results for the AM parameters in table C.2. DL+AM is able to detect the car at all frames. However, these parameters start producing wrong classifications in the upper part of the figure compared to the results presented in figure C.2.

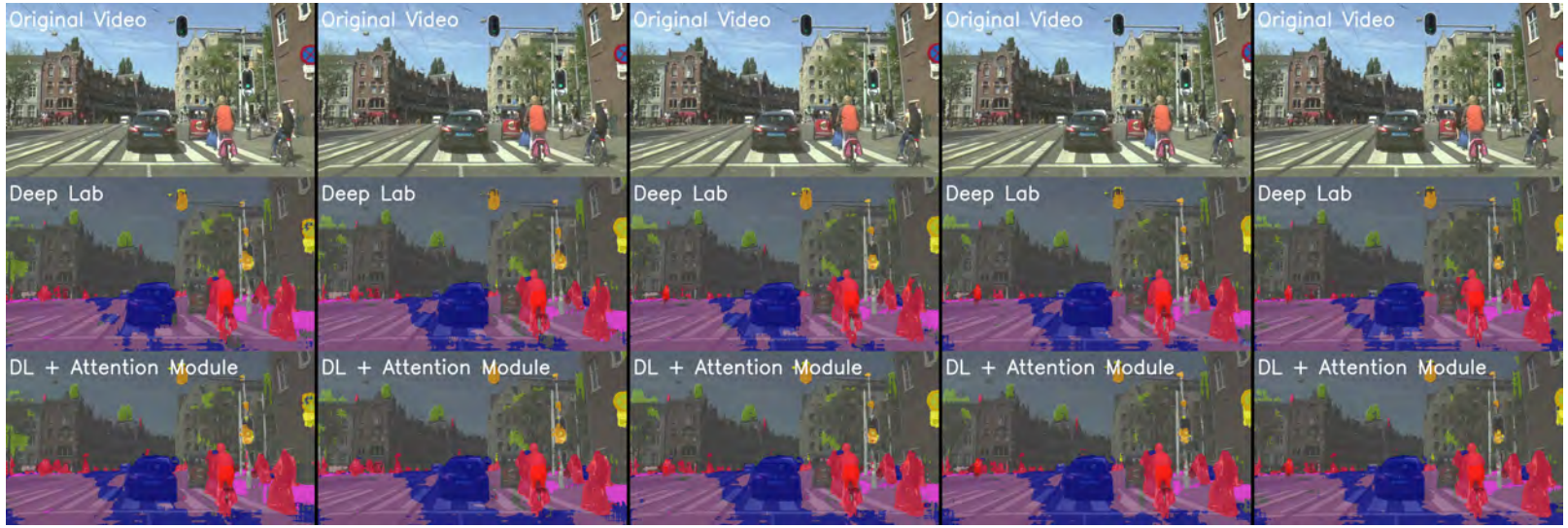

Figure C.6: Segmentation results for the AM parameters in table C.2. DL+AM produces a bigger amount of false positive classifications than the previous results presented in figure C.3.

C.3 Third test:

I_0	I_1	I_2	I_3	T
4	3	2	1	0

Table C.3: Third test's parameters for equation 4.2.

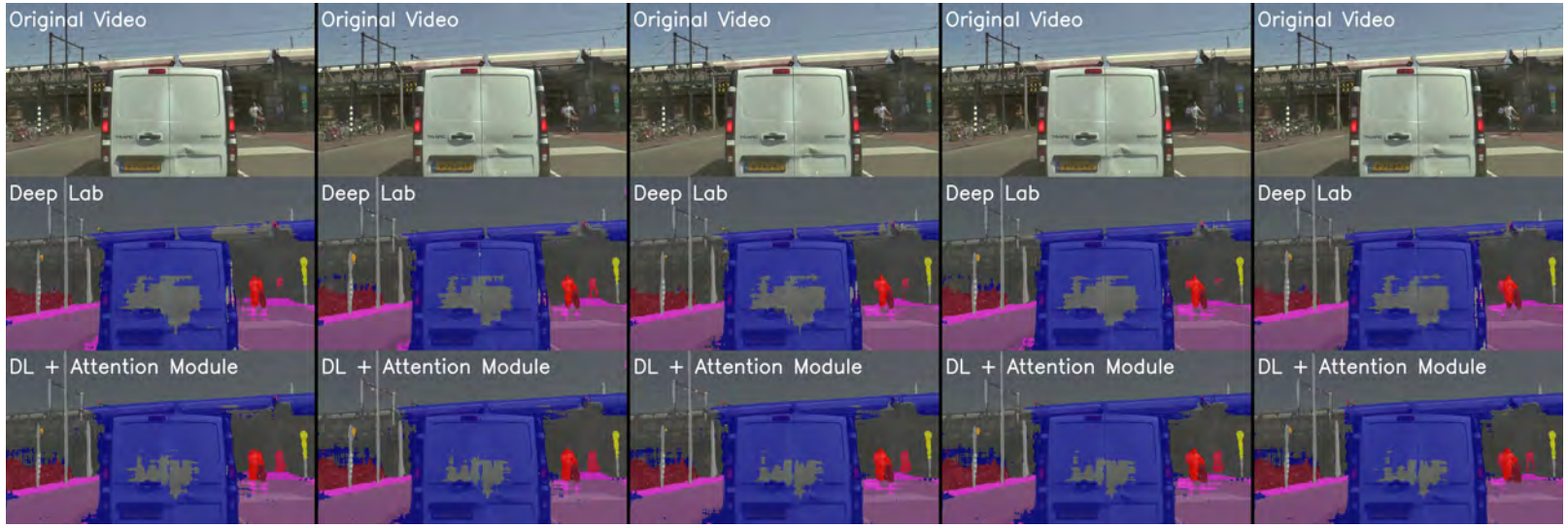

Figure C.7: Segmentation results for the AM parameters in table C.3. In this figure, it is evident how an increase on the weights that affect the final segmentation. The filling effect in the hole of the van is noticiable and the people in the background are detected almost completely for all the frames.

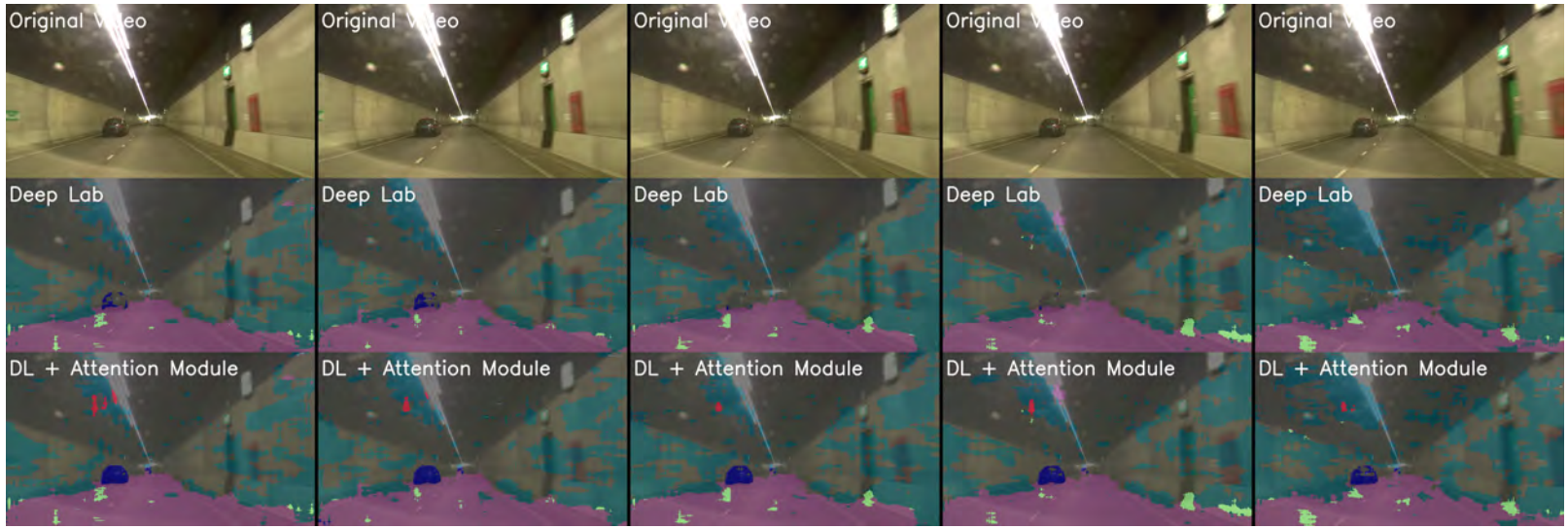

Figure C.8: Segmentation results for the AM parameters in table C.3. In the figure, DL+AM is able to detect the car at all frames. However, the wrong classifications in the upper part of the image start increasing with respect to the previous set of parameters figure C.8, this effect can be attenuated by means of a threshold that limits the augmented probabilities to a minimum value (table C.4).

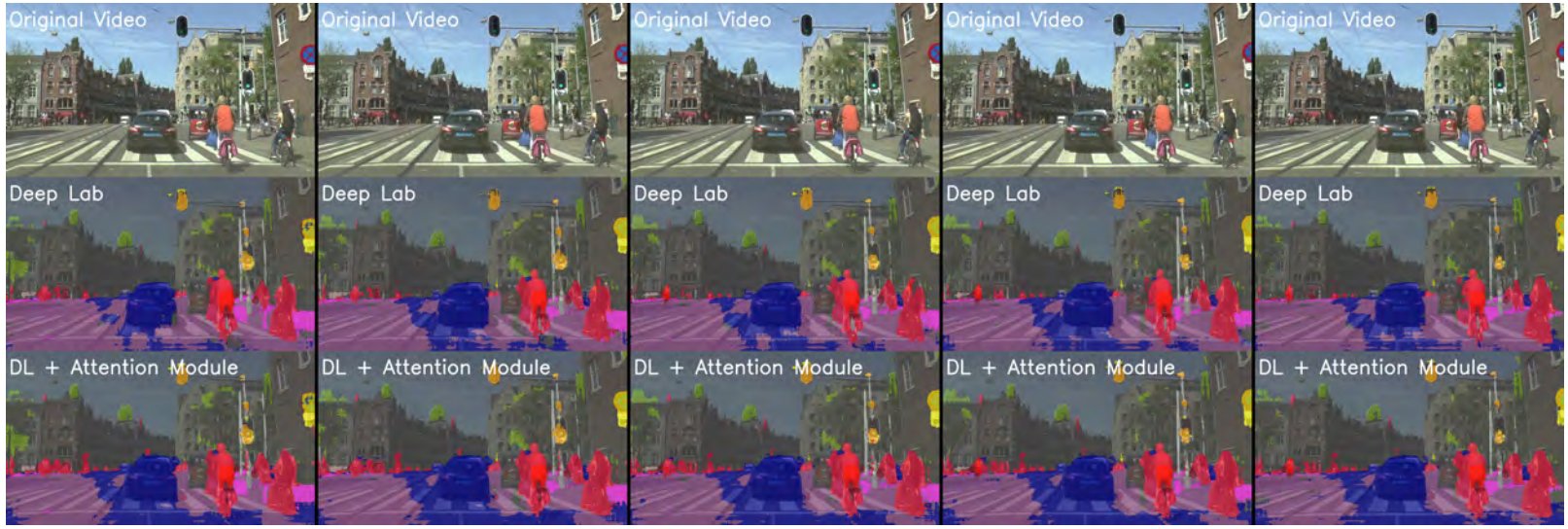

Figure C.9: Segmentation results for the AM parameters in table C.3. This figure shows how the increment of the weights (table C.3) produces an increase in the false positives classification in the road of DL+AM compared to the baseline segmentation.

C.4 Fourth test: Increasing the threshold value

Compared to the rest of the tests included in this appendix, this combination of parameters obtains the best qualitative performance of the algorithm for the three evaluated videos. For this reason, this combination will be used in the experiments' chapter.

I_0	I_1	I_2	I_3	T
4	3	2	1	1

Table C.4: Fourth test's parameters for equation 4.2.

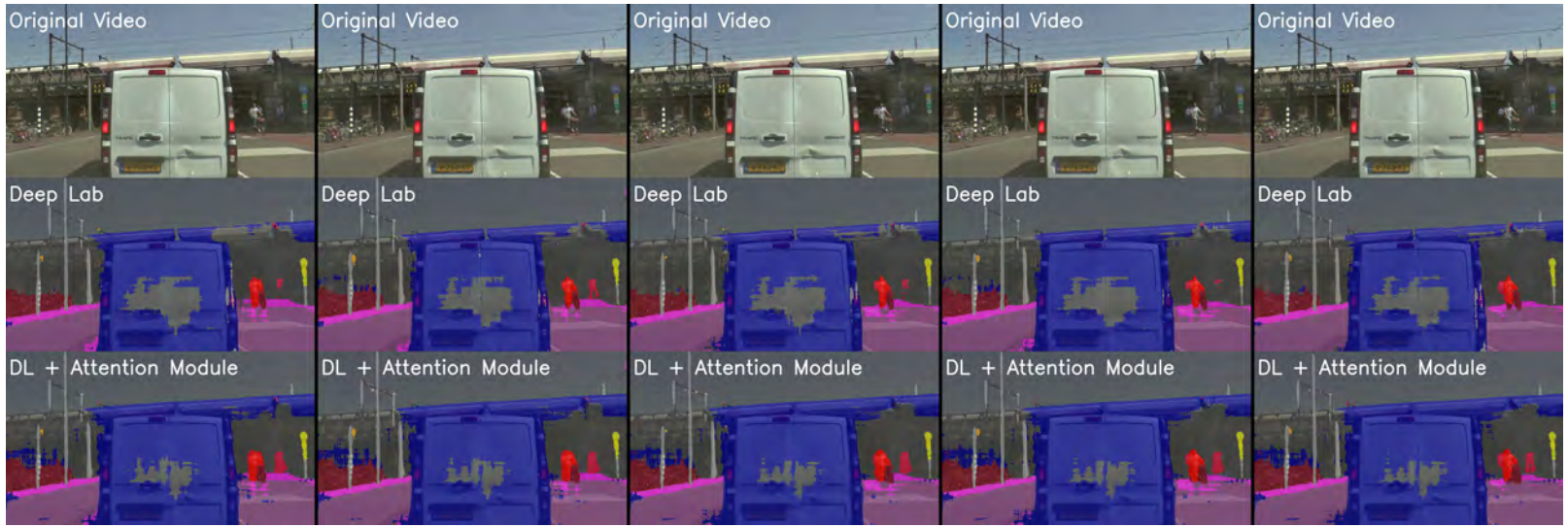

Figure C.10: Segmentation results for the AM parameters in table C.4. The segmentation level achieved by DL+AM for the van and the people in the background is higher than the one obtained by the baseline model.

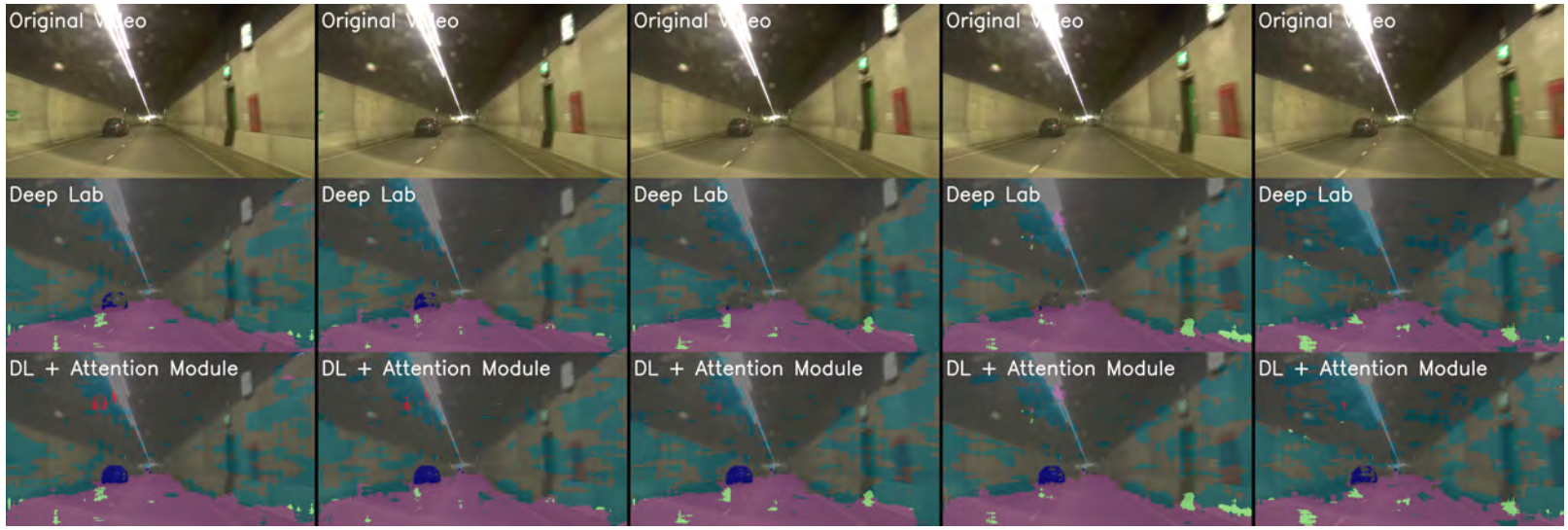

Figure C.11: Segmentation results for the AM parameters in table C.4. DL+AM is able to detect the car at all frames. In addition, the threshold parameter allows to control the amount of wrong classifications in the upper part of each frame.

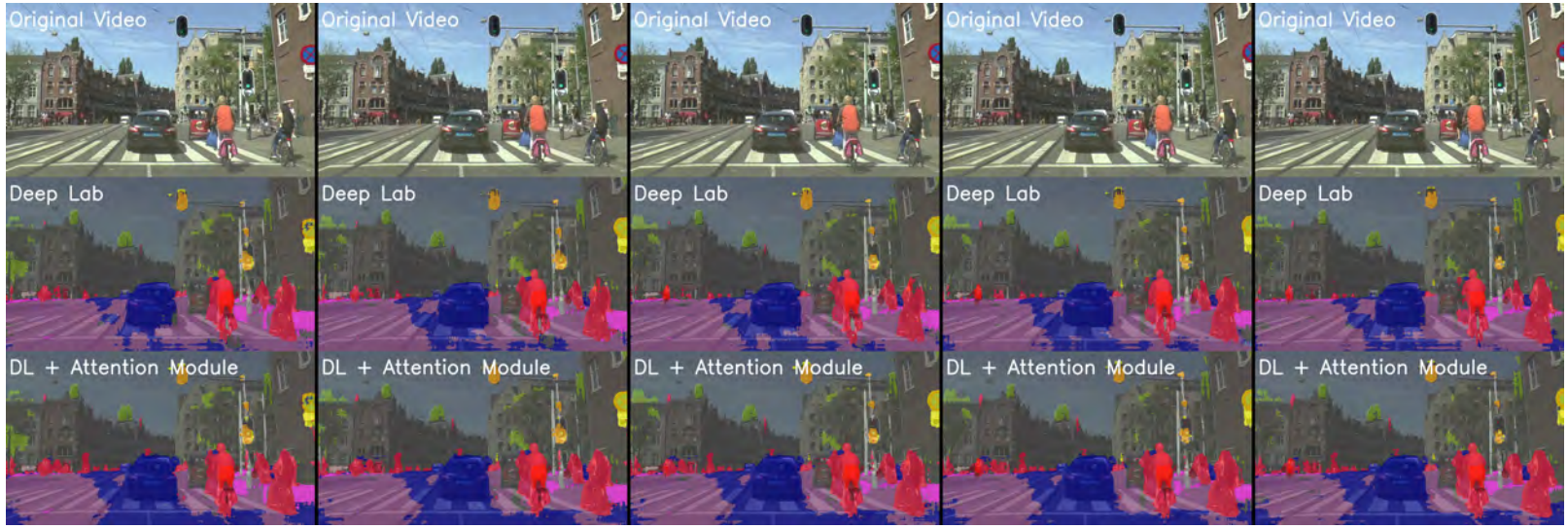

Figure C.12: Segmentation results for the AM parameters in table C.4. The threshold allows to control the amount of wrong classifications obtained by the DL+AM approach in the middle of the road.

D Appendix 1

D.1 Metrics' Graphs

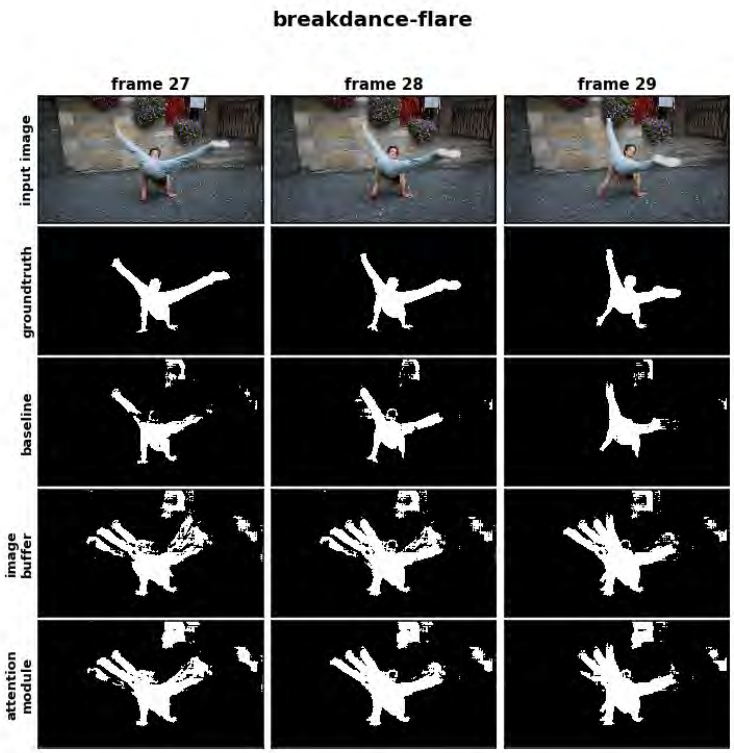

Figure D.1: Figure comparison of the segmentation of 3 consecutive frames and its corresponding groundtruth annotation. The evaluation was performed under: DeepLabv3 (sec 3.1.3), Image Buffer (sec 4.1) and Attention Module (sec 4.2). Sequence source: DAVIS dataset [51] - 'breakdance-flare' category.

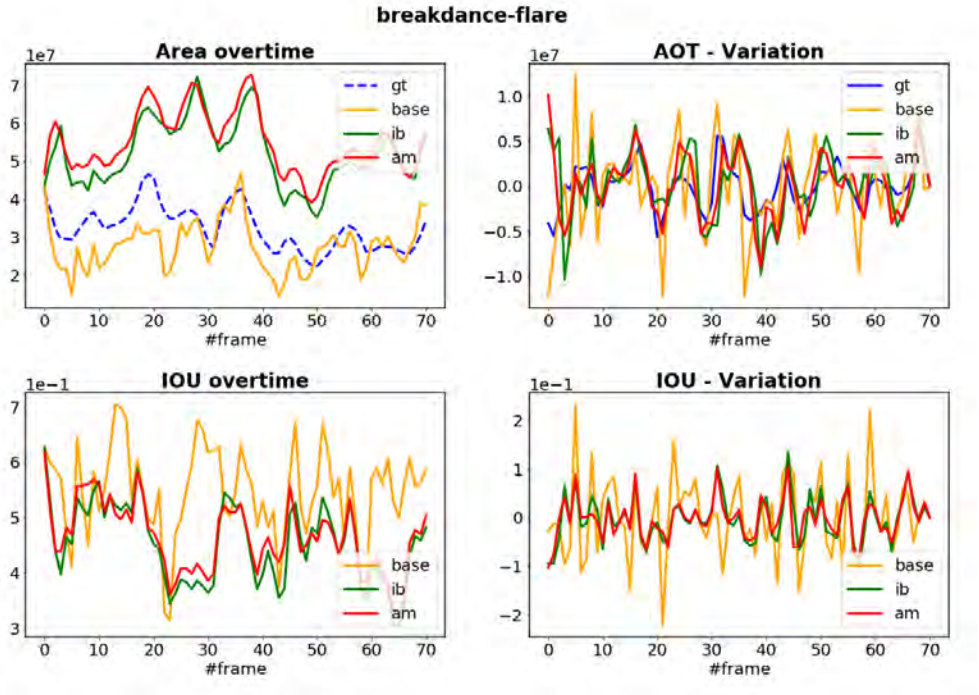

Figure D.2: Graph evaluation of the 'breakdance-flare' category (DAVIS dataset [51]) using the metrics defined in section 3.2.1.

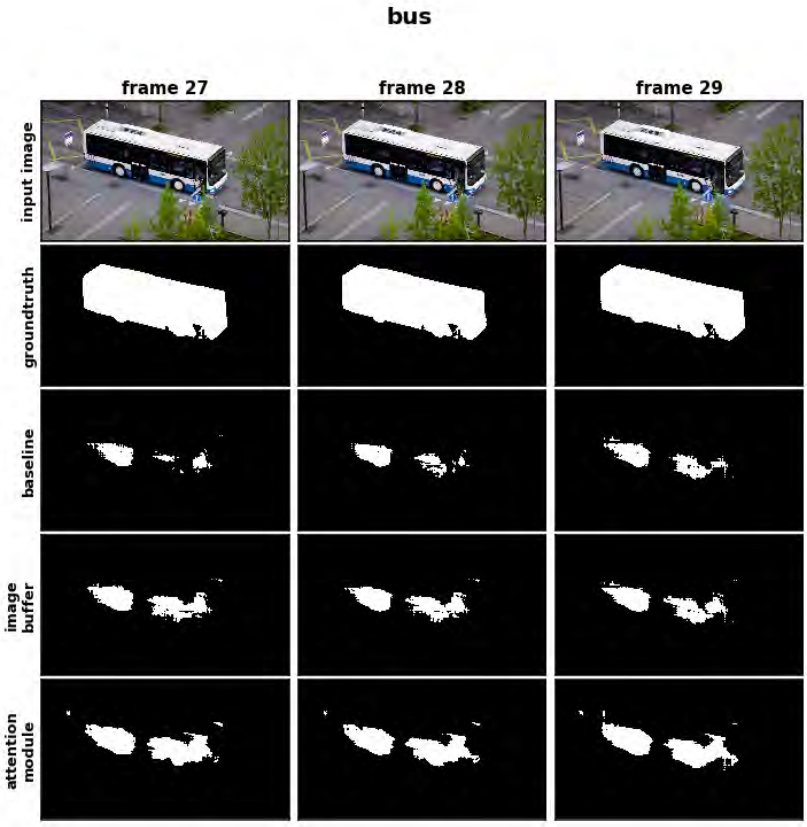

Figure D.3: Figure comparison of the segmentation of 3 consecutive frames and its corresponding groundtruth annotation. The evaluation was performed under: DeepLabv3 (sec 3.1.3), Image Buffer (sec 4.1) and Attention Module (sec 4.2). Sequence source: DAVIS dataset [51] - 'bus' category.

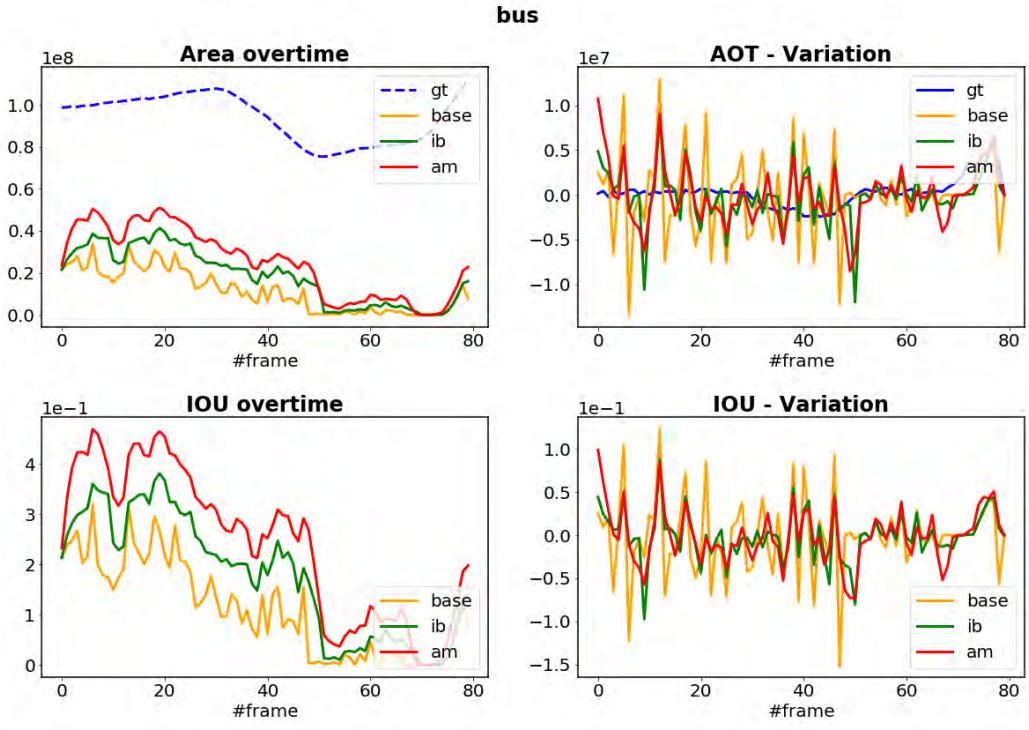

Figure D.4: Graph evaluation of the 'bus' category (DAVIS dataset [51]) using the metrics defined in section 3.2.1.

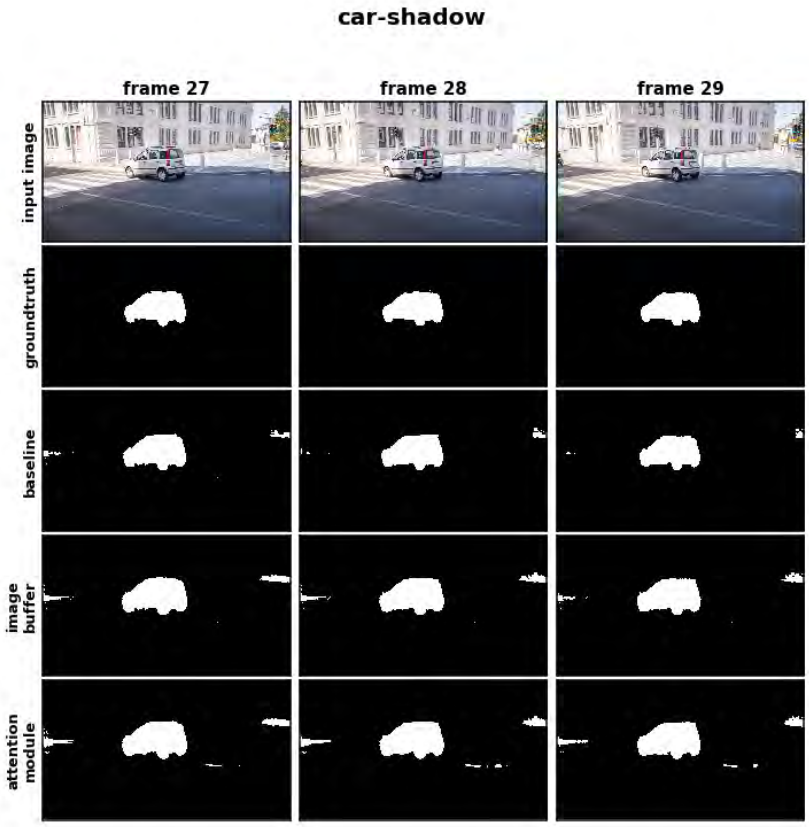

Figure D.5: Figure comparison of the segmentation of 3 consecutive frames and its corresponding groundtruth annotation. The evaluation was performed under: DeepLabv3 (sec 3.1.3), Image Buffer (sec 4.1) and Attention Module (sec 4.2). Sequence source: DAVIS dataset [51] - 'car-shadow' category.

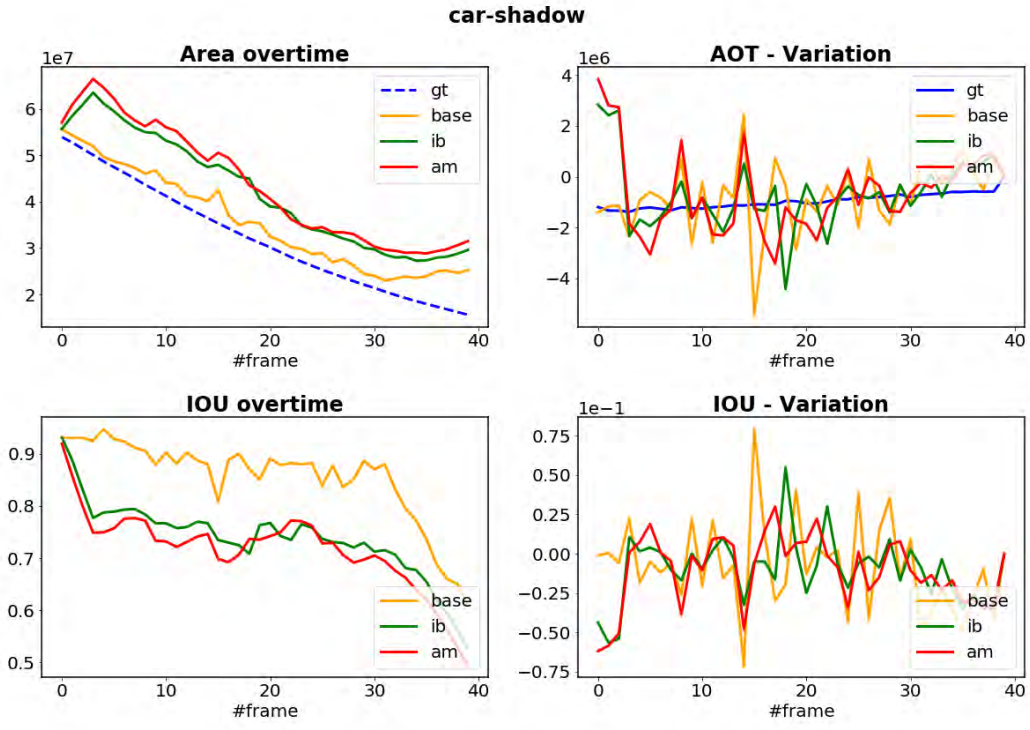

Figure D.6: Graph evaluation of the 'car-shadow' category (DAVIS dataset [51]) using the metrics defined in section 3.2.1.

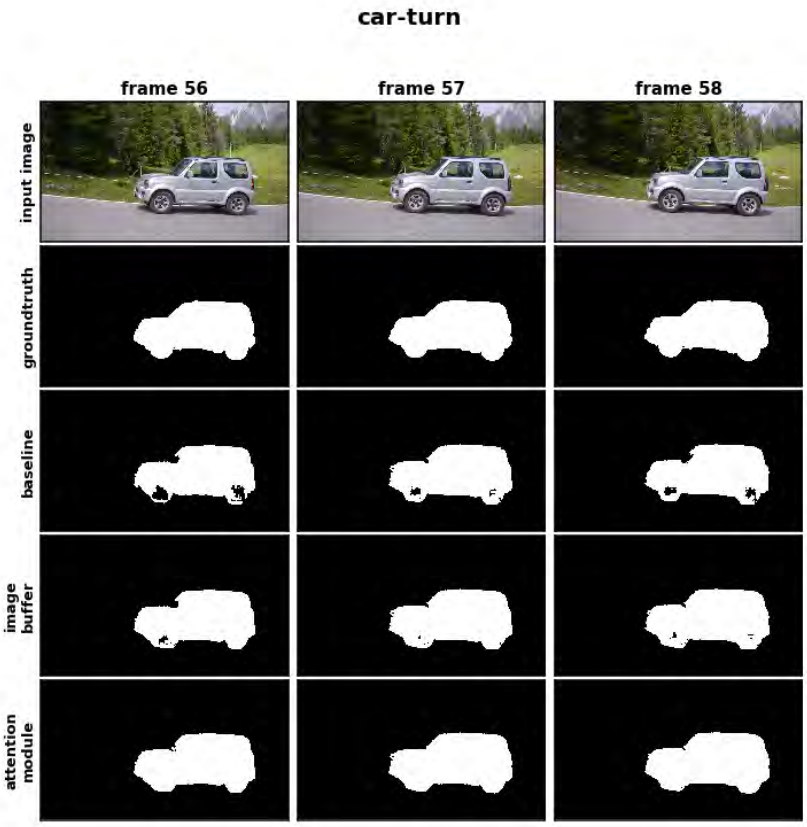

Figure D.7: Figure comparison of the segmentation of 3 consecutive frames and its corresponding groundtruth annotation. The evaluation was performed under: DeepLabv3 (sec 3.1.3), Image Buffer (sec 4.1) and Attention Module (sec 4.2). Sequence source: DAVIS dataset [51] - 'car-turn' category.

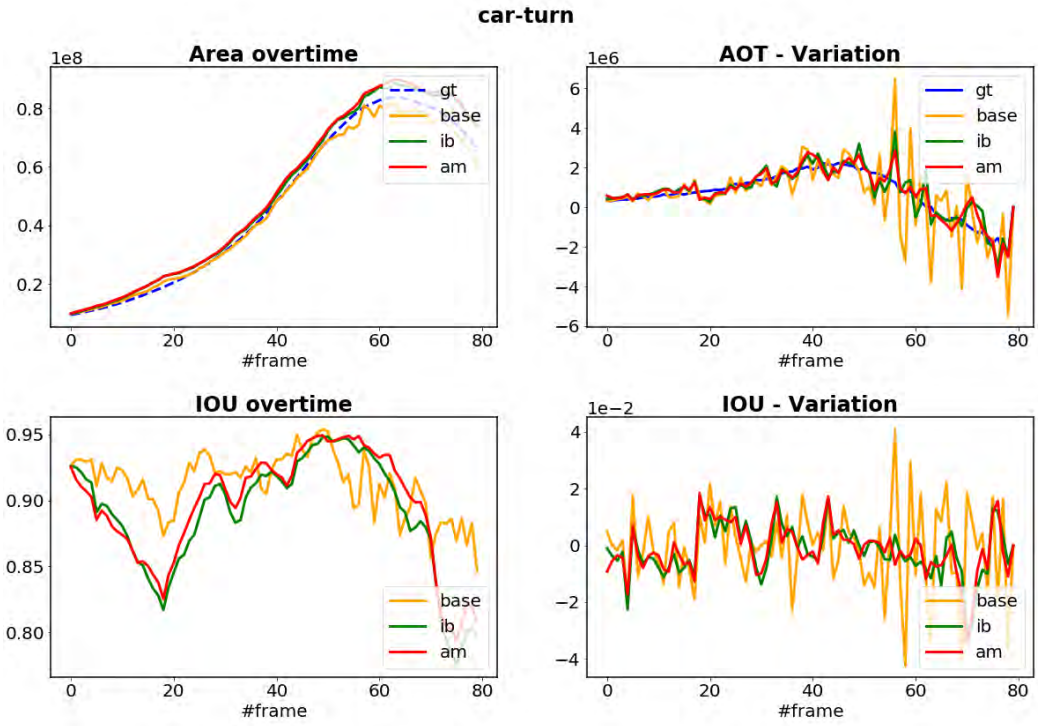

Figure D.8: Graph evaluation of the 'car-turn' category (DAVIS dataset [51]) using the metrics defined in section 3.2.1.

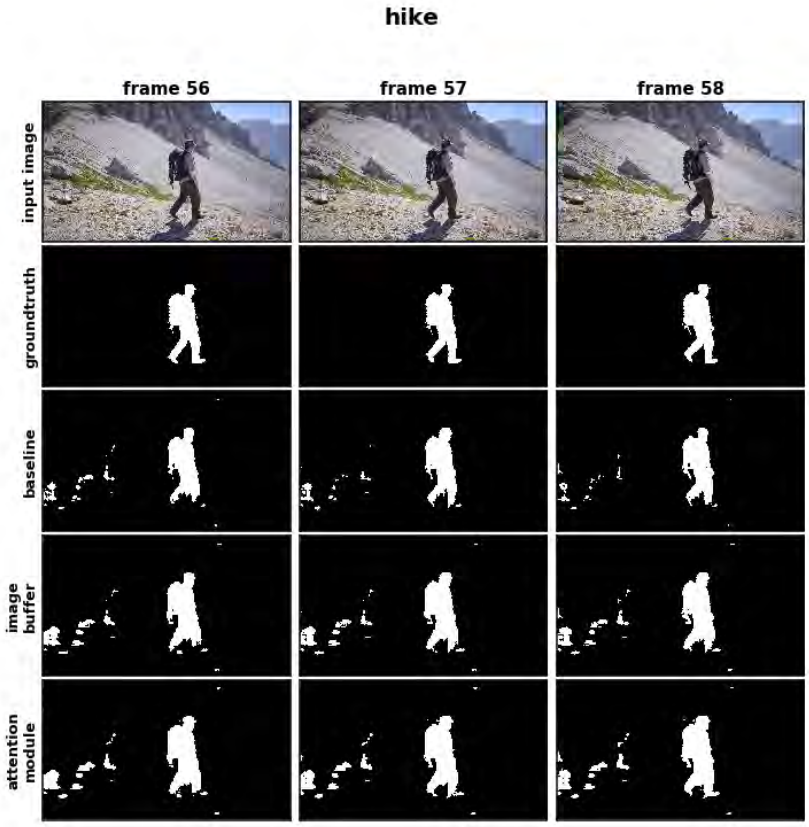

Figure D.9: Figure comparison of the segmentation of 3 consecutive frames and its corresponding groundtruth annotation. The evaluation was performed under: DeepLabv3 (sec 3.1.3), Image Buffer (sec 4.1) and Attention Module (sec 4.2). Sequence source: DAVIS dataset [51] - 'hike' category.

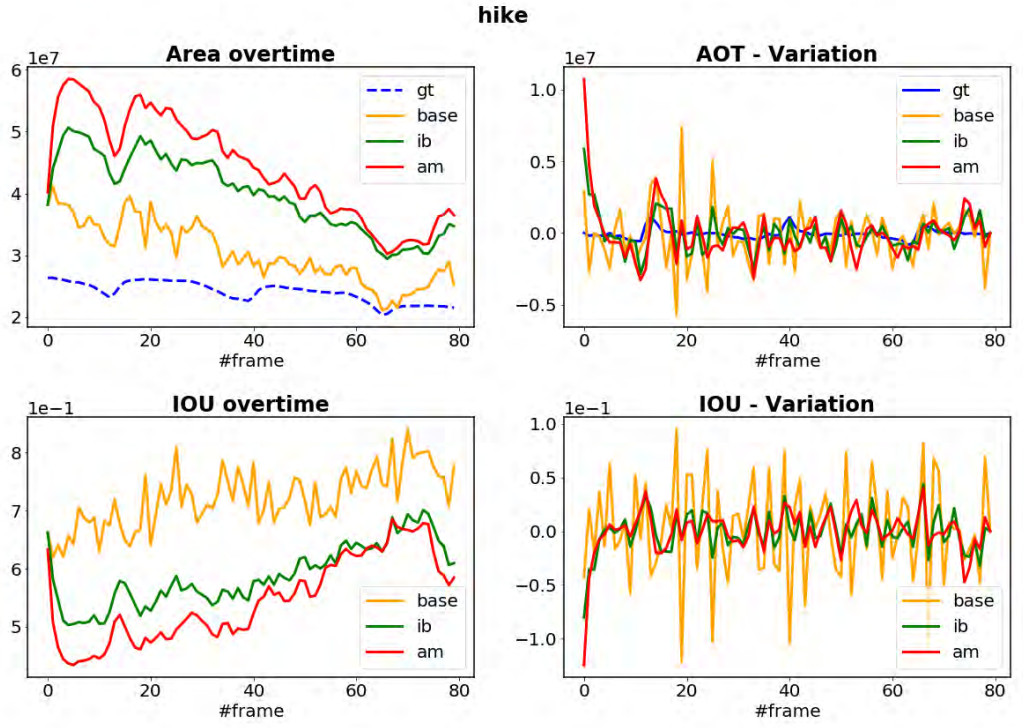

Figure D.10: Graph evaluation of the 'hike' category (DAVIS dataset [51]) using the metrics defined in section 3.2.1.

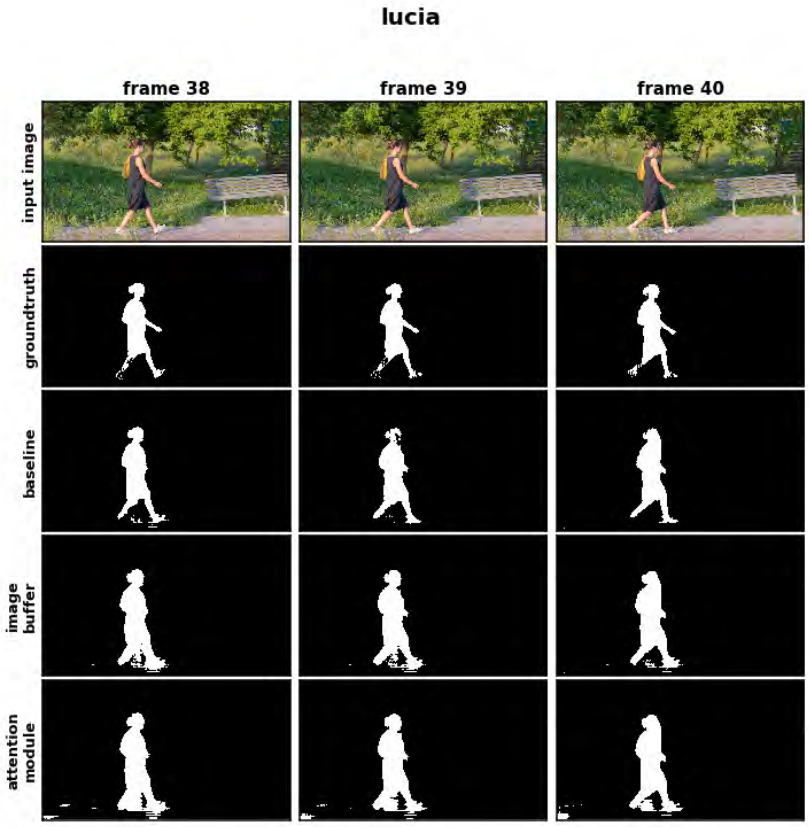

Figure D.11: Figure comparison of the segmentation of 3 consecutive frames and its corresponding groundtruth annotation. The evaluation was performed under: DeepLabv3 (sec 3.1.3), Image Buffer (sec 4.1) and Attention Module (sec 4.2). Sequence source: DAVIS dataset [51] - 'lucia' category.

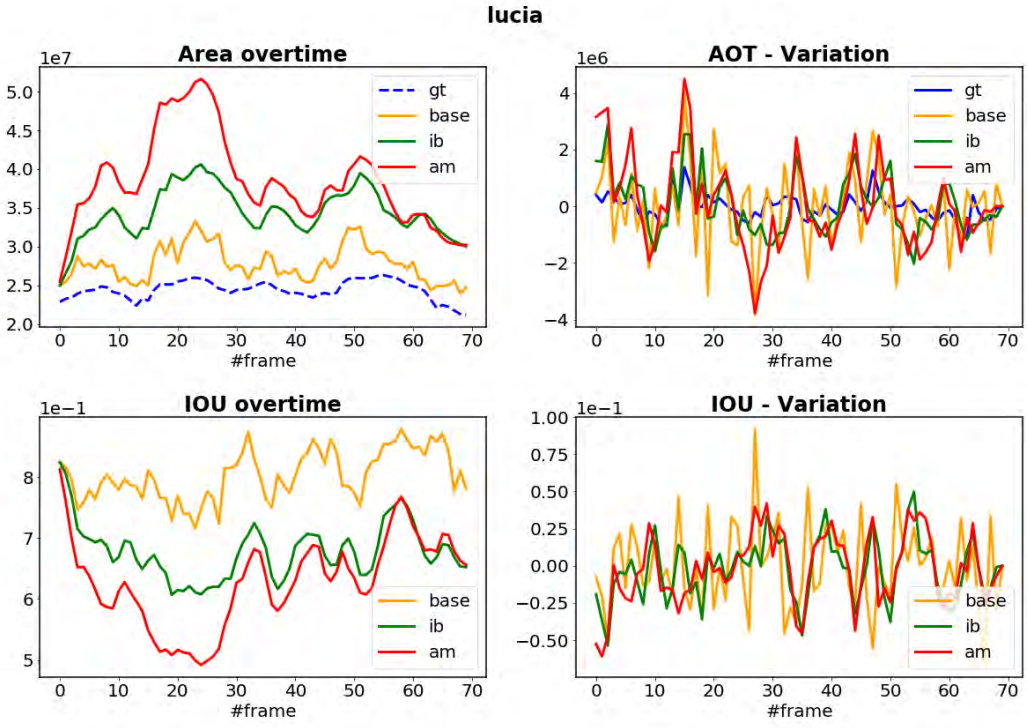

Figure D.12: Graph evaluation of the 'lucia' category (DAVIS dataset [51]) using the metrics defined in section 3.2.1.

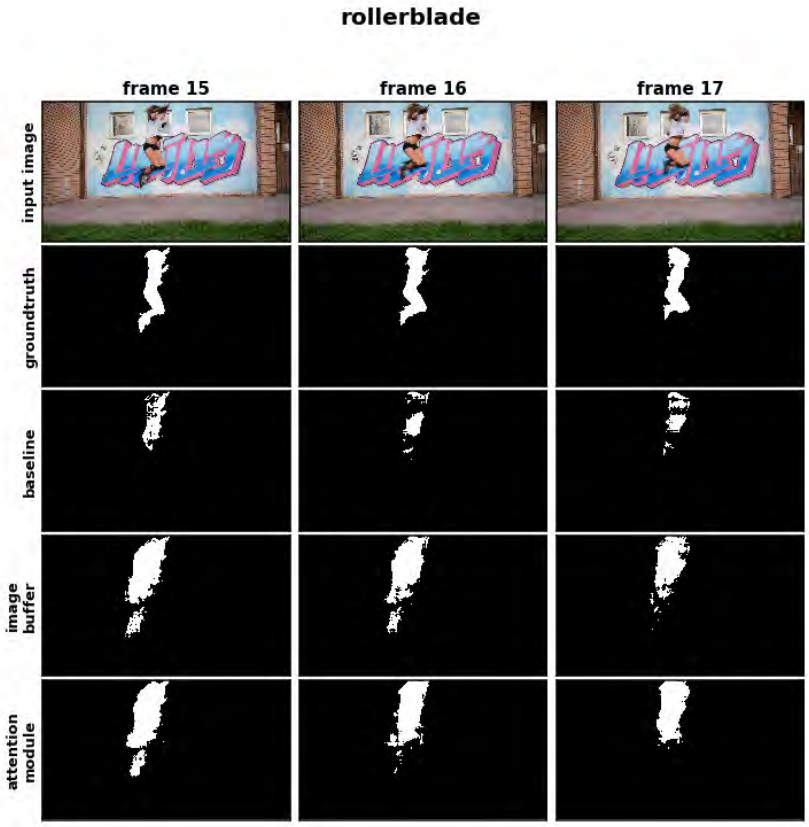

Figure D.13: Figure comparison of the segmentation of 3 consecutive frames and its corresponding groundtruth annotation. The evaluation was performed under: DeepLabv3 (sec 3.1.3), Image Buffer (sec 4.1) and Attention Module (sec 4.2). Sequence source: DAVIS dataset [51] - 'rollerblade' category.

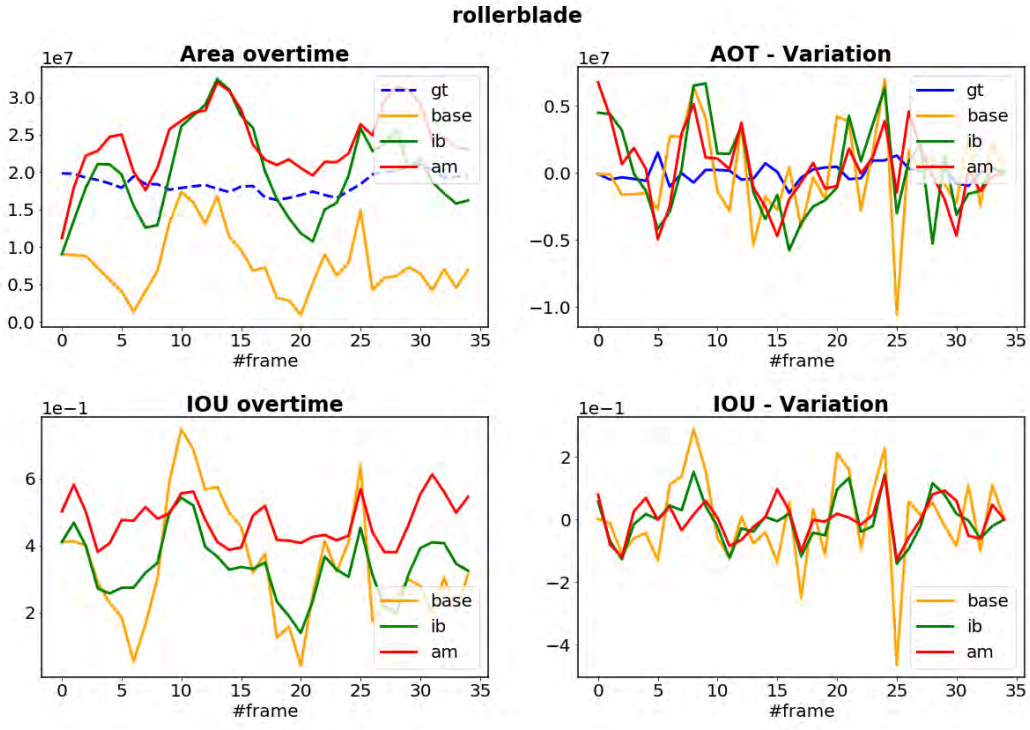

Figure D.14: Graph evaluation of the 'rollerblade' category (DAVIS dataset [51]) using the metrics defined in section 3.2.1.

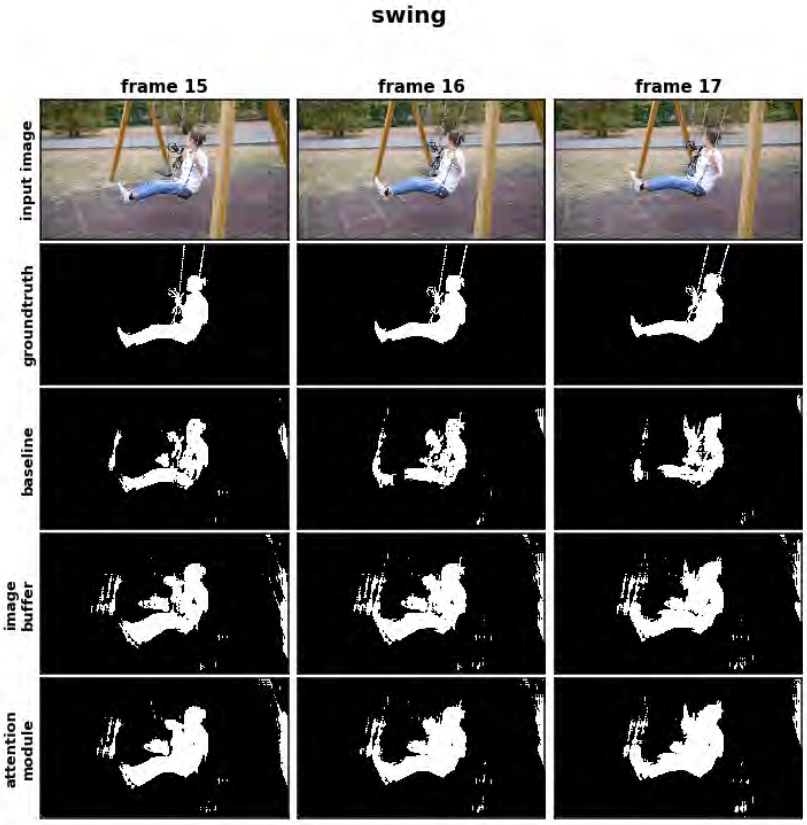

Figure D.15: Figure comparison of the segmentation of 3 consecutive frames and its corresponding groundtruth annotation. The evaluation was performed under: DeepLabv3 (sec 3.1.3), Image Buffer (sec 4.1) and Attention Module (sec 4.2). Sequence source: DAVIS dataset [51] - 'swing' category.

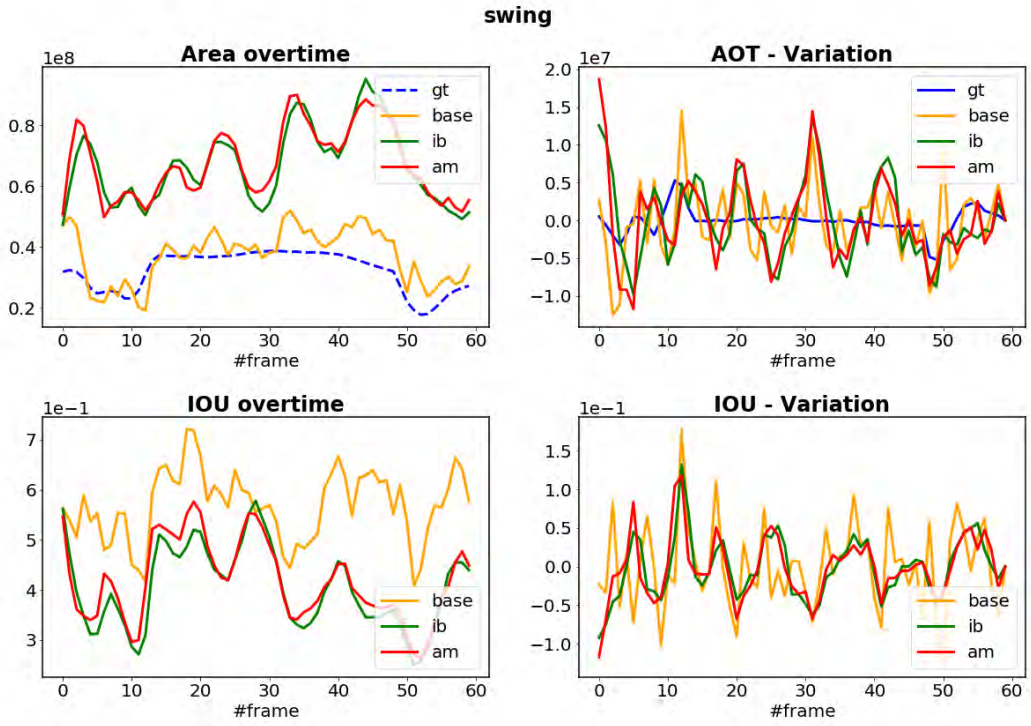

Figure D.16: Graph evaluation of the 'swing' category (DAVIS dataset [51]) using the metrics defined in section 3.2.1.

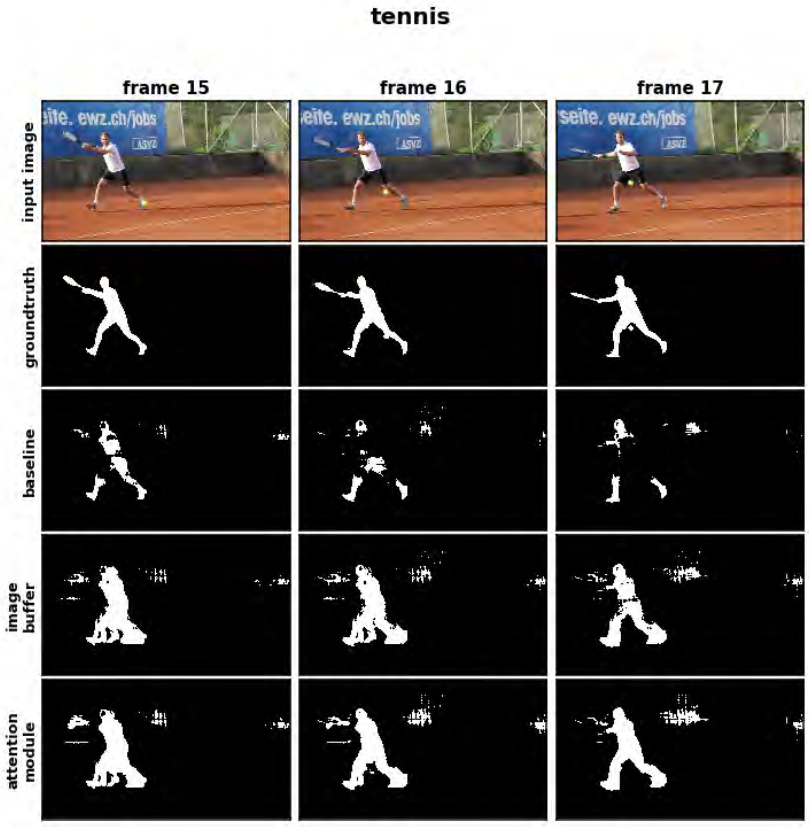

Figure D.17: Figure comparison of the segmentation of 3 consecutive frames and its corresponding groundtruth annotation. The evaluation was performed under: DeepLabv3 (sec 3.1.3), Image Buffer (sec 4.1) and Attention Module (sec 4.2). Sequence source: DAVIS dataset [51] - 'tennis' category.

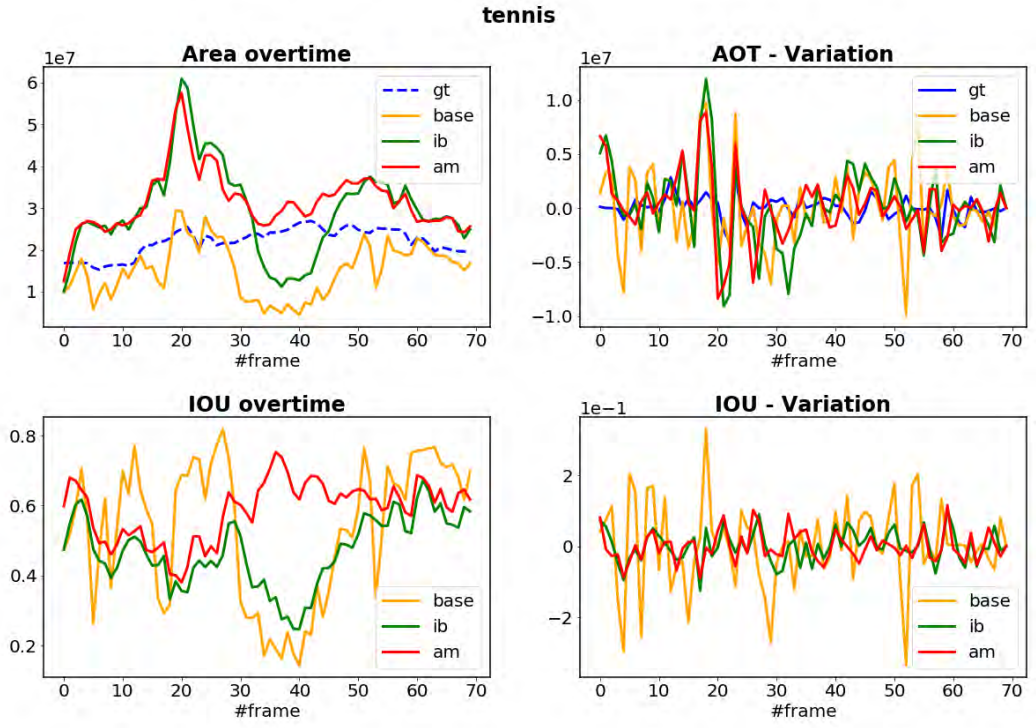

Figure D.18: Graph evaluation of the 'tennis' category (DAVIS dataset [51]) using the metrics defined in section 3.2.1.

E Qualitative experiments

E.1 Citadel - University of Twente

This video was taken on a normal day between lectures at the University of Twente. It was chosen to evaluate the performance of these methods in an environment rich on features with different targets present at the same time. Figure shows how both of the extensions help completing the partial segmentation of some targets.

Figure E.1: Figure comparison of the segmentation of three consecutive frames for a video sequence recorded outdoors in the presence of multiple target categories. It can be observed how the baseline segmentation (DeepLab) is able to detect most of the subjects and how the suggested extensions help completing the segmentation.

E.2 Caree (modified) - University of Twente

This experiment recreates the unfavorable scenario in which the system goes blind for a fraction of a second. It can be observed how both, the Image Buffer and Attention Module are able to maintain the information of the image even when the input frame is lost.

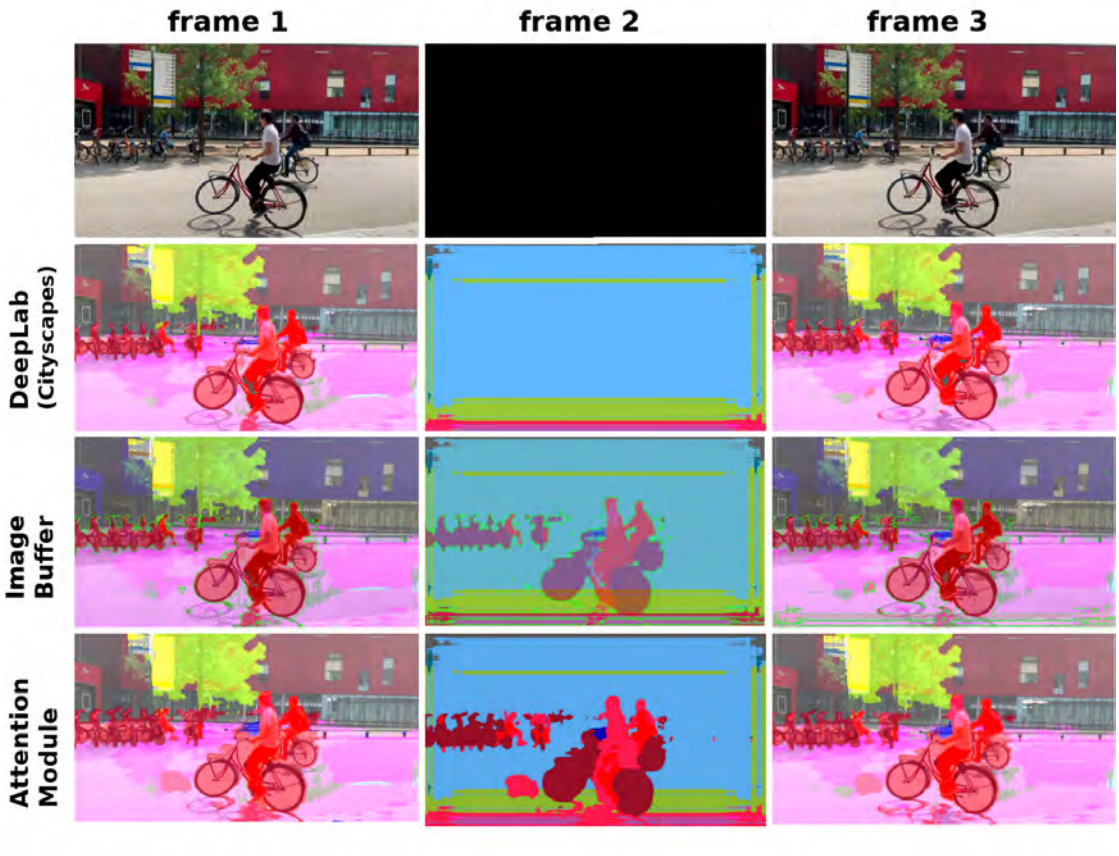

Figure E.2: Figure comparison of the segmentation of three consecutive frames for a video sequence where a frame in between has been removed. It can be observed how a frame by frame semantic segmentation of the sequence is not aware of the present objects when the input image goes missing (DeepLab). On the other hand, a combination of past frames for the segmentation of the present allows to hold information about the environment for a short amount of time.

E.3 Driving in a tunnel

This video [52] contains a sequence of frames in which a car is passing through a tunnel. The sequence shown in figure E.3 is an example where the baseline segmentation model is not able to detect the car placed in front of the camera. In contrast, both of the methods that combine past frames for the segmentation are able to detect the car for a longer period.

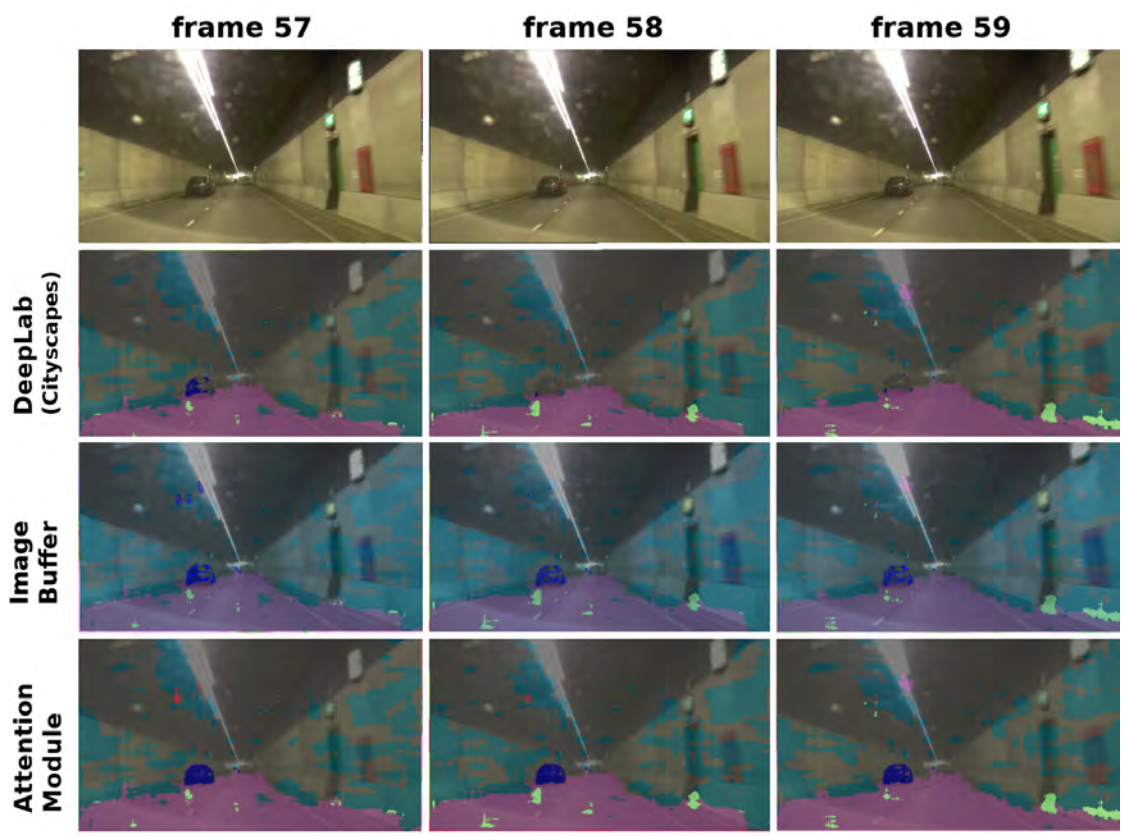

Figure E.3: Figure comparison of the segmentation of three consecutive frames of a car inside in a tunnel using different methods. It can be observed how DeepLab is not able to detect the car during the whole series, while the other two approaches obtain a better detection.

E.4 Driving under the rain

This video [53] presents a driving scenario with heavy rain. As a result, none of the evaluated methods is able to capture accurate information about the road nor the obstacles. Apart from the bad quality of the image (due to the rain), the results can also be explained due to the lack of adverse weather samples in the Cityscapes data-set [5].

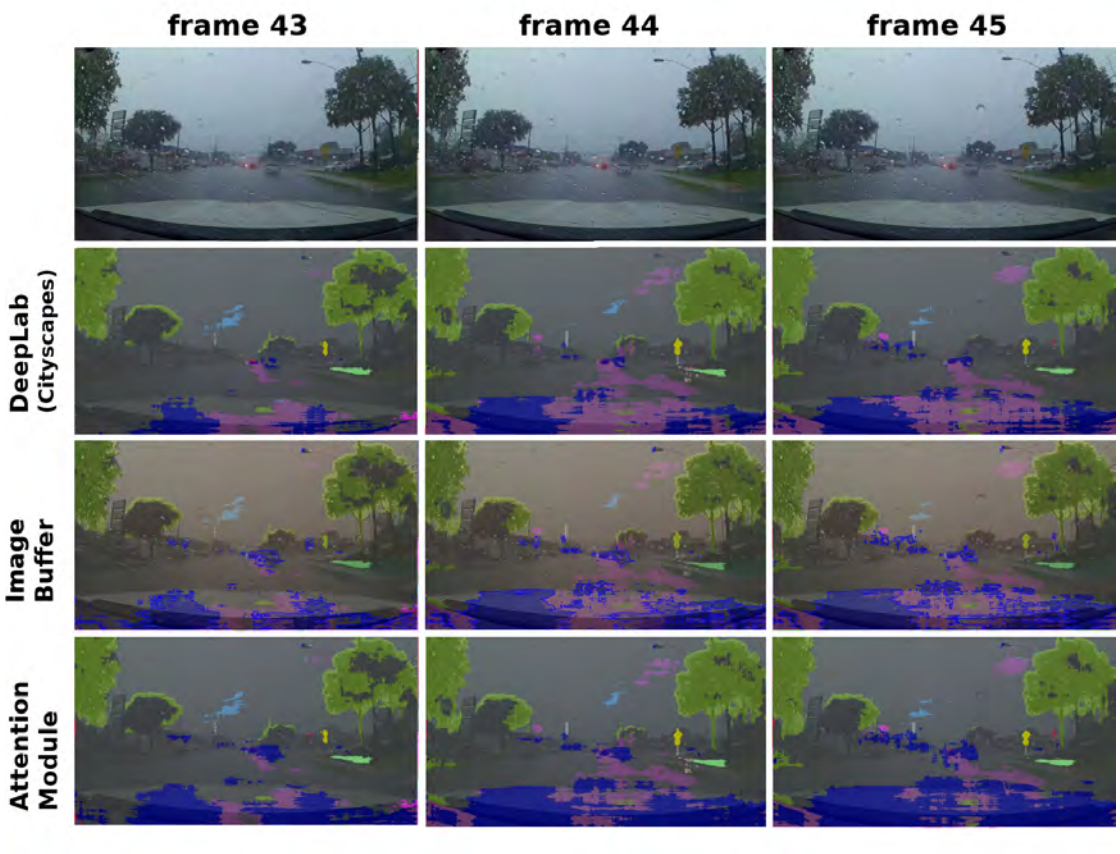

Figure E.4: Figure comparison of the segmentation of three consecutive frames of a heavy rain driving scenario. It can be observed how none of the evaluated methods is able to capture accurate information about the road nor the obstacles.

E.5 Driving in the night

This video [54] evaluates the performance of the methods under poor light conditions. The results show only the area illuminated by the car is able to classify 'correctly' part of the road and the cars on the sides. This shows one of the main disadvantages of using rgb-cameras as the main source of information. However, apart from using a different camera, this bad result could be alleviated by adding more samples that capture poor illuminated scenarios to the data-set.

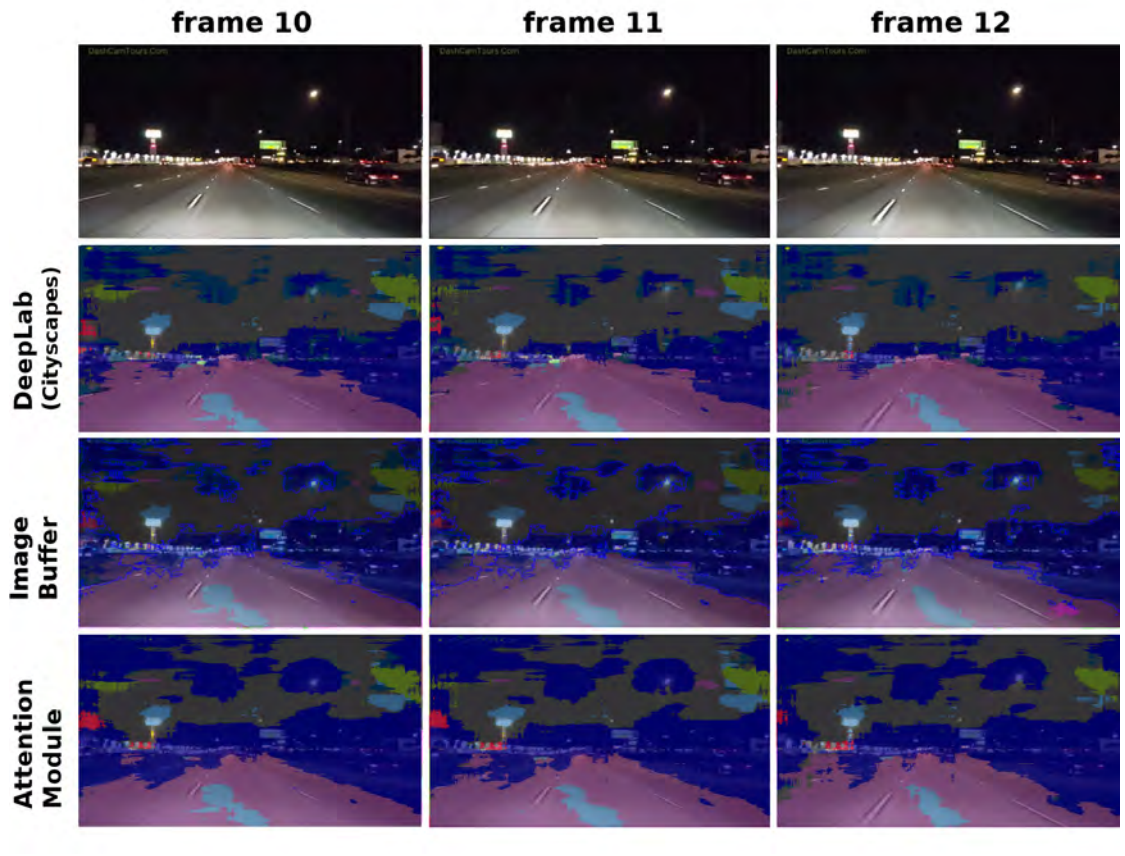

Figure E.5: Figure comparison of the segmentation of three consecutive frames of a night-driving scenario. It can be observed how the baseline model gives the best performance on the segmentation of the road, the rest of the image does not have enough features to make a good segmentation.

E.6 Driving in low sun

This video [54] evaluates the performance of the methods under extreme light conditions, particularly when the sun is low. This is one of the most dangerous moments to drive, due to the angle of the sun and it is expected to be a challenging case-study for the camera and the baseline model. In figure E.6 it can be seen how none of the presented models is able to detect the cyclist in the side of the road (highlighted in green), on top of that the rest of the segmentations are mostly wrong.

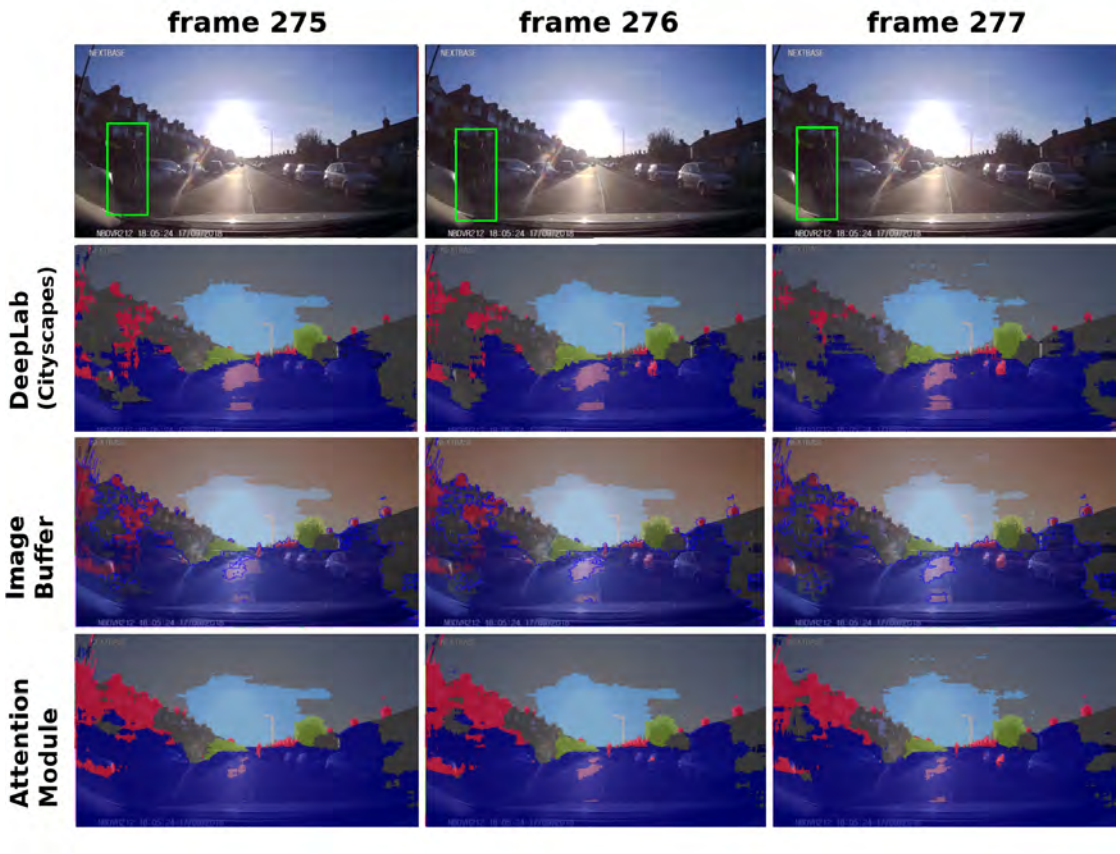

Figure E.6: Figure comparison of the segmentation of three consecutive frames of a scenario driving under low sun. It can be observed how none of the evaluated methods is able to detect the cyclist and how the rest of the labels are mainly wrong.

F Algorithms

F.1 Image Buffer

Algorithm 1 contains the pseudo-code representation of the Image Buffer approach (section 4.1). The target categories are 'person', 'bike', 'rider' and 'car'; although the procedure is flexible to cover different ones.

Algorithm 1: Image Buffer

input : videoSource - A RGB-video source "HxWx3"
output: augSeg - Augmented segmentation map "HxWx3"

```
imageBuffer = zeros(3xHxW)            // Array that stores the "filtered" label-map of frames (t-1), (t-2) and (t-3)
target = ['person', 'bike', 'rider', 'car']              // List of target categories

while videoSource do
    videoFrame = ExtractFrame(videoSource)        // Extract frame from video source
    labelMap = DeepLabv3(videoFrame)             // Baseline label prediction
    augLabels = Overlap(labelMap, imageBuffer)       // Construct the augmented label-map
    augSeg = ToColor(augLabels)                 // Assign colors to each label
    augSeg → OUTPUT                            // Output the augmented seg map

    filteredMap = Filter(labelMap, target)     // Filter the target classes only
    imageBuffer = Update(filteredMap, imageBuffer)     // Update the imageBuffer array:  add recent label-map, remove the oldest
end
```

F.2 Attention Module

Algorithm 2 contains the pseudo-code representation of the Attention Module approach (section 4.2). The target categories are 'person', 'bike', 'rider' and 'car'; although the procedure is flexible to cover different ones.

Algorithm 2: Attention Module

input : videoSource - A RGB-video source "HxWx3"
output: augSeg - Augmented segmentation map "HxWx3"

```
probabilityBuffer = zeros(3xHxWxnumberOfTargets)      // Array that stores the "filtered" probability-map of frames (t-1), (t-2) and (t-3)
target = ['person', 'bike', 'rider', 'car']           // List of target categories

while videoSource do
    videoFrame = ExtractFrame(videoSource)            // Extract frame from video source
    probabilityMap = DeepLabv3(videoFrame, activation='softmax')      // Obtain the probability map
    augLogits = WeightedSum(probabilityMap, probabilityBuffer)        // Construct the augmented logit-map
    augLogits = Treshold(augLogits)  // Threshold the aug.  logits to reduce False Positive Classifications
    augLabels = Argmax(augLogits)                     // Obtain the final labels
    augSep = ToColor(augLabels)                       // Assign colors to each label
    augSeg → OUTPUT                                   // Output the augmented seg map

    filteredProb = Filter(probabilityMap, target)     // Filter the target classes only
    probabilityBuffer = Update(filteredProb, probabilityBuffer)       // Update the probabilityBuffer array:  add recent probability-map, remove the oldest
end
```

G Supplemental material

G.1 Intersection over Union

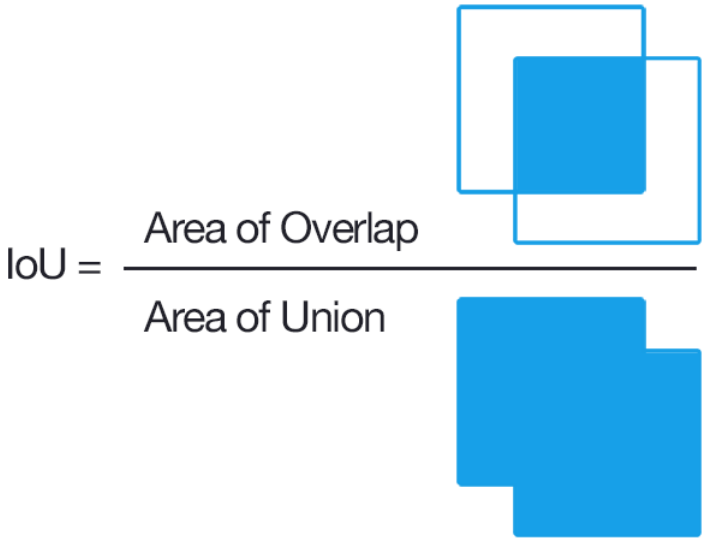

Figure G.1: Figure representation of the Intersection over Union metric. The IoU is a ratio between the area overlap and the area union of the prediction labels with the groundtruth annotation. Image source: [59].

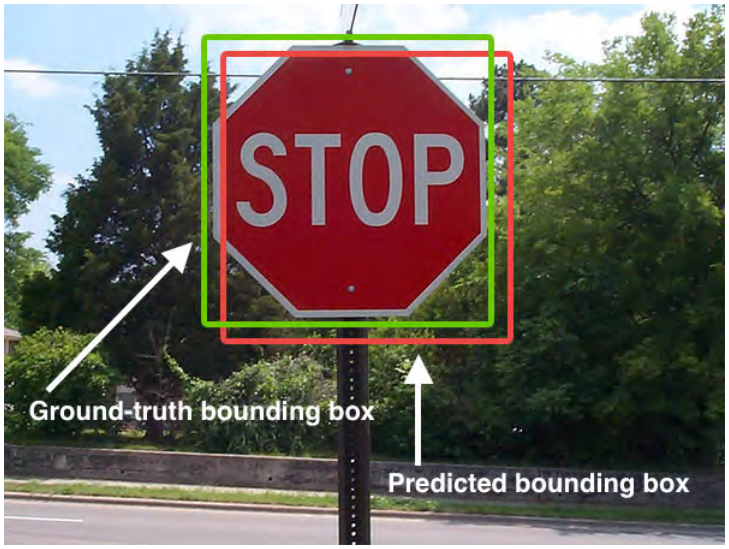

Figure G.2: Illustrative example for the calculation of the IoU. A perfect match between the bounding boxes (or the object shape in case of semantic image segmentation) would result in a IoU equal to 1. Image source: [59].

G.2 Color legend used for annotation

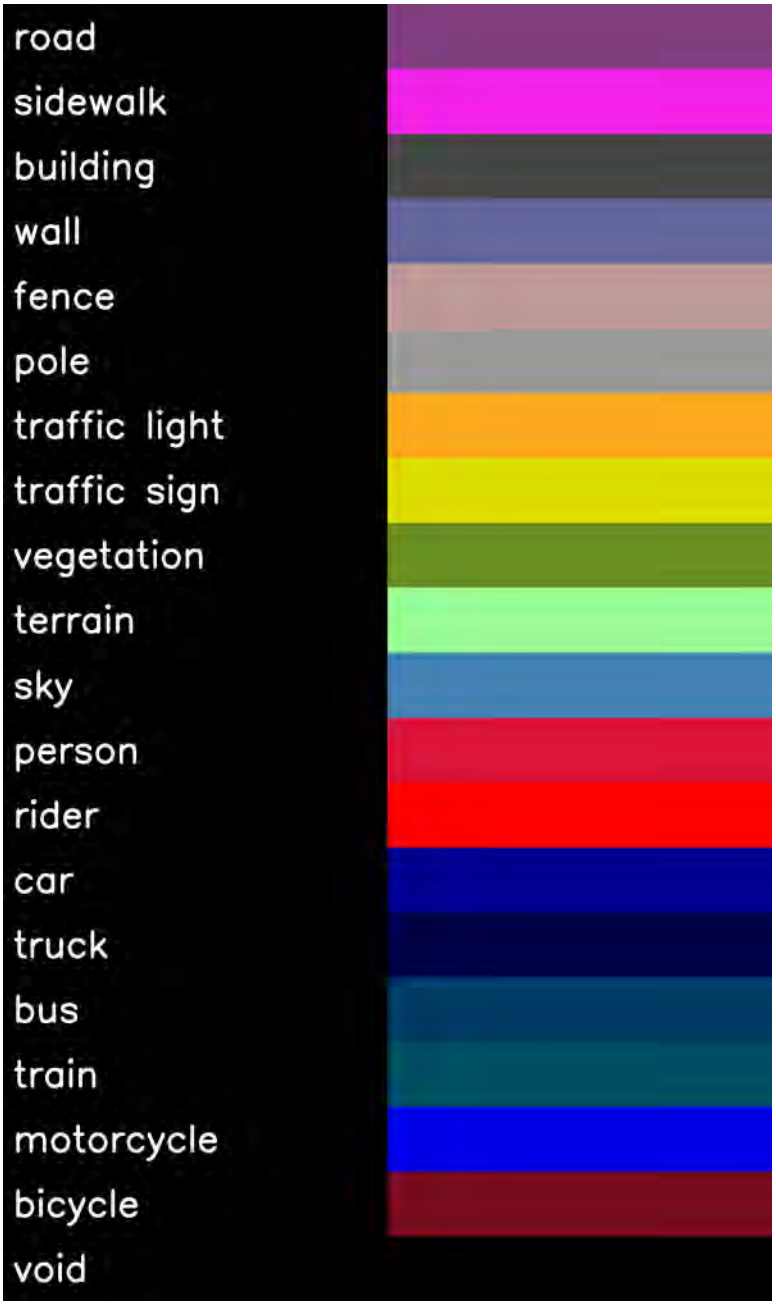

Figure G.3: This is the color palette used by the image segmentation model under study. After all the classification labels are assigned to each of the pixels in the image, a visualization function uses these labels as an index that points to each one of the colors present in this figure, e.g. a pixel labeled with the value '0' will refer to the first color on the list. The visualization function transforms the HxW output image that contains the labels for each pixel (H and W are the dimensions of the input image) into a HxWx3 RGB-image like the one in figure 1.3.

G.3 Scoremaps produced by DeepLabv3

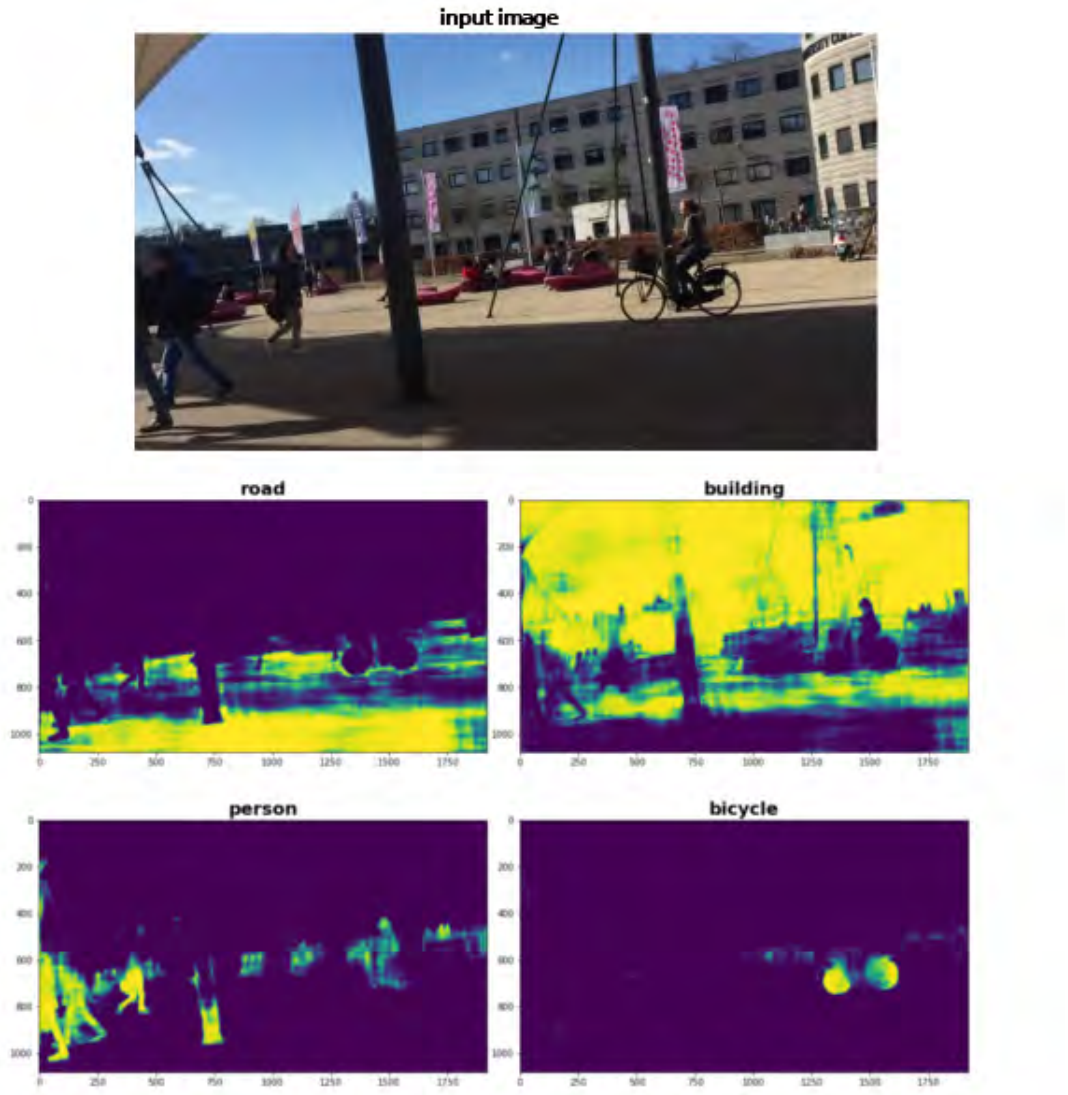

Figure G.4: Scoremaps produced by DeepLabv3 [4] pretrained on Cityscapes dataset [5]. This figure shows the scoremap of 4 different categories for a given input image (section 3.1.4). The highlighted areas for each map represent the areas with a higher confidence for classification. These scoremaps have been normalized to values between 0 and 1 using a softmax function (section 4.2).

G.4 Quadratic loss function Vs Cross-entropy loss function

This section extends on the differences between using a quadratic or a cross-entropy loss function in a classification task. A machine learning problem usually consists of a neural network, a loss function, a dataset (that contains training, testing and evaluation data) and an optimization algorithm.

For the purpose of showing the differences between using the two announced loss functions, the example to be analyzed consists on training a single neuron using gradient descent in a binary-classification problem (example source [56]). Equations G.1 and G.2, define the behavior of a single neuron, the first one creates a relation between the input value (x), a weight and a bias (w and b, trainable parameters); the second one is the activation function and it determines the output value of the neuron, adding non-linearity in the result. Although there are multiple types of activation functions, a sigmoid function will be used in this example (figure G.5).

$$z = wx + b \tag{G.1}$$

$$a = \sigma(z) \tag{G.2}$$

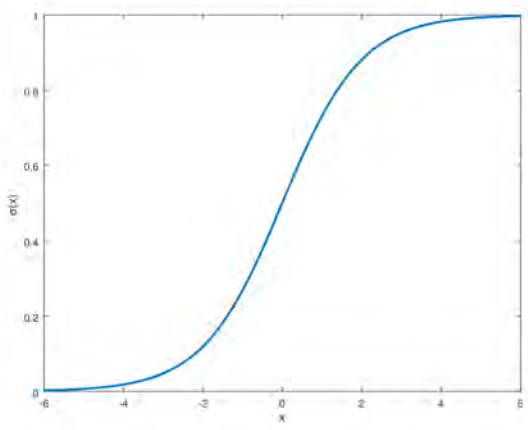

Figure G.5: Graph of a sigmoid function.

The next step is defining the loss function. First, the quadratic loss function will be analyzed, it is is defined as:

$$L = \frac{(y-a)^2}{2} \tag{G.3}$$

where a is the output value of the neuron (or the prediction) and y is the true value for the prediction. Therefore a high discrepancy between the output a and y will result in a high loss. The purpose of the optimization algorithm is to update the values of the weights and bias until a minimum of the loss function is achieved. The gradient descent optimization algorithm calculates the derivative of the loss function with respect to each trainable variable and updates their values in the direction that reduces the loss function. Taking partial derivatives:

$$\frac{\partial L}{\partial w} = -\frac{2}{2}(y-a)\sigma'(z)x = (a-y)\sigma'(z)x \tag{G.4}$$

$$\frac{\partial L}{\partial b} = -\frac{2}{2}(y-a)\sigma'(z) = (a-y)\sigma'(z) \tag{G.5}$$

After some reorganization:

$$\frac{\partial L}{\partial w} = (a-y)\sigma'(z)x \tag{G.6}$$

$$\frac{\partial L}{\partial b} = (a-y)\sigma'(z) \tag{G.7}$$

It can be observed how both of the equations are multiplied by the derivative of the activation function. As an example, in a situation where the true value is equal to 1, the gradients will be small both when the model is completely wrong (when the output is 0) and when the predicted label is close to the truth (output is close to 1), this can produce slow learning when the initialization of the model leads to results that are far from the ground-truth label. However, this can be solved by using a different loss function. The cross-entropy loss function (equation G.8):

$$L = -(y \ln a + (1-y)\ln(1-a)) \tag{G.8}$$

And the partial derivatives of this loss function with respect to weights and bias:

$$\begin{aligned}\frac{\partial L}{\partial w} &= -(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)})\frac{\partial \sigma}{\partial w} \\ &= -(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)})\sigma'(z)x\end{aligned} \tag{G.9}$$

$$\frac{\partial L}{\partial b} = \frac{\sigma'(z)x}{\sigma(z)(1-\sigma(z))}(\sigma(z)-y) \tag{G.10}$$

After some reorganization:

$$\frac{\partial L}{\partial w} = x(\sigma(z)-y) \tag{G.11}$$

$$\frac{\partial L}{\partial b} = \sigma(z)-y \tag{G.12}$$

It can be observed how, in contrast to the quadratic loss function, the gradients of the cross-entropy loss function (equations G.11 and G.12) are proportional to the error of the prediction, speeding up the learning process when the predicted value is further from the true label. This is why the cross-entropy loss function is used in classification tasks.

Bibliography

[1] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 834–848, 2016.

[2] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6230–6239, 2016.

[3] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," *ArXiv*, vol. abs/1704.08545, 2017.

[4] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *ArXiv*, vol. abs/1706.05587, 2017.

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] D. Loesche, "How americans commute to work," 2017.

[7] N. McCarthy, "Fewer americans are driving to work," 2018.

[8] P. McNamara, "How did audi make the first car with level 3 autonomy?," 2017.

[9] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*, 1999.

[10] Cityscapes-Team, "Cityscapes dataset," 2019.

[11] K. P. Murphy, "Machine learning - a probabilistic perspective," in *Adaptive computation and machine learning series*, 2012.

[12] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *NIPS*, 2001.

[13] J. Lasserre, C. M. Bishop, J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. M. Smith, and M. A. West, "Generative or discriminative? getting the best of both worlds," 2007.

[14] P. M. Domingos and M. J. Pazzani, "Beyond independence: Conditions for the optimality of the simple bayesian classifier," in *ICML*, 1996.

[15] V. Powell, "Markov chains," 2014.

[16] D. Conklin, "Music generation from statistical models," *In AISB Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pp. 30–35, 2003.

[17] D. Nilsson and C. Sminchisescu, "Semantic video segmentation by gated recurrent flow propagation," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6819–6828, 2016.

[18] A. Kundu, V. Vineet, and V. Koltun, "Feature space optimization for semantic video segmentation," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3168–3175, 2016.

[19] B. Liu and X. He, "Multiclass semantic video segmentation with object-level active inference," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4286–4294, 2015.

[20] O. Miksik, D. Munoz, J. A. Bagnell, and M. Hebert, "Efficient temporal consistency for streaming video scene analysis," *2013 IEEE International Conference on Robotics and Automation*, pp. 133–139, 2013.

[21] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649, 2017.

[22] A. Bewley, Z. Ge, L. Ott, F. T. Ramos, and B. Upcroft, "Simple online and realtime tracking," *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, 2016.

[23] R. Restrepo, "What is semantic segmentation?," 2017.

[24] X. Li, Z. Liu, P. Luo, C. C. Loy, and X. Tang, "Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6459–6468, 2017.

[25] H. Fan, X. Mei, D. V. Prokhorov, and H. Ling, "Multi-level contextual rnns with attention model for scene labeling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, pp. 3475–3485, 2016.

[26] G. Lin, C. Shen, I. D. Reid, and A. van den Hengel, "Efficient piecewise training of deep structured models for semantic segmentation," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3194–3203, 2015.

[27] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3309–3318, 2016.

[28] G. Ghiasi and C. C. Fowlkes, "Laplacian pyramid reconstruction and refinement for semantic segmentation," in *ECCV*, 2016.

[29] G. Lin, A. Milan, C. Shen, and I. D. Reid, "Refinenet : Multipath refinement networks with identity mappings for high-resolution semantic segmentation," 2016.

[30] X. Li, Z. Jie, W. Wang, C. Liu, J. Yang, X. Shen, Z. L. Lin, Q. Chen, S. Yan, and J. Feng, "Foveanet: Perspective-aware urban scene parsing," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 784–792, 2017.

[31] J. Krapac, I. Kreso, and S. Segvic, "Ladder-style densenets for semantic segmentation of large natural images," *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 238–245, 2017.

[32] X. Jin, X. Li, H. Xiao, X. Shen, Z. L. Lin, J. Yang, Y. Chen, J. Dong, L. Liu, Z. Jie, J. Feng, and S. Yan, "Video scene parsing with predictive feature learning," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5581–5589, 2016.

[33] R. Zhang, S. Tang, M. Lin, J. Li, and S. Yan, "Global-residual and local-boundary refinement networks for rectifying scene parsing predictions," in *IJCAI*, 2017.

[34] R. Zhang, S. Tang, Y. Zhang, J. Li, and S. Yan, "Scale-adaptive convolutions for scene parsing," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2050–2058, 2017.

[35] F. Shen, R. Gan, S. Yan, and G. Zeng, "Semantic segmentation via structured patch prediction, context crf and guidance crf," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5178–5186, 2017.

[36] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. W. Cottrell, "Understanding convolution for semantic segmentation," *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1451–1460, 2017.

[37] R. Gadde, V. Jampani, and P. V. Gehler, "Semantic video cnns through representation warping," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4463–4472, 2017.

[38] Z. Wu, C. Shen, and A. van den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *ArXiv*, vol. abs/1611.10080, 2016.

[39] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *CoRR*,

vol. abs/1412.7062, 2014.

[40] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2014.

[41] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2009.

[42] D. Scherer, A. C. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *ICANN*, 2010.

[43] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015.

[44] K. Grauman and T. Darrell, "The pyramid match kernel: discriminative classification with sets of image features," *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, pp. 1458–1465 Vol. 2, 2005.

[45] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 2169–2178, 2006.

[46] I. Hadji and R. Wildes, "What do we understand about convolutional networks?," 03 2018.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[48] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *ECCV (1)*, pp. 44–57, 2008.

[49] T. ScharwÃd'chter, "Daimler urban segmentation,"

[50] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[51] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Computer Vision and Pattern Recognition*, 2016.

[52] Kroonenberg, "A drive downtown amsterdam," 2019.

[53] TexasHighDef, "A drive downtown amsterdam," 2018.

[54] D. C. Tours, "Night driving on california freeway. no music.," 2017.

[55] konradc12, "Driving in low sun," 2018.

[56] M. Nielsen, "Neural networks and deep learning," 2019.

[57] ML-cheatsheet, "Loss functions," 2018.

[58] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W. chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *ArXiv*, vol. abs/1506.04214, 2015.

[59] A. Rosebrock, "Intersection over union (iou) for object detection," 2016.